МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное автономное
образовательное учреждение высшего образования
«Южно-Уральский государственный университет
(национальный исследовательский университет)»

Высшая школа электроники и компьютерных наук
Кафедра «Электронные вычислительные машины»

РАБОТА ПРОВЕРЕНА
Рецензент
_____
«___»_____ 2019 г.

ДОПУСТИТЬ К ЗАЩИТЕ
Заведующий кафедрой ЭВМ
_____ Г.И. Радченко
«___»_____ 2019 г.

Разработка подсистемы отчетности и информирования клиентов для CRM-системы стоматологической клиники

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА К ВЫПУСКНОЙ КВАЛИФИКАЦИОННОЙ РАБОТЕ

Руководитель работы,
к.т.н., доцент каф. ЭВМ
_____Е.С. Ярош
«___»_____2019 г.

Автор работы,
студент группы КЭ-452
_____А.А. Шумакова
«___»_____2019 г.

Нормоконтролёр,
ст. преп. каф. ЭВМ
_____ С.В. Сяськов
«___»_____2019 г.

Челябинск-2019

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное автономное
образовательное учреждение высшего образования
«Южно-Уральский государственный университет
(национальный исследовательский университет)»
Высшая школа электроники и компьютерных наук
Кафедра «Электронные вычислительные машины»

УТВЕРЖДАЮ
Заведующий кафедрой ЭВМ
_____ Г.И. Радченко
«___»_____2019 г.

ЗАДАНИЕ
на выпускную квалификационную работу бакалавра
студенту группы КЭ-452
Шумаковой Анастасии Александровне
обучающемуся по направлению
09.03.01 «Информатика и вычислительная техника»

1. **Тема работы:** «Разработка подсистемы отчетности и информирования клиентов для CRM-системы стоматологической клиники» утверждена приказом по университету от 25 апреля 2019 г. №899

2. **Срок сдачи студентом законченной работы:** 1 июня 2019 г.

3. **Исходные данные к работе:**

– СУБД Firebird 2.5. (https://firebirdsql.org);

– операционная система не ниже Windows 7.

Обеспечить основной функционал приложения:

– отправку СМС уведомления о записи на прием;
– отправку СМС уведомления за день до приема;
– отправку СМС уведомления за 3 часа до приема;
– поддержание актуальности данных о приемах пациентов для уведомления пациентов;

– интерфейс для работы администраторов клиники, Call-центра и экономистов;
– модифицированную систему отчетности.

4. **Перечень подлежащих разработке вопросов:**
   – выбор сервиса отправки СМС;
   – выбор инструментов для разработки системы;
   – разработка архитектуры системы;
   – разработка модуля обработки данных;
   – разработка модуля отправки СМС;
   – разработка модуля администрирования и отчетности;
   – тестирование разработанного программного обеспечения.

   Дата выдачи задания: 1 декабря 2018 г.

Руководитель работы _____/*Е.С. Ярош*/

Студент _____/*А.А. Шумакова*/

КАЛЕНДАРНЫЙ ПЛАН

Этап	Срок сдачи	Подпись руководителя
Введение и выбор сервиса отправки СМС	01.01.2019	
Разработка модели, проектирование	10.01.2019	
Разработка модуля обработки данных	10.02.2019	
Разработка модуля отправки СМС	10.03.2019	
Разработка модуля администрирования и отчетности	10.04.2019	
Тестирование, отладка, эксперименты	10.05.2019	
Оформление пояснительной записки	20.05.2019	
Подготовка презентации и доклада	11.06.2019	

Руководитель работы _____/*Е.С. Ярош*/

Студент _____/*А.А. Шумакова*/

Аннотация

А.А. Шумакова. Разработка подсистемы отчетности и информирования клиентов для CRM-системы стоматологической клиники. – Челябинск: ФГАОУ ВО «ЮУрГУ (НИУ)», ВШЭКН; 2019, 196 с., 43 ил., библиогр. список – 6 наим.

В рамках выпускной квалификационной работы разработана система, включающая в себя три модуля: модуль обработки данных, модуль отправки СМС и модуль администрирования и отчетности. Произведен выбор сервиса отправки СМС и языка программирования. Выполнено проектирование архитектуры системы. Построены и проанализированы диаграммы потоков данных, что позволило объединить ряд потоков в отдельные нити. Разработанное программное обеспечение протестировано и введено в эксплуатацию. В результате его внедрения снижена нагрузка на работников call-центра и уменьшены простои в работе стоматологической клиники.

ОГЛАВЛЕНИЕ

ВВЕДЕНИЕ

Забота о клиентах — это основа успешной деятельности любой компании. Необходимо понимать, что в медицинских организациях это особенно важно, ведь любые недочеты в работе системы могут привести к ухудшению здоровья человека. Если говорить о стоматологической клинике, то своевременное реагирование на все изменения в расписании приемов и наличие СМС-информирования клиентов повышает лояльность и качество предоставляемого сервиса. Учет этих компонентов при создании системы в сумме дает уменьшение простоев в расписании, тем самым увеличивая прибыль компании, а также повышает удовлетворенность клиентов качеством обслуживания. Автоматизированная система отчетности позволяет сотрудникам проводить быстрый и глубокий анализ бизнес-процессов компании.

Большие компании для работы с клиентами используют CRM-системы. Это прикладное программное обеспечение для организаций, предназначенное для автоматизации стратегий взаимодействия с заказчиками (клиентами). Ее элементы используют как для работы с клиентами, так и для организации работы внутри компании.

Для улучшения работы компании было принято решение дополнить имеющуюся CRM-систему. Так как существующий функционал не полностью отвечал требованиям заказчика, возникла необходимость создания специального модуля, который расширяет возможности существующей CRM-системы. Он будет отвечать за СМС-информирование клиентов и за систему отчетности.

1. АНАЛИЗ РЕШАЕМОЙ ЗАДАЧИ

1.1. Актуальность задачи

В исходной версии CRM-системы встроенная система СМС-информирования не отвечала требованиям заказчика в полной мере. Часто возникали ситуации, когда клиенты забывали о приеме, а call-центр компании не всегда успевал всех обзванивать и напоминать. Также возникали ситуации, когда клиенты записывались на прием, а потом передумывали и не приходили. Из-за всего этого возникали простои, что плохо сказывалось на прибыли компании.

В целях улучшения качества обслуживания клиентов было принято решение создать 3 модуля.

Первый модуль – модуль обработки данных. Данный модуль обрабатывает данные, поступившие в расписание, формирует таблицы для дальнейшего использования их в модулях, а также записывает подтверждения приемов в расписание.

Второй модуль – модуль по отправке СМС уведомлений. Данный модуль работает с данными, созданными ранее модулем обработки данных. Он работает в зависимости от режима, выставленного в третьем модуле, а также считывает статусы подтверждения приемов.

Третий модуль – модуль администрирования и отчетности. Данный модуль позволяет выставлять режимы работы отправки СМС во втором модуле, а также в данном модуле реализован функционал формирования отчетов.

1.2. Цель выпускной квалификационной работы

Цель представленной выпускной квалификационной работы заключается в разработке системы отчетности и информирования клиентов для CRM-системы стоматологической клиники. Система предназначена для автоматизации стратегий взаимодействия с клиентами компании, занимающейся стоматологической практикой.

Для достижения поставленной цели, необходимо решить следующие задачи:

1. Выбрать сервис отправки СМС.
2. Выбрать инструменты для разработки системы.
3. Разработать архитектуру системы.
4. Разработать модуль обработки данных.
5. Разработать модуль отправки СМС.
6. Разработать модуль администрирования и отчетности.

1.3. Анализ основных технологических решений

1.3.1. Выбор языка программирования

Перед началом разработки приложения необходимо определиться на каком языке его разрабатывать. Это очень важный вопрос, потому что от языка зависит то, как быстро, качественно и комфортно будет написано приложение.

Необходимо написать три приложения, три модуля, два из которых будут крутиться на сервере двадцать четыре часа в сутки, а третье является графическим клиентом для удобного представления информации и управления модулями.

Для сравнения возьмем 3 самых популярных языка программирования для десктопных приложений:

1. C#.
2. Python.
3. Java.

Все три этих языка на сегодняшний момент очень популярны среди разработчиков, их любят, уважают и развивают.

У заказчика было условие, что приложения должны быть развернуты на сервере с системой Windows, так как сервера компании используют данную систему.

Критерии сравнения языков:

1. Производительность.

2. Многопоточность.

3. Скорость разработки.

4. Удобство создания графического интерфейса.

5. Поддержка.

1.3.1.1. Производительность

Все три языка имеют достаточно высокую производительность на сегодняшний день, они могут запускаться на Windows, однако язык C# имеет более глубокую интеграцию в операционную систему. Платформа .Net с каждым годом развивается, что хорошо отображается на производительности и оптимизации компиляции программного кода.

1.3.1.2. Многопоточность

Данный критерий относится ко всем языкам, так как все три языка умеют работать в многопоточном режиме, для этого они используют класс Thread, который позволяет максимально эффективно использовать время процессора и раскрывать его потенциал.

1.3.1.3. Скорость разработки

В данном критерии побеждает язык Python, так как его лаконичный и минималистичный синтаксис привлекает многих программистов. В случае данного проекта сроки реализации были не так сжаты и приоритет выбора в данном критерии падает на программиста.

1.3.1.4. Удобство создания графического интерфейса

В данном критерии безусловно побеждает C# со своими Windows формами, которые позволяют очень быстро, красиво и гибко настроить и запрограммировать форму под нужды конкретного проекта. Конечно, GUI десктопа можно писать на Java и Python, но это будет немного более трудозатратно по времени и удобству разработки.

1.3.1.5. Поддержка

Безусловно в данном проекте очень важна поддержка языка и регулярные обновления. Поскольку C# разработан и работает в экосистеме Windows у него очень большая поддержка, как со стороны сообщества, так и со стороны экспертов самой компании Microsoft. Java тоже имеет очень большое сообщество, однако, если мы имеем дело с Windows, то у C# здесь имеется преимущество. Если говорить о Python, то здесь хочется сказать, что этот язык с открытым кодом и развитие идет только со стороны сообщества, которое не может гарантировать тот же уровень поддержки.

1.3.1.6. Вывод

Подводя итог хочется отметить, что все три языка на сегодняшний день очень хорошо развиты по-своему, если конкретно говорить про данный проект и учесть все условия, требования, временные возможности, предыдущий опыт разработки программиста и требуемую поддержку, то выбор падает на язык программирования C#. В данной работе он будет самым сбалансированным языком, подходящим для реализации всего функционала, именно его и будем использовать.

1.3.2. Выбор сервиса отправки СМС

Так как каждый сторонний сервис, подключаемый к проекту, нацелен на выполнение конкретной задачи, необходимо сначала сосредоточиться на возможностях.

Критерии выбора сервиса для отправки СМС:

1. Удобство интеграции.
2. Тарифы.
3. Способы оплаты.
4. Поддержка клиентов.
5. Пользовательский интерфейс.

1.3.2.1. Удобство интеграции

Первый и важный пункт: насколько удобно работать с сервисом и интегрировать его в любой проект. В таблице 1.1 представлен выбор сервиса.

Таблица 1.1 - Удобство интеграции

Сервис	Https запрос	Реализация на C#	Полная документация	Краткая документация
SMS Aero [1]	✓	✓		✓
SMSC [2]	✓	✓		✓
SMS.ru [3]	✓	✓		✓
SMS GOROD [4]	✓			✓
WEBSMS [5]	✓		✓	
REDSMS [6]	✓			✓

1.3.2.2 Тарифы

Для сравнения тарифов были выбраны 4 популярных компании Челябинской области, а также другие провайдеры. В месяц отправляется около 10 000 СМС уведомлений. Выбор тарифа приведен в таблице 1.2.

Таблица 1.2 - Тарифы

Сервис	Мегафон, руб.	МТС, руб.	Билайн, руб.	TELE2, руб.	Другое, руб.
SMS Aero [1]	2,32	2,25	2,37	2,48	2,40
SMSC [2]	2,00	1,62	2,10	2,10	2,18
SMS.ru [3]	1,81	1,50	1,90	1,83	2,01
SMS GOROD [4]	2,05	1,72	2,22	2,22	2,22
WEBSMS [5]	2,30	2,00	2,45	2,30	2,45
REDSMS [6]	1,97	1,62	2,12	2,07	2,17

1.3.2.3. Способы оплаты

Для удобства использования сервиса желательно, чтобы он имел несколько способов оплаты. В таблице 1.3 приведены способы оплаты.

Таблица 1.3 - Способы оплаты

Сервис	Яндекс Деньги	Webmoney	Карты	Безнал	PayPal
SMS Aero [1]	✓	✓	✓	✓	✓
SMSC [2]	✓	✓	✓	✓	✓
SMS.ru [3]	✓	✓	✓	✓	✓
SMS GOROD [4]			✓	✓	
WEBSMS [5]	✓	✓	✓	✓	✓
REDSMS [6]	✓	✓	✓	✓	

1.3.2.4. Поддержка клиентов

Немаловажную роль играет техническая поддержка и её компетентность, скорость реагирования на запрос и оперативность устранения проблемы. В таблице 1.4 представлен выбор сервиса по критерию поддержка клиентов.

Таблица 1.4 - Поддержка клиентов

Сервис	Чат	Форум	Почта	Скайп	Телефон
SMS Aero [1]	✓		✓	✓	✓
SMSC [2]	✓		✓	✓	✓
SMS.ru [3]		✓	✓		✓
SMS GOROD [4]			✓	✓	✓
WEBSMS [5]			✓		✓
REDSMS [6]	✓		✓		✓

1.3.2.5. Пользовательский интерфейс

Одним из важных критериев является пользовательский интерфейс, так как через него будет осуществляться ежедневный просмотр работы программы. Цель просмотра – убедиться, что программа работает и СМС уведомления отправляются.

Дизайн интерфейса — дело вкуса, главное — чтобы интерфейс был интуитивно понятным, и не пришлось долго разбираться с навигацией по сайту.

На рисунках 1.1 - 1.6 показаны экранные формы рассмотренных программных средств.

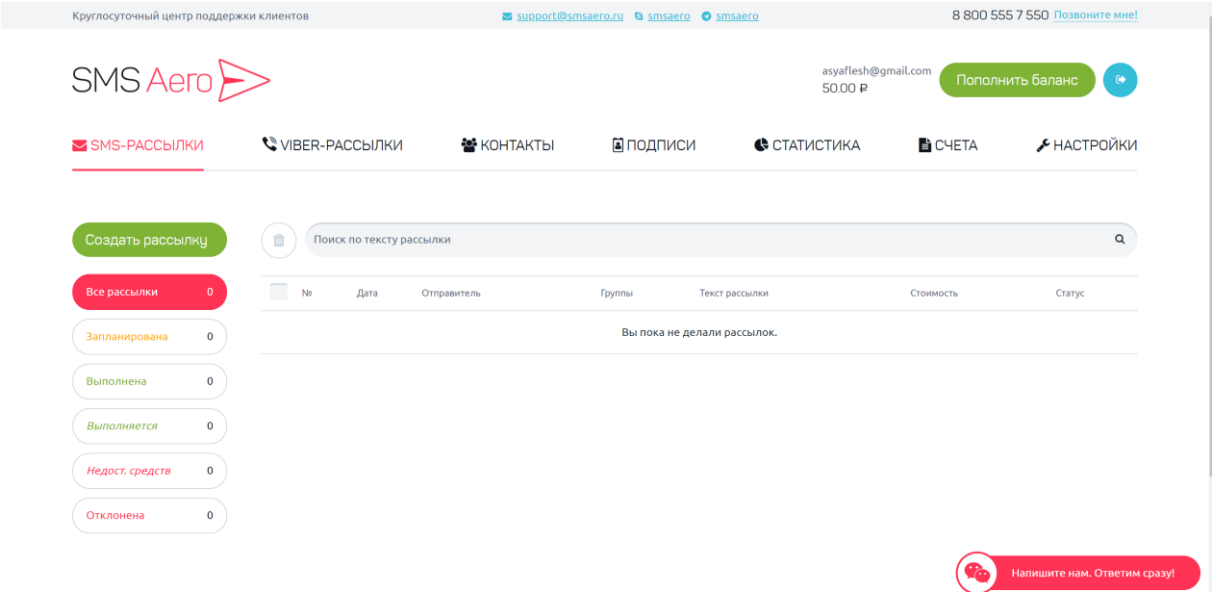

Рисунок 1.1 – Экранная форма SMS Aero

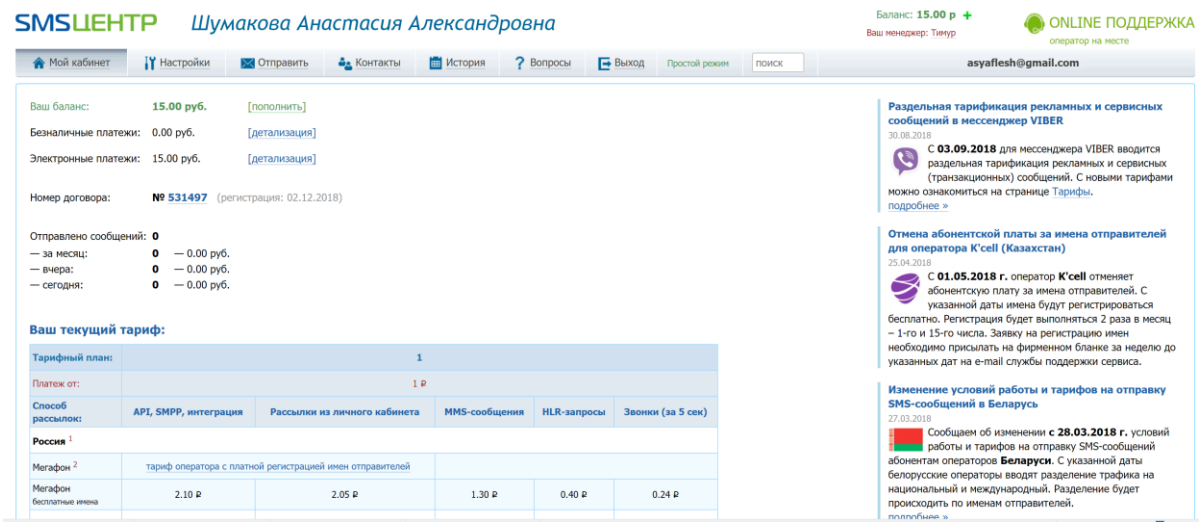

Рисунок 1.2 – Экранная форма SMSC

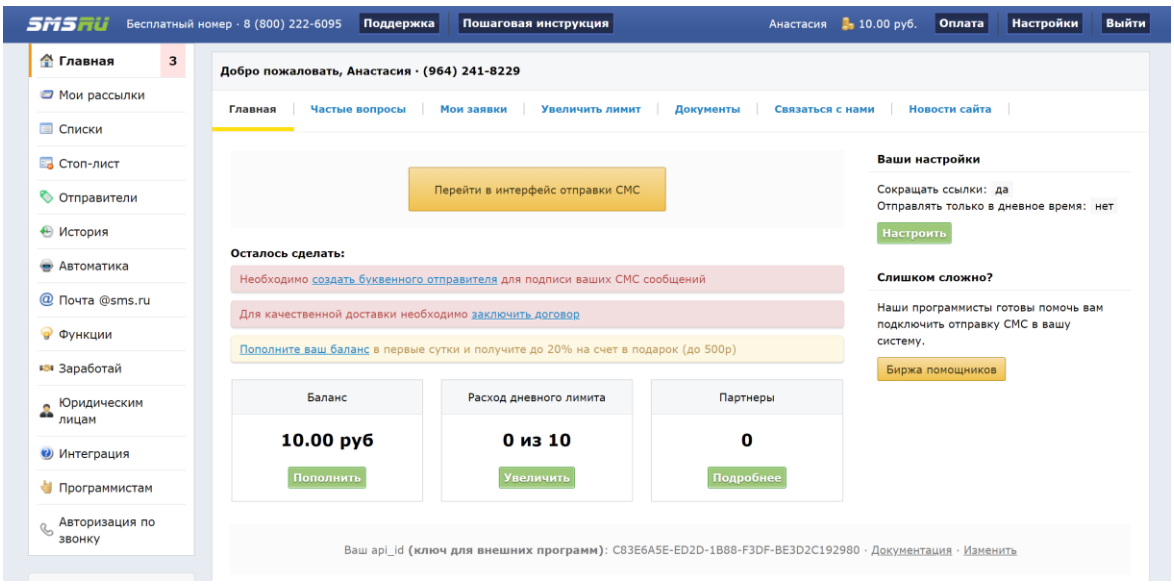

Рисунок 1.3 – Экранная форма SMS.ru

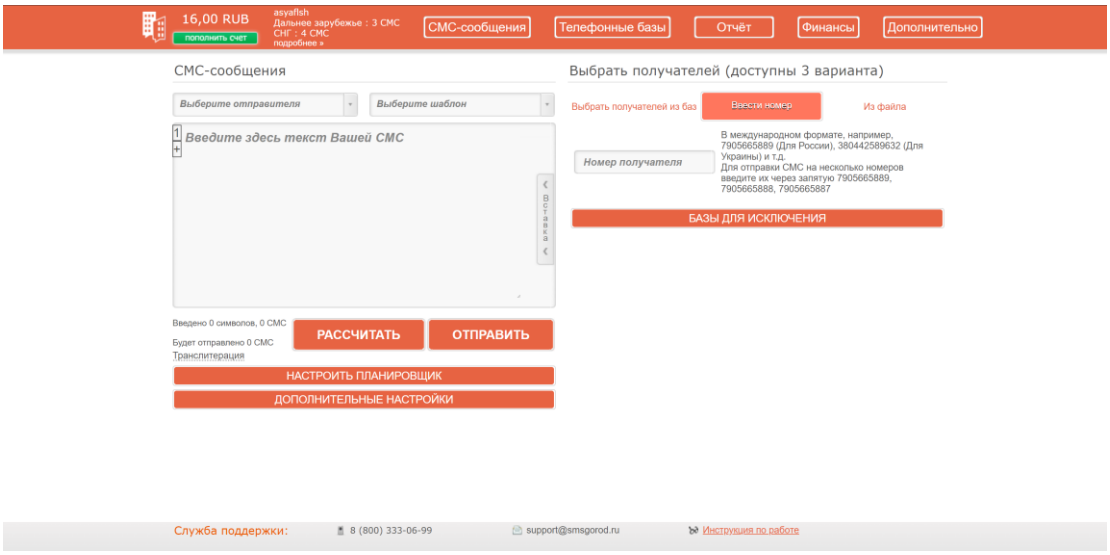

Рисунок 1.4 – Экранная форма SMS GOROD

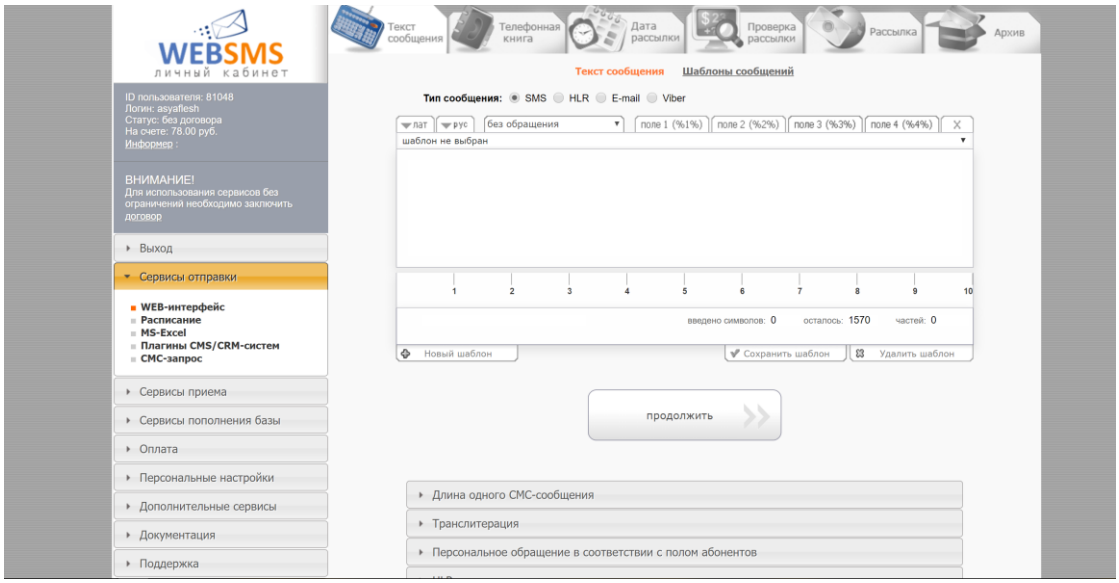

Рисунок 1.5 – Экранная форма WEBSMS

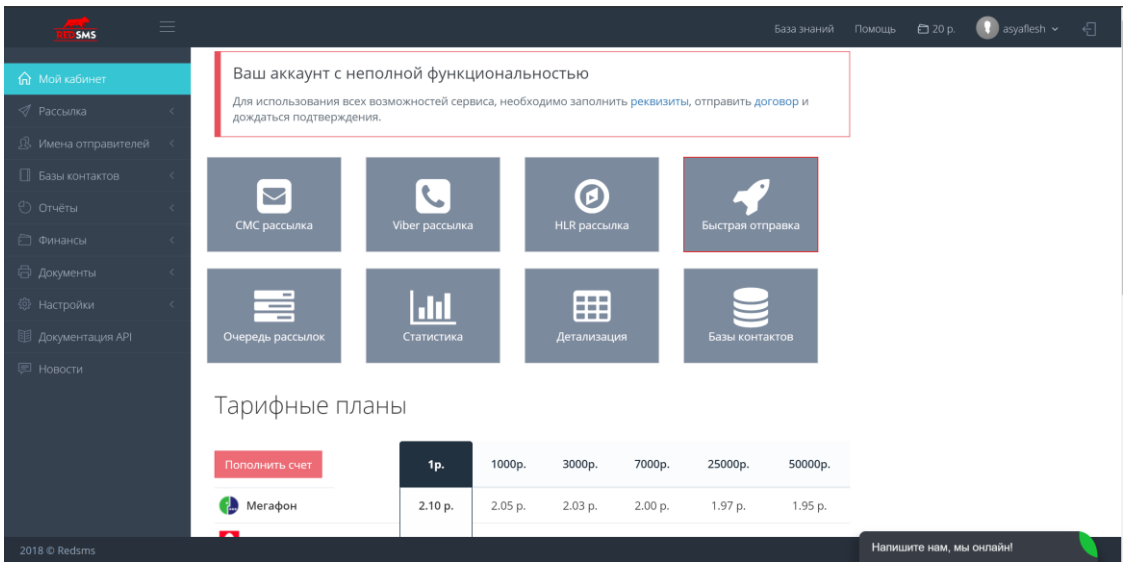

Рисунок 1.6 – Экранная форма REDSMS

1.3.2.6. Вывод

Исходя из критериев выбора сервиса по отправке СМС уведомлений было принято решение выбрать SMS.ru.

У этого сервиса есть вся необходимая информация для удобной интеграции в проект, минимальный тариф, пользовательский интерфейс удобен, в поддержке имеется форум. СМС уведомления приходят быстро, техподдержка оперативная, персонал вежливый.

2. ОПРЕДЕЛЕНИЕ ТРЕБОВАНИЙ

Для реализации данной системы необходим следующий набор подсистем:

1. Модуль обработки данных. Приложение должно обрабатывать поступившие в расписание данные, поддерживать их актуальность, формирует таблицы для дальнейшего использования их в модулях, а также записывать подтверждения приемов в расписание.

2. Модуль по отправке СМС уведомлений. Приложение должно работать с данными, созданными ранее модулем обработки данных. Оно должно работать в зависимости от режима, выставленного в модуле администрирования и отчетности, а также считывать статусы подтверждения приемов.

3. Модуль администрирования и отчетности. Данный модуль должен позволять выставлять режимы работы отправки СМС во втором модуле, а также предоставлять функционал формирования отчетов.

4. База данных Firebird 2,5. База данных обеспечивает хранение данных, для дальнейшей обработки.

2.1. Функциональные требования

Для того, чтобы выделить функциональные требования к системе построим диаграмму прецедентов. Она отображает события, возникающие в системе с точки зрения пользователя. Диаграмма прецедентов в нотации UML изображена на рисунке 2.1.

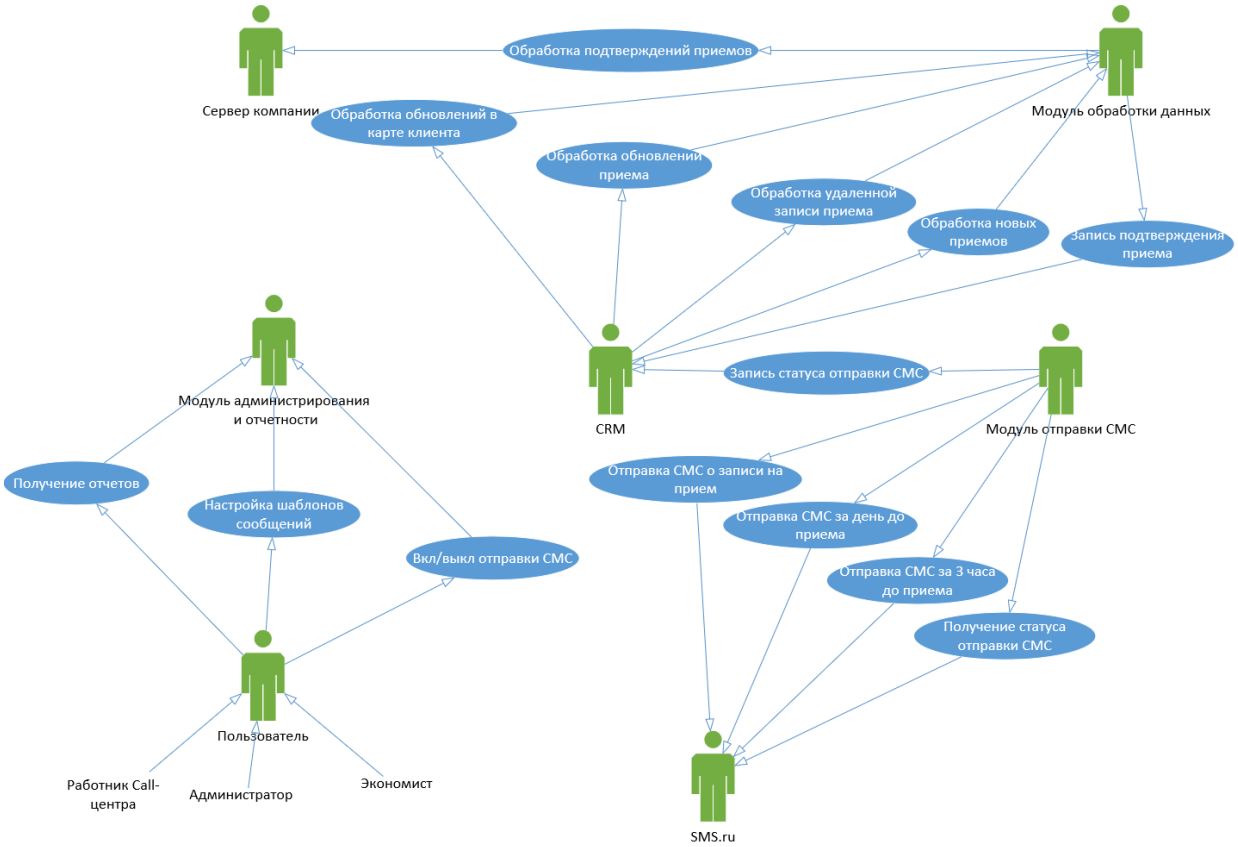

Рисунок 2.1 – Диаграмма вариантов использования системы

Диаграмма используется для отражения прагматики разрабатываемой системы, описания структуры объектов, из которых она состоит, их взаимосвязи

и атрибутов. Роли распределены между пользователем, модулем администрирования и отчетности, SMS.ru, модулем отправки СМС, CRM системой компании, модулем обработки данных, сервером компании. В овалы, в свою очередь, заключены прецеденты использования разрабатываемой системы.

Пользователь взаимодействует только с модулем администрирования и отчетности. Вся остальная часть системы существует без участия пользователя.

Модуль обработки данных взаимодействует с сервером компании для обработки подтверждения приема и с CRM системой компании для записи подтверждения приема в расписание. С CRM системы компании на модуль поступают данные для дальнейшей обработки и добавления в базу данных или обновления/удаления данных уже имеющихся в базе, чтобы данные всегда были актуальны.

Модуль отправки СМС взаимодействует, в основном, с SMS.ru для осуществления отправки СМС уведомлений пациентам, а также для получения статуса подтверждения отправки. После этого модуль взаимодействует с CRM системой компании, куда отправляет статусы подтверждения.

2.2. Нефункциональные требования

2.2.1. Требования к системе в целом

1. Обеспечить интерфейс для работы администраторов клиники, Call-центра и экономистов.

2. Обеспечить актуальность данных о приемах пациентов для уведомления пациентов.

3. Обеспечить сервис SMS-рассылок для пациентов с целью:

– подтверждения записи на прием;
– напоминания о дате и времени приема, за один день до приема и за 3 часа до приема.

Учесть согласие пациента на СМС рассылку.

4. Модифицировать систему отчетности:

– «Большой отчет» – это отчет, который должен выгружаться каждый день после отправки СМС за день до приема и должен содержать информацию о пациентах, их приеме, статусах доставки их СМС, о подтверждении приема, статусе пациента.
– «Статистический отчет» — это отчет о всех отправленных СМС и статистике подтверждений приемов и их посещений.
– «Универсальный отчет» – это отчет, в котором можно настраивать, по каким полям делать выборку: выборка по ИНП (идентификатор номера прикрепления) – выбирает все приемы, связанные с пациентом, статусы отправки СМС, подтверждения приема и посещение; выборка по дате приема – выбирает всех пациентов, у которых был прием в промежутке между выбранными датами и выводит информацию о пациентах, статусы отправки СМС, подтверждения приемов и посещение.

2.2.2. Требования квалификации персонала

Для работы с системой пользователь должен иметь базовые навыки работы с компьютером.

2.2.3. Требования к надежности и безопасности

Доступ к системе должны иметь только зарегистрированные пользователи в соответствии с правами доступа.

Права доступа экономиста:

Права доступа администратора – просмотр отчета по отправленным СМС сообщениям только для своей клиники.

Права доступа работника Call-центра - просмотр отчета по отправленным СМС сообщениям только для всех клиник.

Обеспечение информационной безопасности в рамках закона о персональных данных и закона о коммерческой тайне обеспечивается

общесистемными средствами CRM-системы и не входит в задачи данной разработки.

2.2.4. Требования к платформе

Система должна работать на персональном компьютере с операционной системой Windows не ниже 7 версии.

3. ПРОЕКТИРОВАНИЕ

3.1. Функциональный состав

Для описания функционального состава системы можно представить функциональную схему. Она поясняет отдельные виды процессов, протекающих в целостных функциональных блоках.

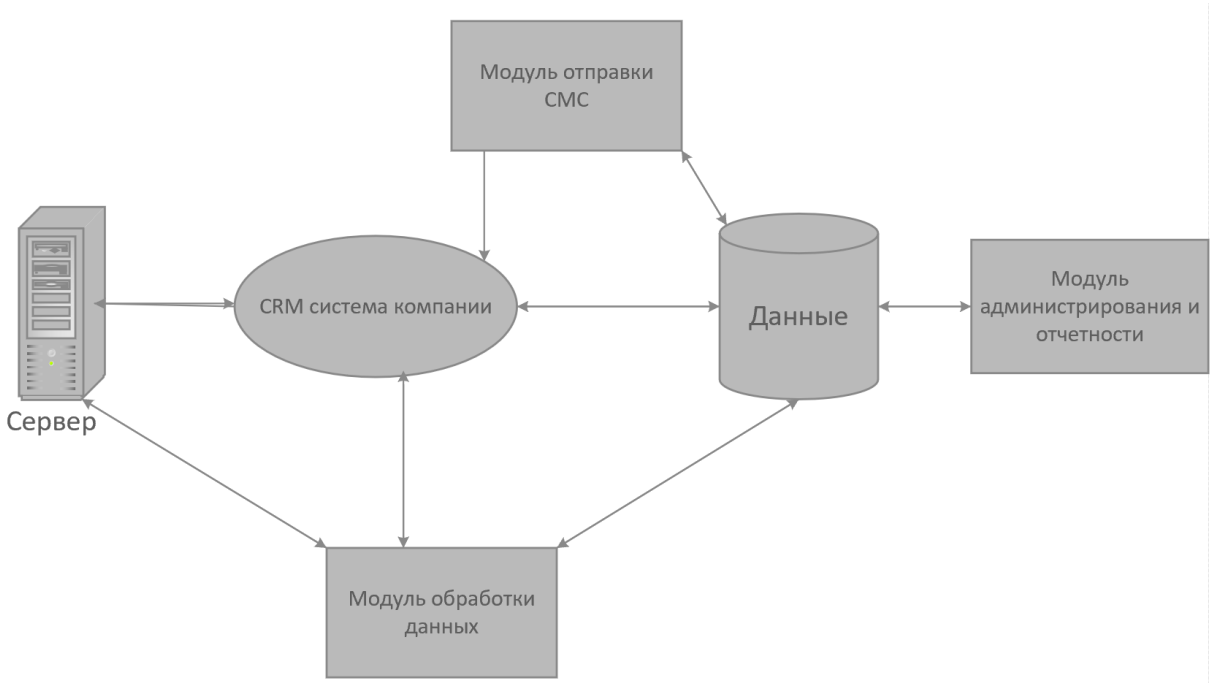

Рисунок 3.1 – Функциональная схема системы

Из рисунка 3.1 видно, что Модуль обработки данных должен взаимодействовать с Сервером компании, с CRM системой, и с Базой данных.

Модуль отправки СМС должен отправлять данные в CRM систему компании, а также взаимодействовать с Базой данных.

Модуль администрирования и отчетности должен взаимодействовать только с Базой данных.

3.2. Описание данных

Необходимо составить схему БД для нашего проекта. Схема базы данных— её структура, описанная на формальном языке, поддерживаемом СУБД.

Инфодент – CRM система стоматологической клиники.

Структурная схема базы данных системы представлена на рисунке 3.2.

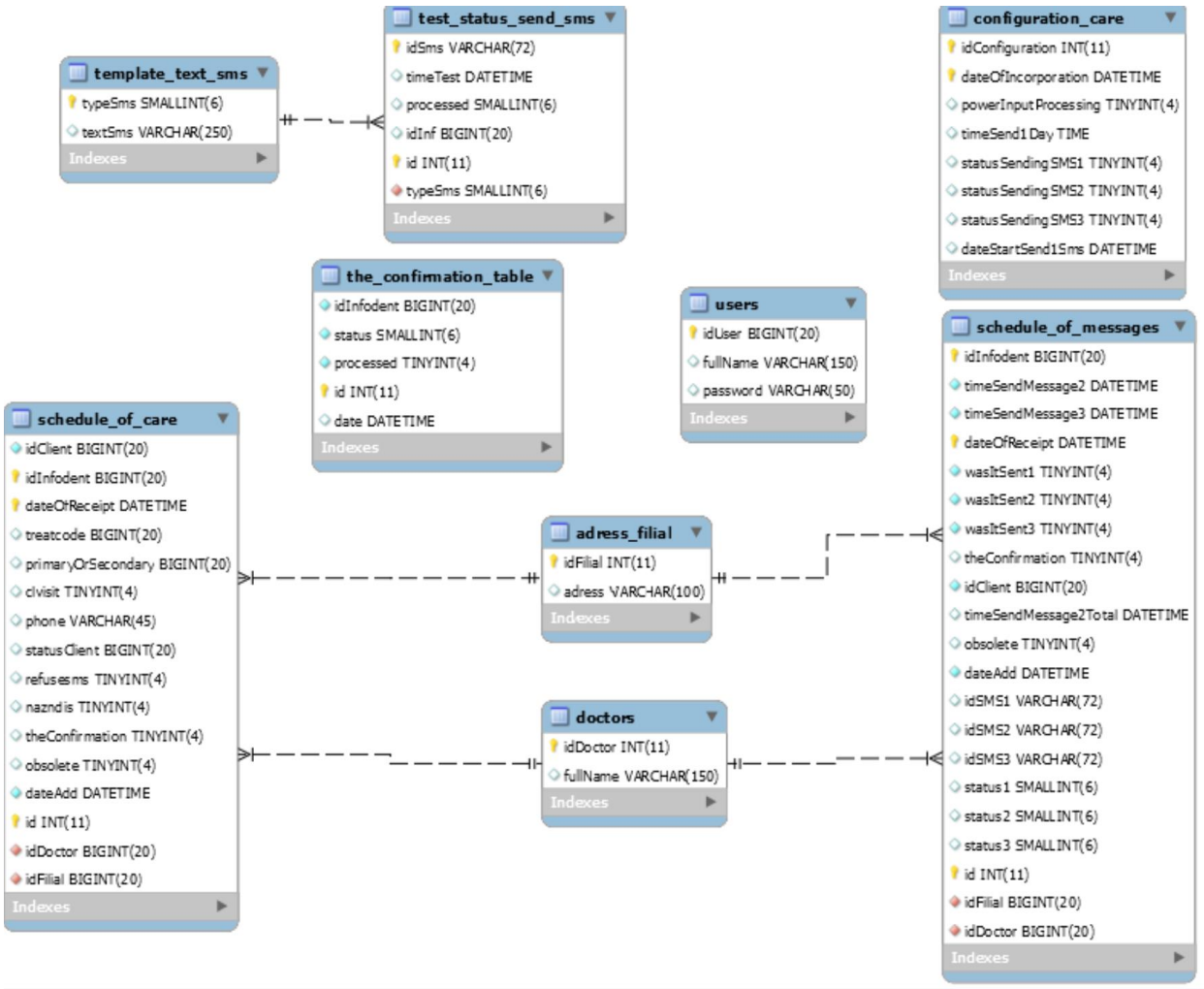

Рисунок 3.2 – Схема базы данных

Список таблиц базы данных приведен в таблице 3.1.

Таблица 3.1– Таблицы базы данных

Наименование	**Назначение**
template_text_sms	Шаблон текста сообщений
test_status_send_sms	Проверка статуса отправки СМС на SMS.ru
configuration_care	Конфигурация для администрирования
the_confirmation_table	Подтверждение приема
Users	Пользователи
schedule_of_messages	Для отправки СМС сообщений и статусов отправленных
schedule_of_care	Информация о приеме
address_filial	Адрес филиала
doctors	ФИО доктора

Таблица «template_text_sms» содержит 2 поля. Она связана с таблицей «test_status_send_sms». Для нее она является подчиненной. Описание полей таблицы приведено в таблице А.1 приложения А.

Таблица «test_status_send_sms» содержит 6 полей. Она связана с таблицей «template_text_sms». Для нее она является главной. Описание полей таблицы приведено в таблице А.2 приложения А.

Таблица «configuration_care» содержит 8 полей. Она не связана с другими таблицами, так как является настройкой для модулей. Описание полей таблицы приведено в таблице А.3 приложения А.

Таблица «the_confirmation_table» содержит 5 полей. Она не связана с другими таблицами, так как она нужна для того, чтобы отслеживать подтверждение/отказ пациентов от приемов. Описание полей таблицы приведено в таблице А.4 приложения А.

Таблица «Users» содержит 3 поля. Она не связана с другими таблицами, так как используется только для модуля администрирования и отчетности. Описание полей таблицы приведено в таблице А.5 приложения А.

Таблица «schedule_of_messages» содержит 21 поле. Она связана с таблицами «address_filial» и «doctors». Для обеих таблиц она является главной. Описание полей таблицы приведено в таблице А.6 приложения А.

Таблица «schedule_of_care» содержит 16 полей. Она связана с таблицами «address_filial» и «doctors». Для обеих таблиц она является главной. Описание полей таблицы приведено в таблице А.7 приложения А.

Таблица «address_filial» содержит 2 поля. Она связана с таблицами «schedule_of_messages» и «schedule_of_care». Для обеих таблиц она является подчиненной. Описание полей таблицы приведено в таблице А.8 приложения А.

Таблица «doctors» содержит 2 поля. Она связана с таблицами «schedule_of_messages» и «schedule_of_care». Для обеих таблиц она является подчиненной. Описание полей таблицы приведено в таблице А.9 приложения А.

Также необходимо создать индексы. Индекс – объект базы данных для повышения производительности поиска данных. Таблица с индексами приведена в приложении А.10.

3.3. Алгоритмы решения задач

Необходимо построить диаграмму потоков данных. Она позволит увидеть все потоки данных, которые должны быть реализованы в системе.

Также для увеличения производительности системы и снятия нагрузки с ЦП (центральный процессор), необходимо выделить потоки данных в отдельные нити и установить время сна нити. Это необходимо, чтобы процессы выполнялись параллельно друг другу и не потребляли много ресурсов, так как некоторые процессы не должны работать постоянно.

Диаграмма потоков данных приведена на рисунке 3.3.

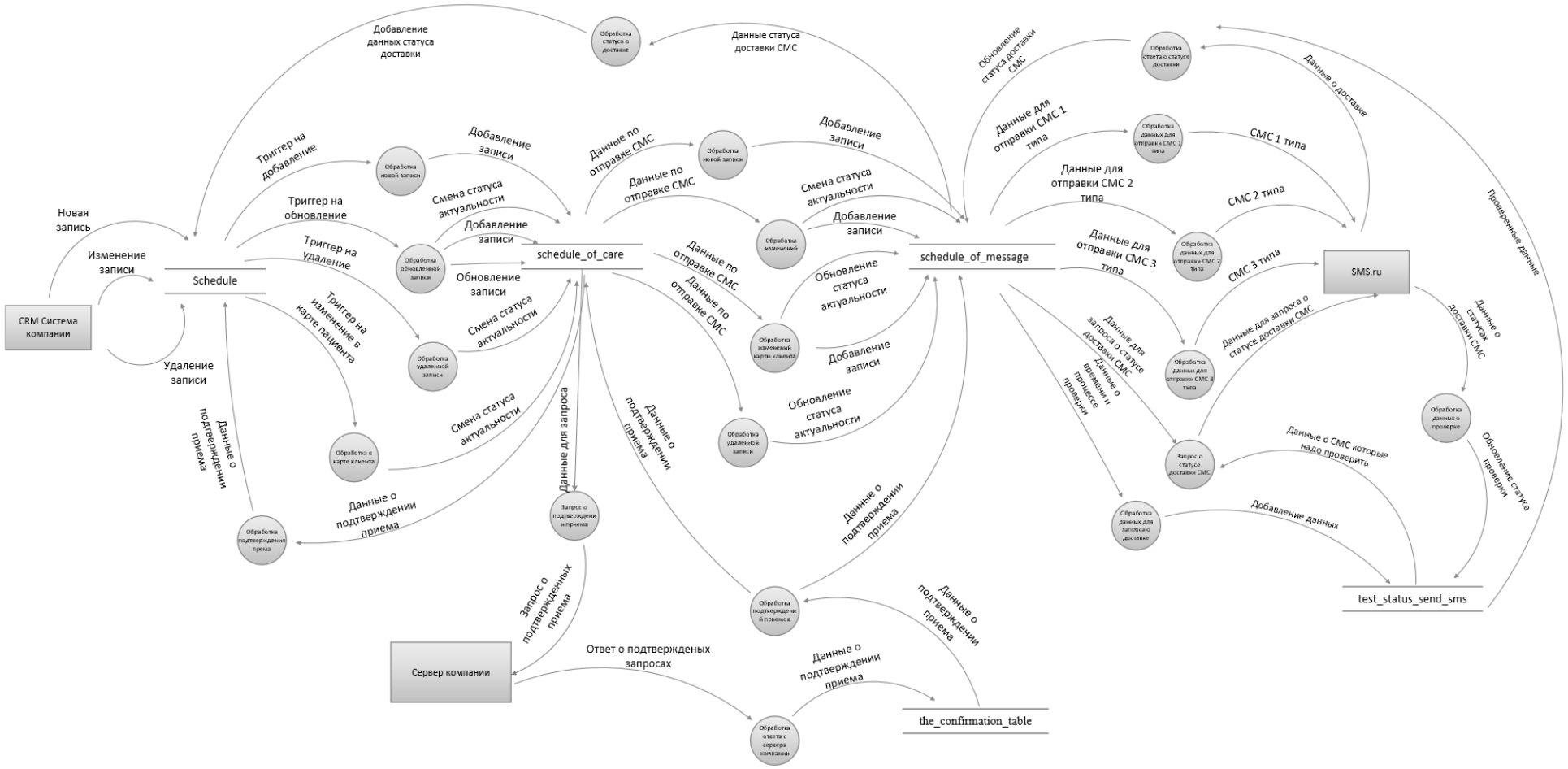

Рисунок 3.3 – Диаграмма потоков данных

Латиницей поименованы используемые таблицы. Из них schedule - таблица расписания из CRM системы, остальные таблицы принадлежат рассматриваемой разработке. Их подробное описание приведено в пункте 3.2.

Из рисунка 3.3 видно, что в системе циркулирует большой поток данных, который может замедлить работу системы и нагружать ЦП.

На рисунке 3.4 приведена часть диаграммы для модуля обработки данных, в которой выделены потоки данных, образующие нити.

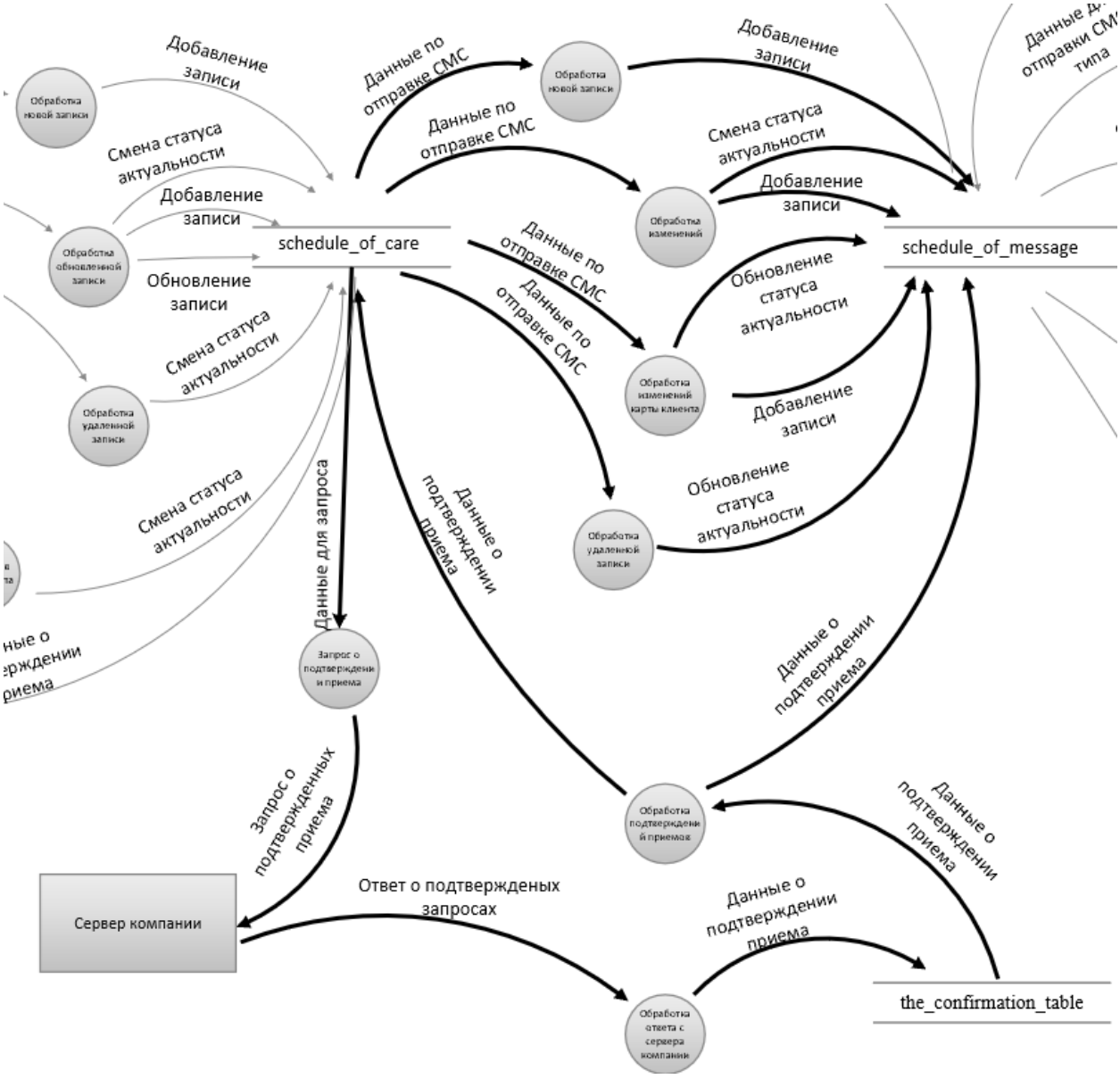

Рисунок 3.4 – Потоки данных модуля обработки данных

Из рисунка 3.4 видно, что можно выделить в отдельные нити следующие потоки данных: добавление нового приема, обновление приема, обновление в карте пациента, удаление приема, подтверждение приема.

Также необходимо рассчитать для каждой нити время сна:

– при отсутствии подключения к базе данных – 1 минута;
– при добавлении нового приема – 6 сек.;
– при обновлении приема – 6 сек.;
– при обновлении в карте пациента – 6 сек.;
– при удалении приема – 2 минуты;
– при подтверждении приема – 10 сек.

Время 6 сек. было выбрано потому, что это время влияет на скорость отправки СМС. Скорость отправки СМС уведомлений пациентам влияет на удовлетворенность пациентов качеством обслуживания.

На рисунке 3.5 приведена часть диаграммы, в которой выделены потоки данных в нити для модуля отправки СМС.

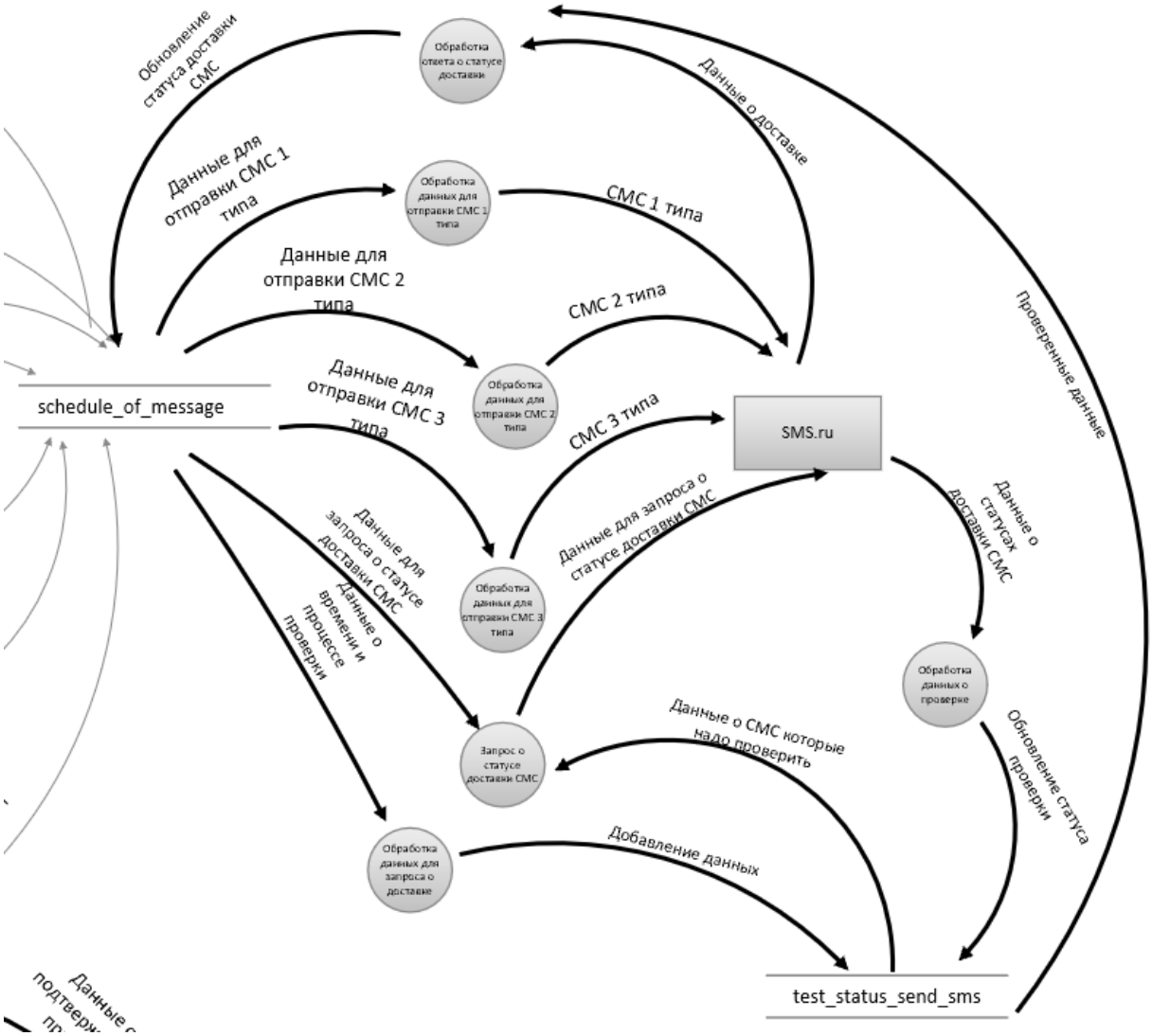

Рисунок 3.5 – Потоки данных модуля отправки СМС

Из рисунка 3.5 видно, что можно выделить потоки данных в отдельные нити: отправка СМС о записи на прием, отправка СМС за день до приема, отправка СМС за 3 часа до приема, проверка статуса отправки СМС, обновление статуса доставки СМС.

Также необходимо рассчитать для каждой нити время сна:

– при отсутствии подключения к базе данных – 1 минута;

– при отправке СМС о записи на прием – 1 минута, если недостаточно средств отправки – 1 минута;

– при отправке СМС за день до приема если недостаточно средств – 1 минута, до следующей отправки рассчитывается по формуле:

((sendDateTime.Hour - time.Hours) * 3600000 + (sendDateTime.Minute - time.Minutes) * 60000 + (sendDateTime.Second - time.Seconds) * 1000 + sendDateTime.Millisecond

,где sendDateTime – текущее системное время, time – время следующей отправки;

– при отправке СМС за 3 часа до приема – 10 минут, если недостаточно средств отправки – 1 минута;

– при проверке статуса отправки – 1 сек.;

– при обновлении статуса доставки СМС – 1 сек.

4. РЕАЛИЗАЦИЯ

4.1. Модуль обработки данных

В таблице 4.1 приведены классы модуля обработки данных.

Таблица 4.1 – Классы модуля обработки данных

Название класса	**Назначение**
Program	Начало работы программы. Запуск нитей.
ConnectionDB	Подключения к базам. Запросы для получения и обработки данных.
TemplateRecordClient	Информация о клиенте для отправки СМС.
TemplateRecordSchedule	Информация о приеме для отправки СМС.
TemplateTestRecord	Информация о старой дате и времени приема или враче, при обновлении в карточке клиента врача и даты и времени приема.
Parse	Парсинг подтверждений приемов.
StatusSmsModel	Статус отправки СМС.

На рисунке 4.1 приведена диаграмма классов модуля обработки данных с указанием основных методов.

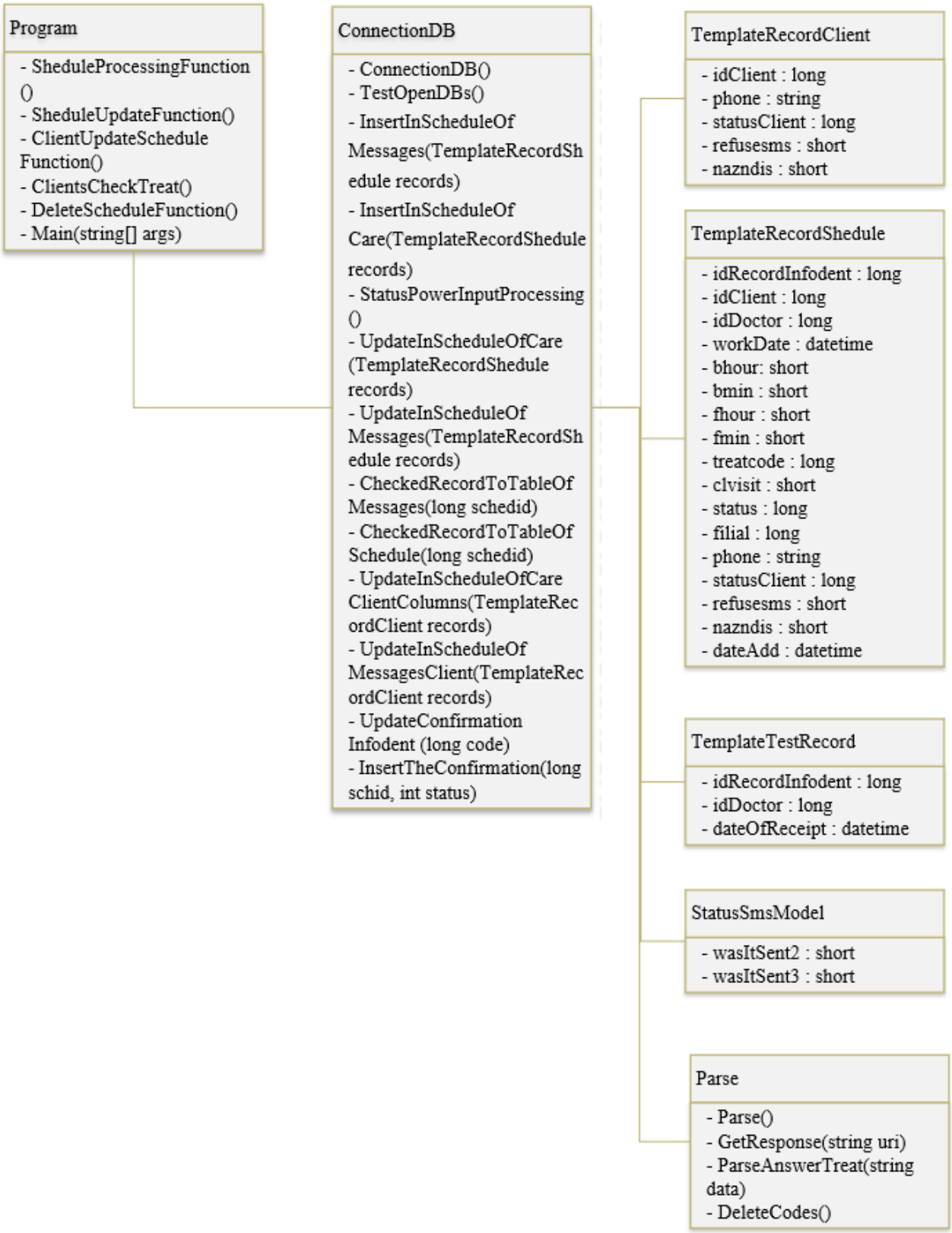

Рисунок 4.1 – Диаграмма классов модуля обработки данных

Класс «Program» приведен в листинге приложения Д. Класс имеет 6 методов. Описания методов приведены в таблице Б.1 приложения Б. Класс запускает нити: обработка новых записей, обработка обновления в записях на прием, обработка обновлений в карте пациента, обработка удаленных приемов, обработка подтверждений приемов. Класс связан с классом «ConnectionDB» с помощью отношения ассоциации.

Класс «ConnectionDB» приведен в листинге приложения Е. Класс имеет 45 методов. Из них один конструктор, 6 подключений к базам данных, остальные получают и обрабатывают данные. Описания методов приведены в таблице Б.2 приложения Б. Класс связан с классами: «TemplateRecordClient», «TemplateRecordSchedule», «TemplateTestRecord», «Parse» и «StatusSmsModel» с помощью отношения ассоциации.

Класс «TemplateRecordClient» имеет 5 полей позволяющих задать информацию о клиенте. Описания полей приведены в таблице Б.3 приложения Б.

Класс «TemplateRecordSchedule» имеет 17 полей позволяющих задать информацию о приеме. Описания полей приведены в таблице Б.4 приложения Б.

Класс «Parse» имеет 4 метода. Из них один конструктор, а остальные обрабатывают подтверждения приемов. Описания методов приведены в таблице Б.5 приложения Б.

Класс «StatusSmsModel» имеет 3 поля, в которых содержится информации о статусе отправки СМС. Описания полей приведены в таблице Б.6 приложения Б.

4.2. Модуль отправки СМС

В таблице 4.2 приведены классы модуля отправки СМС уведомлений.

Таблица 4.2 – Классы модуля отправки СМС уведомлений

Название класса	**Назначение**
Program	Начало работы программы. Запуск нитей.

Окончание таблицы 4.9

ConnectionDB	Подключения к базам. Запросы для получения и обработки данных.
TemplateRecordSchedule	Информация о приеме для отправки СМС.
ReportModel	Информация для большого отчета.
TestSendSmsModel	Проверка отправки СМС.

На рисунке 4.2 приведена диаграмма классов модуля отправки СМС с указанием основных методов.

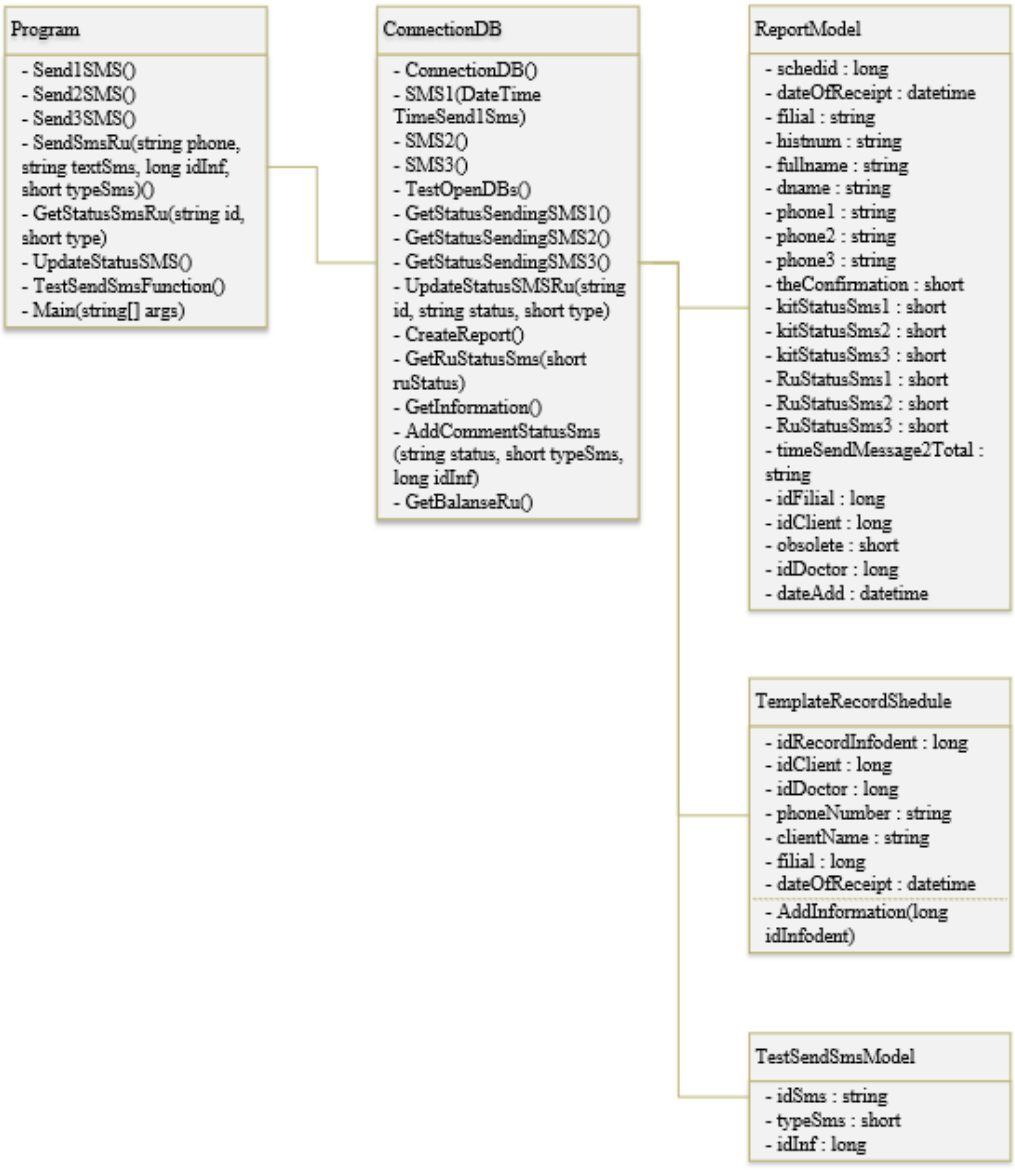

Рисунок 4.2– Диаграмма классов модуля обработки данных

Класс «Program» приведен в листинге приложения Ж. Класс имеет 8 методов. Описания методов приведены в таблице В.1 приложения В. Класс запускает нити: отправка СМС о записи на прием, отправка СМС за день до приема, отправка СМС за 3 часа до приема, проверка статуса отправки СМС, обновление статуса доставки СМС. Класс связан с классом «ConnectionDB» с помощью отношения ассоциации.

Класс «ConnectionDB» приведен в листинге приложения З. Класс имеет 50 методов. Из них один конструктор, 6 подключений к базам данных, остальные получают, обрабатывают данные и отправляют СМС. Описания методов приведены в таблице В.2 приложения В. Класс связан с классами: «ReportModel», «TemplateRecordSchedule» и «TestSendSmsModel» с помощью отношения ассоциации.

Класс «TemplateRecordSchedule» имеет 7 полей, позволяющих задать информацию о приеме, и один метод, добавляющий информацию. Описания полей и метода приведены в таблице В.3 приложения В.

Класс «ReportModel» имеет 23 поля, в которых содержится информация для построения «Большого отчета». Описания полей приведены в таблице В.4 приложения Б.

Класс «TestSendSmsModel» имеет 3 поля, в которых содержится информация для проверки отправки СМСы. Описания полей приведены в таблице В.5 приложения В.

4.3. Модуль администрирования и отчетности

В таблице 4.3 приведены классы модуля администрирования и отчетности.

Таблица 4.3 – Классы модуля администрирования и отчетности

Название класса	**Назначение**
Program	Начало работы программы. Запуск нитей.

Окончание таблицы 4.3

ConnectionDB	Подключения к базам. Запросы для получения и обработки данных.
ReportModel	Информация для отчетов.
Autorization	Форма авторизации.
GlobalForm	Главная форма.
SendSms	Форма отправки СМС.
SendSmsEmail	Отправка СМС и Email.
SettingCase	Форма настройки отправки СМС уведомлений и обработки новых приемов.
SettingReport	Форма отчетов.
SettingTemplate	Форма управления шаблонами СМС.

На рисунке 4.3 приведена диаграмма классов модуля администрирования и отчетности с указанием основных методов.

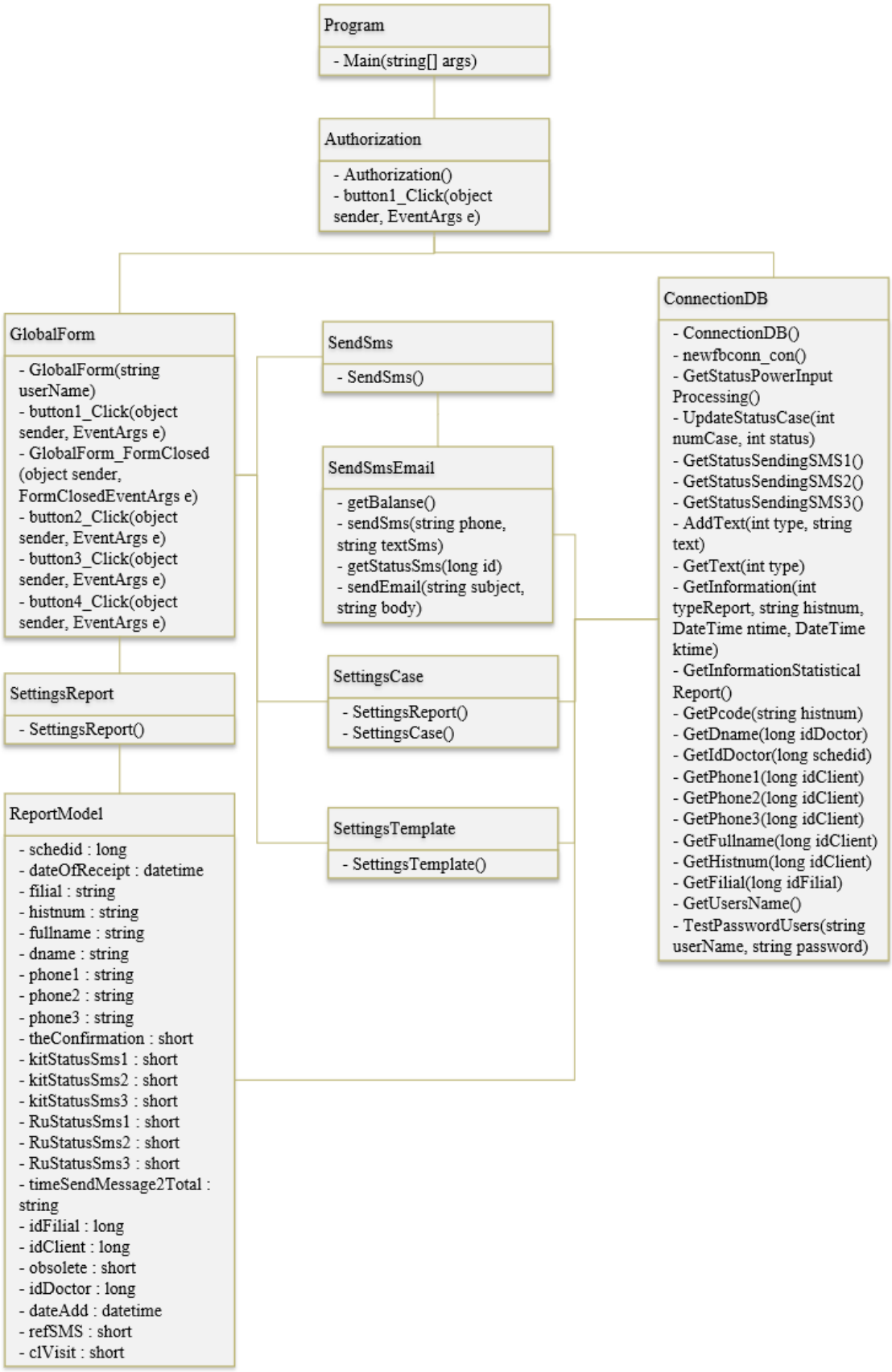

Рисунок 4.3 – Диаграмма классов модуля администрирования и отчетности

Класс «Program» приведен в листинге приложения И. Класс имеет 1 метод. Описания метода приведено в таблице Г.1 приложения Г. Класс запускает Авторизацию. Класс связан с классом «Autorization» с помощью отношения ассоциации.

Класс «ConnectionDB» приведен в листинге приложения К. Класс имеет 22 метода. Из них один конструктор, 1 подключение к базе данных, остальные получают данные. Описания методов приведены в таблице Г.2 приложения Г. Класс связан с классами: «SendSmsEmail», «SettingCase», «SettingTemplate» и «ReportModel» с помощью отношения ассоциации.

Класс «ReportModel» имеет 24 поля, в которых содержится информации для построения отчетов. Описания полей приведены в таблице Г.3 приложения Г.

Класс «Autorization» имеет 2 метода. Описания методов приведены в таблице Г.4 приложения Г. Класс связан с класами «GlobalForm» и «ConnectionDB» с помощью отношения ассоциации.

Класс «GlobalForm» имеет 6 методов. Описания методов приведены в таблице Г.5 приложения Г. Класс связан с класами «SendSms», «SettingCase», «SettingTemplate» и «SettingReport» с помощью отношения ассоциации.

Класс «SendSms» имеет 1 метод для отправки СМС. Описание метода приведено в таблице Г.6 приложения Г. Класс связан с классом «SendSmsEmail» с помощью отношения ассоциации.

Класс «SendSmsEmail» имеет 4 метода. Описание методов приведено в таблице Г.7 приложения Г.

Класс «SettingCase» имеет 2 метода. Описание методов приведено в таблице Г.8 приложения Г.

Класс «SettingReport» имеет 1 метод. Описание метода приведено в таблице Г.9 приложения Г.

Класс «SettingTemplate» имеет 1 метод. Описание метода приведено в таблице Г.10 приложения Г.

5. ТЕСТИРОВАНИЕ

5.1. Методология тестирования

Система прошла отладку и уже введена в эксплуатацию.

Программа прошла альфа тестирование, которое проводилось на тестовом пациенте. Ему создавался прием в расписании в Инфоденте. После создания он обрабатывался модулем обработки данных и появился в таблицах «schedule_of_care» и «schedule_of_messages». Далее проверялась работа модуля отправки СМС, была ли отправлена СМС о записи на прием. Также необходимо было проверить что после изменения в расписании даты приема или врача запись делалась не актуальной и добавлялись новые записи в таблицах «schedule_of_care» и «schedule_of_messages» с актуальными данными, после чего должна была уйти новая СМС о записи на прием. Далее проверялась отправка СМС за день до приема и обработка подтверждений приема, добавление подтверждения или отказа в расписание и статусы отправки СМС. Отправка СМС за 3 часа до приема. Также была проверена работа модуля администрирования и отчетности.

После того, как программа прошла альфа тестирование, она была передана заказчику. Далее тестирование проводилось на реальных данных. Чтобы проверить работоспособность и «автоматизированность» системы, работа программы контролировалась и отслеживалась в течении месяца во избежание не предусмотренных раннее ситуаций и сбоев. После программа была полностью введена в эксплуатацию.

5.2. Проведение процедуры тестирования

В данном пункте приведен пример тестирования на пациенте Test. Прием был добавлен на 30.06 8:00 к врачу Сайгофаровой в филиал «Кит Имплант».

На рисунке 5.1 приведена таблица «schedule_of_care» в которой видно, что прием был обработан модулем обработки данных.

IDCLIENT	IDINFODENT	DATEOFRECEIPT	TREATCODE	PRIMARYORSECONDARY	CLVISIT	PHONE	STATUSCLIENT
50 013 263	50 246 044	2019-06-30 08:00:00	509 108	13	0	79642418229	0

REFUSSMS	NAZNDIS	THECONFIRMATION	OBSOLETE	DATEADD	ID	IDDOCTOR	IDFILIAL
0	0	0	0	2019-06-08 12:34:07	61 286	990 000 826	3

Рисунок 5.1 – Таблица «schedule_of_care»

На рисунке 5.2 приведена таблица «schedule_of_messages», в которой видно, что прием был обработан модулем обработки данных и готов к отправке СМС о записи на прием.

IDINFODENT	TIMESENDMESSAGE2	TIMESENDMESSAGE3	DATEOFRECEIPT	WASITSENT1	WASITSENT2	WASITSENT3
50 246 044	2019-06-29 08:00:00	2019-06-30 08:00:00	2019-06-30 08:00:00	0	0	0

THECONFIRMATION	IDCLIENT	TIMESENDMESSAGE2...	OBSOLETE	DATEADD	IDSMS1	IDSMS2
0	50 013 263	<null>	0	2019-06-08 12:34:07	<null>	<null>

IDSMS3	STATUS1	STATUS2	STATUS3	ID	IDFILIAL	IDDOCTOR
<null>	<null>	<null>	<null>	61 286	3	990 000 826

Рисунок 5.2 – Таблица «schedule_of_messages»

После добавления в таблицу «schedule_of_messages», модуль отправки СМС должен отправить СМС о записи на прием.

На рисунке 5.3 изображена экранная форма СМС о записи на прием.

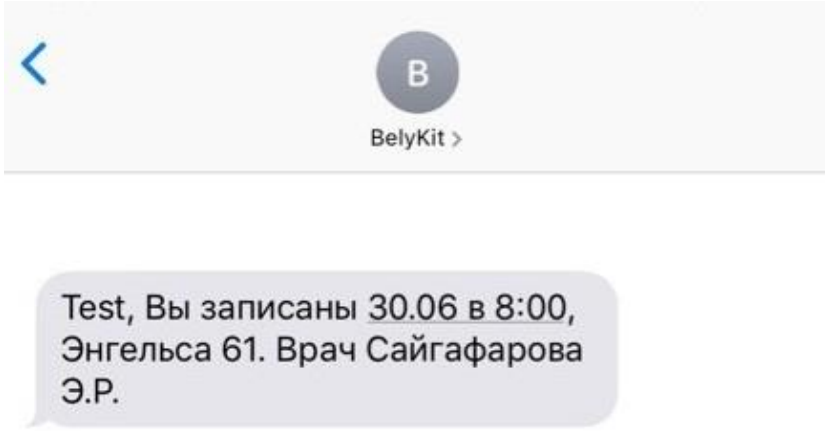

Рисунок 5.3 – Экранная форма СМС о записи на прием

После отправки СМС необходимо проверить что в таблице «test_status_send_sms» появилась запись для проверки статуса доставки СМС.

На рисунке 5.4 приведена таблица «test_status_send_sms».

IDSMS	TIMETEST	PROCESSED	IDINF	ID	TYPESMS
201922-1001718	2019-06-08 12:50:13	1	50 246 044	61 286	1

Рисунок 5.4 – Таблица «test_status_send_sms»

Также необходимо проверить, что в таблице «schedule_of_messages» в поле «wasItSent1» стоит 1, в «idSMS1» написан идентификатор отправленной СМС и «status1» обновился.

На рисунке 5.5 приведена таблица «schedule_of_messages».

IDINFODENT	TIMESENDMESSAGE2	TIMESENDMESSAGE3	DATEOFRECEIPT	WASITSENT1	WASITSENT2	WASITSENT3
50 246 044	2019-06-29 08:00:00	2019-06-30 08:00:00	2019-06-30 08:00:00	1	0	0

THECONFIRMATION	IDCLIENT	TIMESENDMESSAGE2...	OBSOLETE	DATEADD	IDSMS1	IDSMS2
0	50 013 263	<null>	0	2019-06-08 12:34:07	201922-1001718	<null>

IDSMS3	STAT...	STATUS2	STATUS3	ID	IDFILIAL	IDDOCTOR
<null>	4	<null>	<null>	61 286	3	990 000 826

Рисунок 5.5 - Таблица «schedule_of_messages»

Далее необходимо проверить обработку обновлений в карете пациента. У Test была изменена дата и время приема с 30.06 8:00 на 09.06 12:00.

На рисунке 5.6 видно, что модуль обработки данных сделал старую запись не актуальной в таблице «schedule_of_care» присвоив «obsolete» равной 1. И добавил новую запись в таблицу «schedule_of_care» с актуальными данными.

IDCLIENT	IDINFODENT	DATEOFRECEIPT	TREATCODE	PRIMARYORSECONDARY	CLVISIT	PHONE	STATUSCLIENT
50 013 263	50 246 044	2019-06-30 08:00:00	509 108	13	0	79642418229	0
50 013 263	50 246 044	2019-06-09 12:00:00	509 108	13	0	79642418229	0

REFUSSMS	NAZNDIS	THECONFIRMATION	OBSOLETE	DATEADD	ID	IDDOCTOR	IDFILIAL
0	0	0	1	2019-06-08 12:34:07	61 286	990 000 826	3
0	0	0	0	2019-06-08 13:05:18	61 289	990 000 826	3

Рисунок 5.6 - Таблица «schedule_of_care»

На рисунке 5.7 видно, что модуль обработки данных сделал старую запись не актуальной в таблице «schedule_of_messages» присвоив «obsolete» равной 1. И добавил новую запись в таблицу «schedule_of_messages» с актуальными данными.

IDINFODENT	TIMESENDMESSAGE2	TIMESENDMESSAGE3	DATEOFRECEIPT	WASITSENT1	WASITSENT2	WASITSENT3
50 246 044	2019-06-29 08:00:00	2019-06-30 08:00:00	2019-06-30 08:00:00	1	0	0
50 246 044	2019-06-08 12:00:00	2019-06-09 09:00:00	2019-06-09 12:00:00	0	0	0

THECONFIRMATION	IDCLIENT	TIMESENDMESSAGE2...	OBSOLETE	DATEADD	IDSMS1	IDSMS2
0	50 013 263	<null>	1	2019-06-08 12:34:07	201922-1001718	<null>
0	50 013 263	<null>	0	2019-06-08 13:05:18	<null>	<null>

IDSMS3	STAT...	STATUS2	STATUS3	ID	IDFILIAL	IDDOCTOR
<null>	4	<null>	<null>	61 286	3	990 000 826
<null>	<null>	<null>	<null>	61 289	3	990 000 826

Рисунок 5.7 – Таблица «schedule_of_messages»

На рисунке 5.8 изображена экранная форма, показывающая, что СМС о записи на прием была отправлена пациенту.

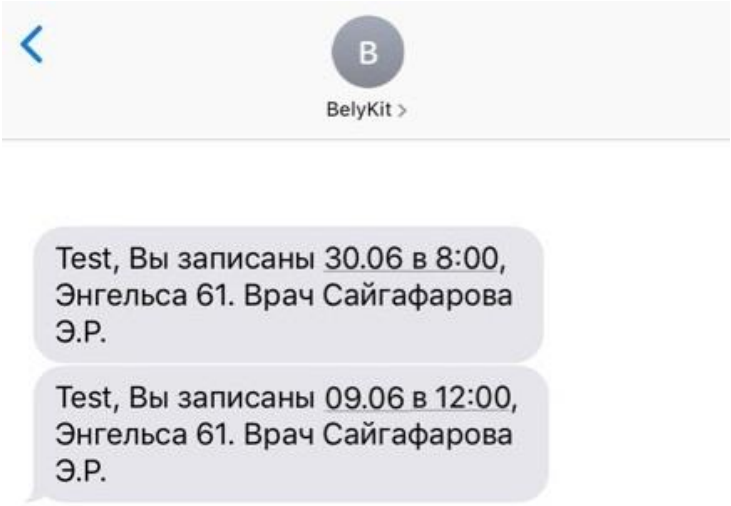

Рисунок 5.8 – Экранная форма СМС о записи на прием

После отправки СМС необходимо проверить, что в таблице «test_status_send_sms» появилась запись для проверки статуса доставки СМС.

На рисунке 5.9 приведена таблица «test_status_send_sms».

IDSMS	TIMETEST	PROCESSED	IDINF	ID	TYPESMS
201922-1001718	2019-06-08 12:50:13	1	50 246 044	61 286	1
201922-1001723	2019-06-08 13:21:08	1	50 246 044	61 289	1

Рисунок 5.9 – Таблица «test_status_send_sms»

Также необходимо проверить, что в таблице «schedule_of_messages» в поле «wasItSent1» стоит 1, в «idSMS1» написан идентификатор отправленной СМС и «status1» обновился.

На рисунке 5.10 приведена таблица «schedule_of_messages».

IDINFODENT	TIMESENDMESSAGE2	TIMESENDMESSAGE3	DATEOFRECEIPT	WASITSENT1	WASITSENT2	WASITSENT3
50 246 044	2019-06-29 08:00:00	2019-06-30 08:00:00	2019-06-30 08:00:00	1	0	0
50 246 044	2019-06-08 12:00:00	2019-06-09 09:00:00	2019-06-09 12:00:00	1	0	0

THECONFIRMATION	IDCLIENT	TIMESENDMESSAGE2...	OBSOLETE	DATEADD	IDSMS1	IDSMS2
0	50 013 263	<null>	1	2019-06-08 12:34:07	201922-1001718	<null>
0	50 013 263	<null>	0	2019-06-08 13:05:18	201922-100123	<null>

IDSMS3	STATUS1	STATUS2	STATUS3	ID	IDFILIAL	IDDOCTOR
<null>	4	<null>	<null>	61 286	3	990 000 826
<null>	4	<null>	<null>	61 289	3	990 000 826

Рисунок 5.10 - Таблица «schedule_of_messages»

Проверим работу отправки СМС за день до приема.

На рисунке 5.11 изображена экранная форма СМС за день до приема.

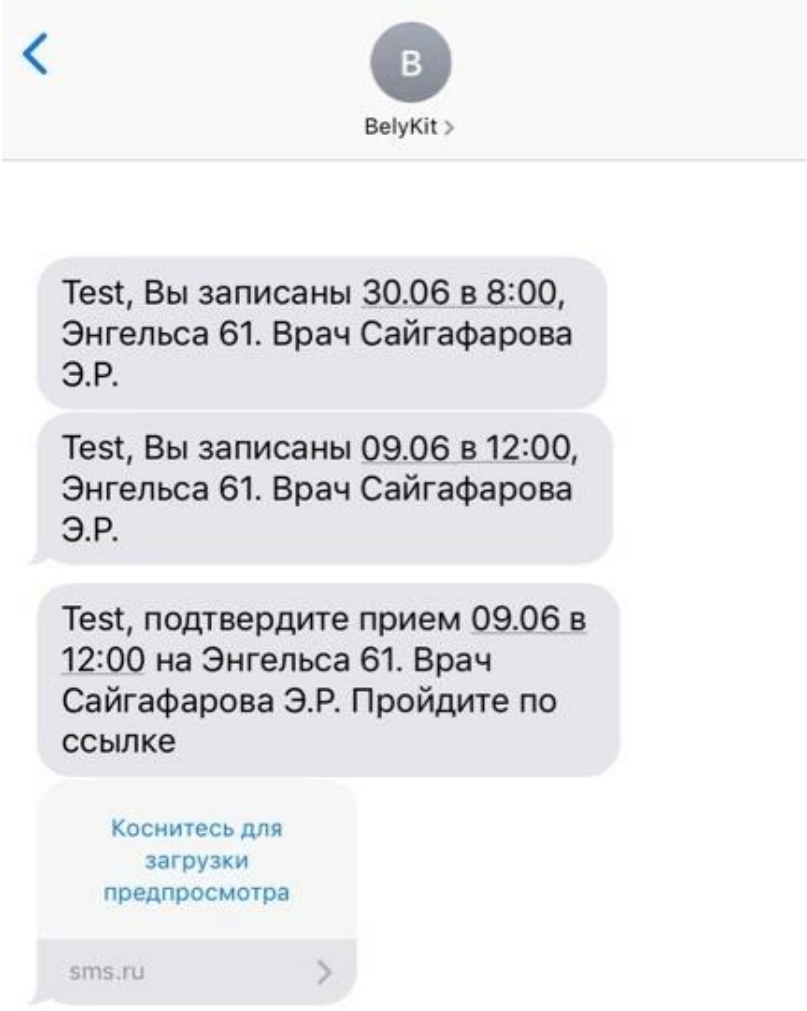

Рисунок 5.11 – Экранная форма СМС за день до приема

После отправки СМС необходимо проверить, что в таблице «test_status_send_sms» появилась запись для проверки статуса доставки СМС.

На рисунке 5.12 приведена таблица «test_status_send_sms».

IDSMS	TIMETEST	PROCESSED	IDINF	ID	TYPESMS
201922-1001718	2019-06-08 12:50:13	1	50 246 044	61 286	1
201922-1001723	2019-06-08 13:21:08	1	50 246 044	61 289	1
201923-1001125	2019-06-08 13:45:18	1	50 246 044	61 289	2

Рисунок 5.12 – Таблица «test_status_send_sms»

Также необходимо проверить, что в таблице «schedule_of_messages» в поле «wasItSent2» стоит 1, в «idSMS2» написан идентификатор отправленной СМС и «status2» обновился.

На рисунке 5.13 приведена таблица «schedule_of_messages».

IDINFODENT	TIMESENDMESSAGE2	TIMESENDMESSAGE3	DATEOFRECEIPT	WASITSENT1	WASITSENT2	WASITSENT3
50 246 044	2019-06-29 08:00:00	2019-06-30 08:00:00	2019-06-30 08:00:00	1	0	0
50 246 044	2019-06-08 12:00:00	2019-06-09 09:00:00	2019-06-09 12:00:00	1	1	0

THECONFIRMATION	IDCLIENT	TIMESENDMESSAGE2...	OBSOLETE	DATEADD	IDSMS1	IDSMS2
0	50 013 263	<null>	1	2019-06-08 12:34:07	201922-1001718	<null>
0	50 013 263	2019-06-08 12:30:18	0	2019-06-08 13:05:18	201922-100123	201923-1001125

IDSMS3	STATUS1	STATUS2	STATUS3	ID	IDFILIAL	IDDOCTOR
<null>	4	<null>	<null>	61 286	3	990 000 826
<null>	4	4	<null>	61 289	3	990 000 826

Рисунок 5.13 - Таблица «schedule_of_messages»

Перейдем по ссылке в СМС для подтверждения приема.

На рисунке 5.14 приведена экранная форма ссылки подтверждения приема.

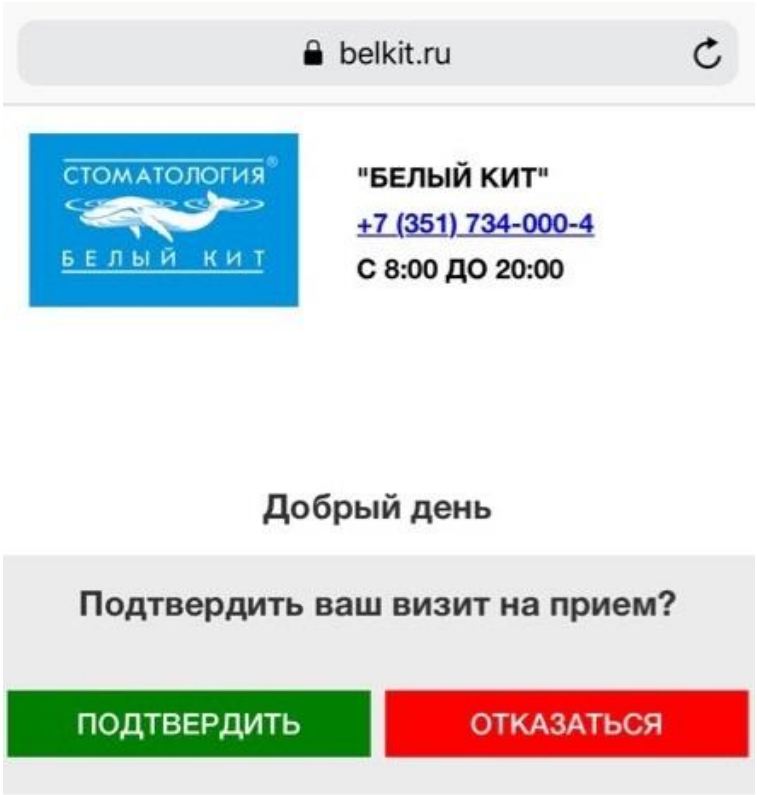

Рисунок 5.14 – Экранная форма подтверждения приема

Выбираем подтвердить.

На рисунке 5.15 изображена экранная форма подтверждения приема.

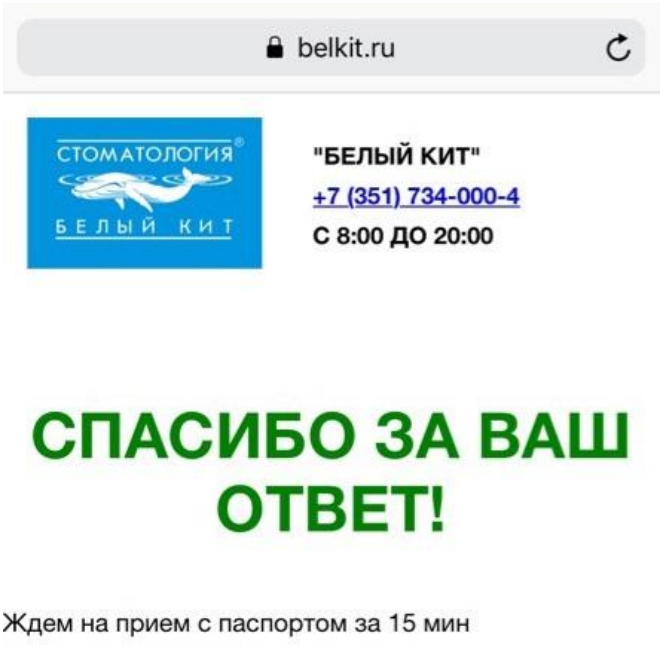

Рисунок 5.15 – Экранная форма подтверждения приема

На рисунке 5.16 изображена экранная форма отказа от приема.

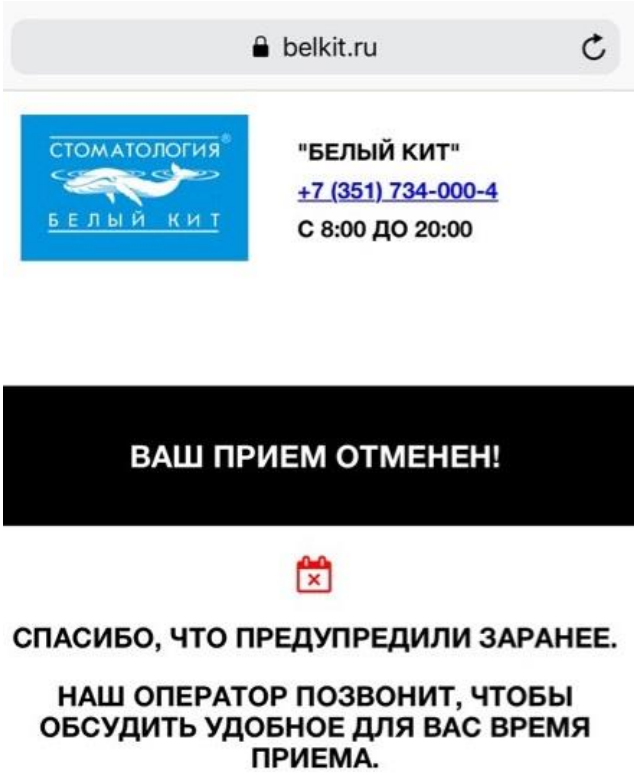

Рисунок 5.16 – Экранная форма отказа от приема

Далее будет проверяться, что произойдет после подтверждения. С отказом будет все аналогично, отличие только в том, что в таблице «schedule_of_messages» поле «obsolete» становится равным 1.

Проверяем, что модуль отправки СМС обработал подтверждение. Для этого необходимо проверить, что в таблицу «the_confirmation_table» добавилась запись, в таблице «schedule_of_messages» и «schedule_of_care» обновился статус подтверждения.

На рисунке 5.17 изображена таблица «the_confirmation_table».

IDINFODENT	STATUS	PROCESSED	ID	DATE
50 246 044	1	1	61 289	2019-06-08 13:32:19

Рисунок 5.17 – Таблица «the_confirmation_table»

На рисунке 5.18 изображена таблица «schedule_of_messages».

IDINFODENT	TIMESENDMESSAGE2	TIMESENDMESSAGE3	DATEOFRECEIPT	WASITSENT1	WASITSENT2	WASITSENT3
50 246 044	2019-06-29 08:00:00	2019-06-30 08:00:00	2019-06-30 08:00:00	1	0	0
50 246 044	2019-06-08 12:00:00	2019-06-09 09:00:00	2019-06-09 12:00:00	1	1	0

THECONFIRMATION	IDCLIENT	TIMESENDMESSAGE2...	OBSOLETE	DATEADD	IDSMS1	IDSMS2
0	50 013 263	<null>	1	2019-06-08 12:34:07	201922-1001718	<null>
1	50 013 263	2019-06-08 12:30:18	0	2019-06-08 13:05:18	201922-100123	201923-1001125

IDSMS3	STATUS1	STATUS2	STATUS3	ID	IDFILIAL	IDDOCTOR
<null>	4	<null>	<null>	61 286	3	990 000 826
<null>	4	4	<null>	61 289	3	990 000 826

Рисунок 5.18 – Таблица «schedule_of_messages»

На рисунке 5.19 изображена таблица «schedule_of_care».

IDCLIENT	IDINFODENT	DATEOFRECEIPT	TREATCODE	PRIMARYORSECONDARY	CLVISIT	PHONE	STATUSCLIENT
50 013 263	50 246 044	2019-06-30 08:00:00	509 108	13	0	79642418229	0
50 013 263	50 246 044	2019-06-09 12:00:00	509 108	13	0	79642418229	0

REFUSSMS	NAZNDIS	THECONFIRMATION	OBSOLETE	DATEADD	ID	IDDOCTOR	IDFILIAL
0	0	0	1	2019-06-08 12:34:07	61 286	990 000 826	3
0	0	1	0	2019-06-08 13:05:18	61 289	990 000 826	3

Рисунок 5.19 – Таблица «schedule_of_care»

Далее необходимо проверить, что отправляются СМС за 3 часа до приема.

На рисунке 5.20 изображена экранная форма СМС за 3 часа до приема.

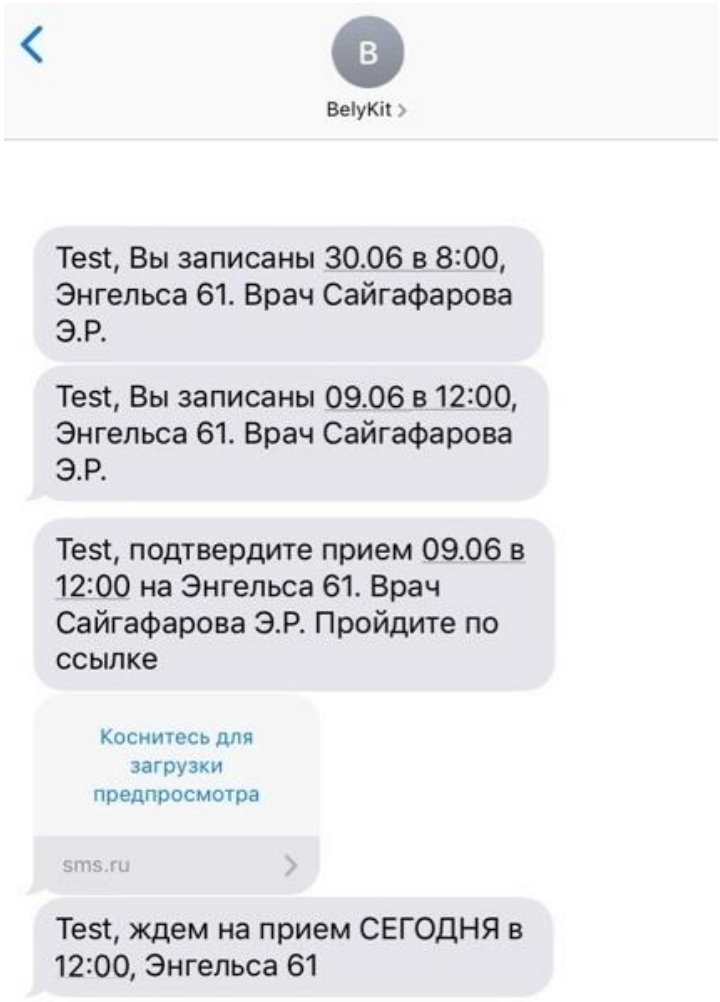

Рисунок 5.20 – Экранная форма отправки СМС за 3 часа до приема

После отправки СМС необходимо проверить, что в таблице «test_status_send_sms» появилась запись для проверки статуса доставки СМС.

На рисунке 5.21 приведена таблица «test_status_send_sms».

IDSMS	TIMETEST	PROCESSED	IDINF	ID	TYPESMS
201922-1001718	2019-06-08 12:50:13	1	50 246 044	61 286	1
201922-1001723	2019-06-08 13:21:08	1	50 246 044	61 289	1
201923-1001125	2019-06-08 13:45:18	1	50 246 044	61 289	2
201924-1001563	2019-06-09 09:00:13	1	50 246 044	61 289	3

Рисунок 5.21 – Таблица «test_status_send_sms»

Также необходимо проверить, что в таблице «schedule_of_messages» в поле «wasItSent3» стоит 1, в «idSMS3» написан идентификатор отправленной СМС и «status3» обновился.

На рисунке 5.22 приведена таблица «schedule_of_messages».

IDINFODENT	TIMESENDMESSAGE2	TIMESENDMESSAGE3	DATEOFRECEIPT	WASITSENT1	WASITSENT2	WASITSENT3
50 246 044	2019-06-29 08:00:00	2019-06-30 08:00:00	2019-06-30 08:00:00	1	0	0
50 246 044	2019-06-08 12:00:00	2019-06-09 09:00:00	2019-06-09 12:00:00	1	1	1

THECONFIRMATION	IDCLIENT	TIMESENDMESSAGE2...	OBSOLETE	DATEADD	IDSMS1	IDSMS2
0	50 013 263	<null>	1	2019-06-08 12:34:07	201922-1001718	<null>
1	50 013 263	2019-06-08 12:30:18	0	2019-06-08 13:05:18	201922-100123	201923-1001125

IDSMS3	STATUS1	STATUS2	STATUS3	ID	IDFILIAL	IDDOCTOR
<null>	4	<null>	<null>	61 286	3	990 000 826
201924-1001563	4	4	4	61 289	3	990 000 826

Рисунок 5.22 - Таблица «schedule_of_messages»

На рисунках 5.23 – 5.28 изображены экранные формы модуля администрирования и отчетности.

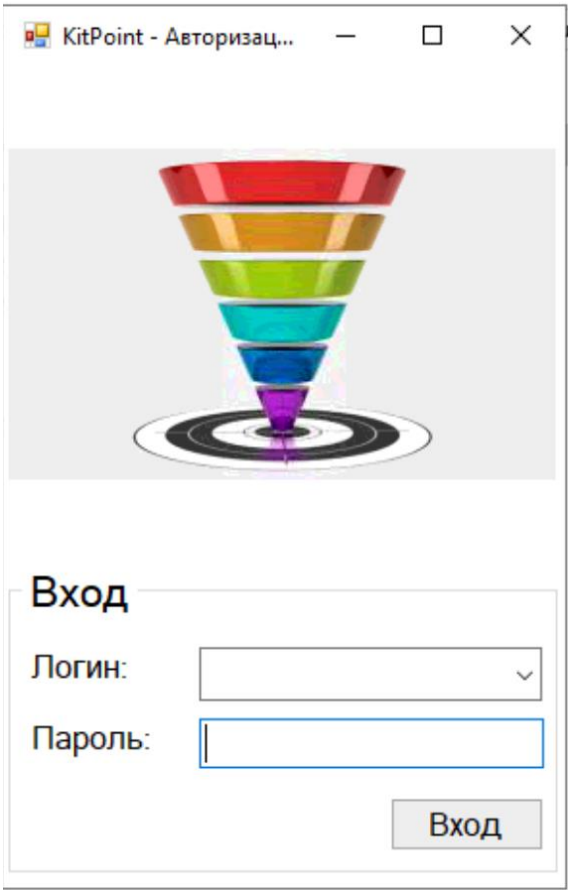

Рисунок 5.23 – Экранная форма авторизации

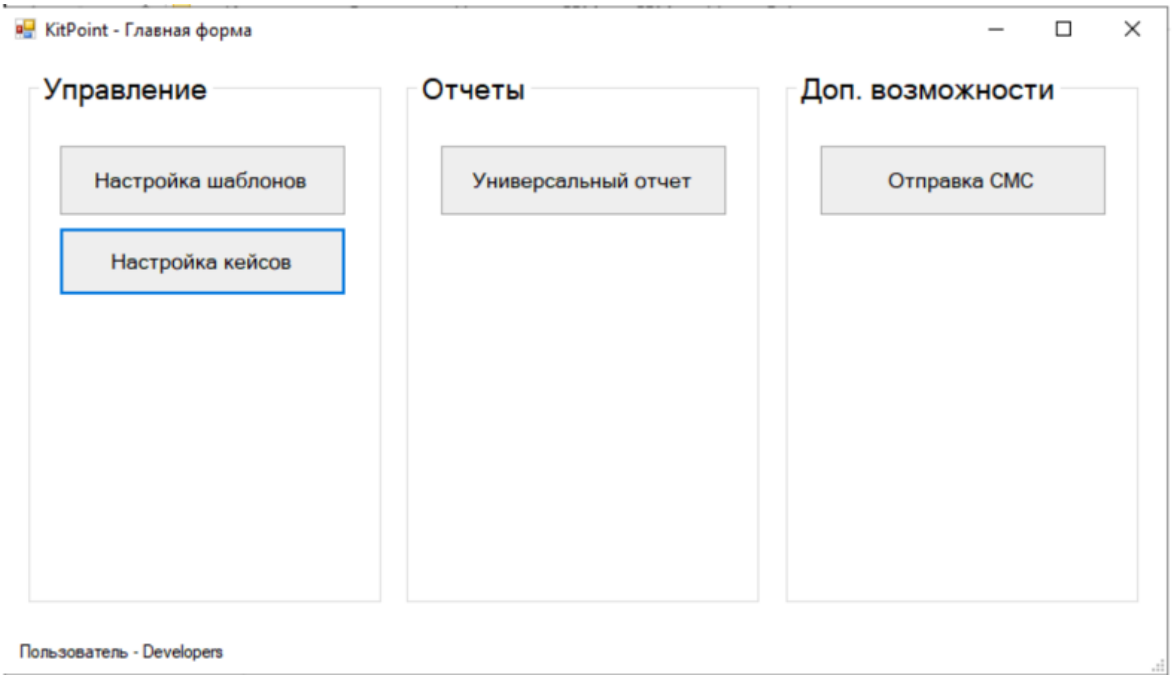

Рисунок 5.24 – Экранная форма главной формы

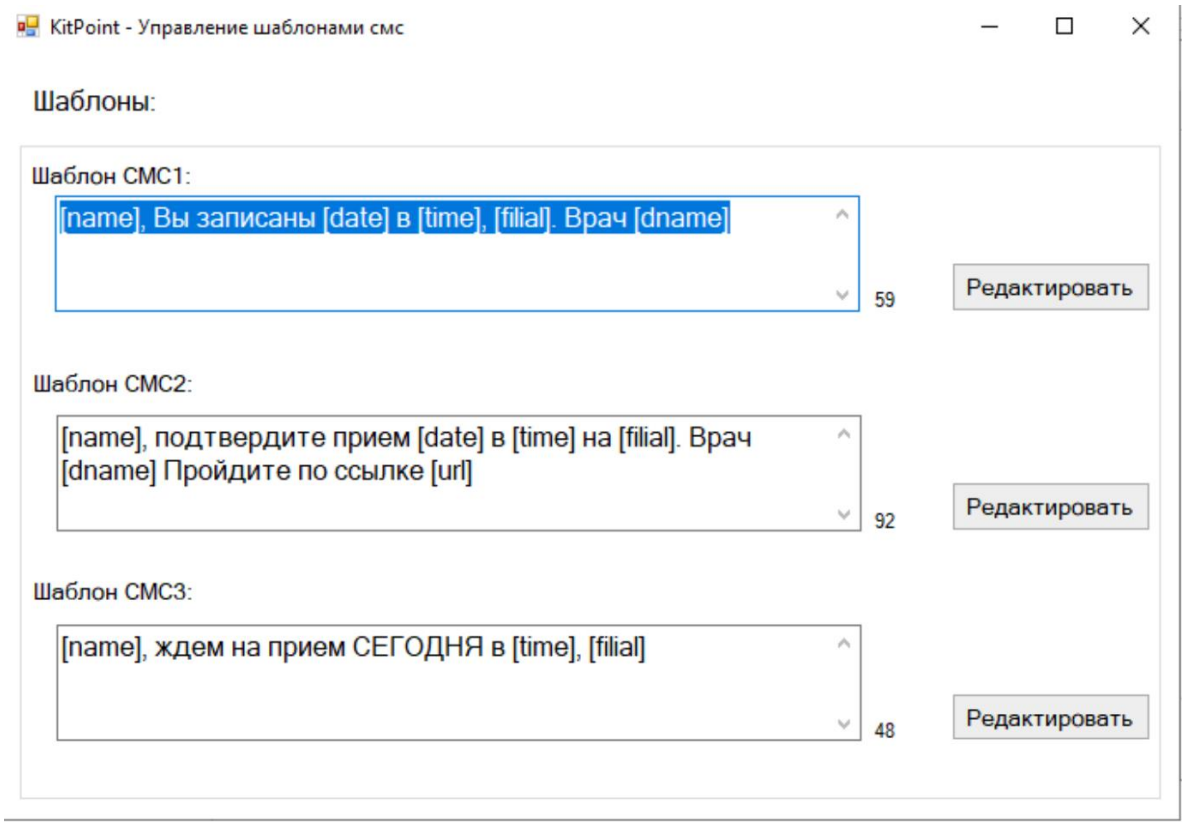

Рисунок 5.25 – Экранная форма управления шаблонами СМС

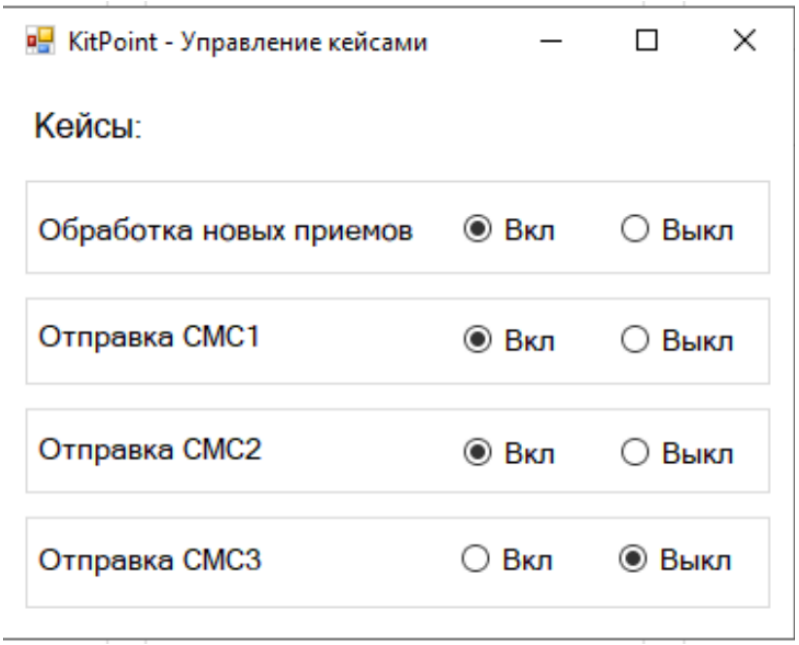

Рисунок 5.26 – Экранная форма управления кейсами

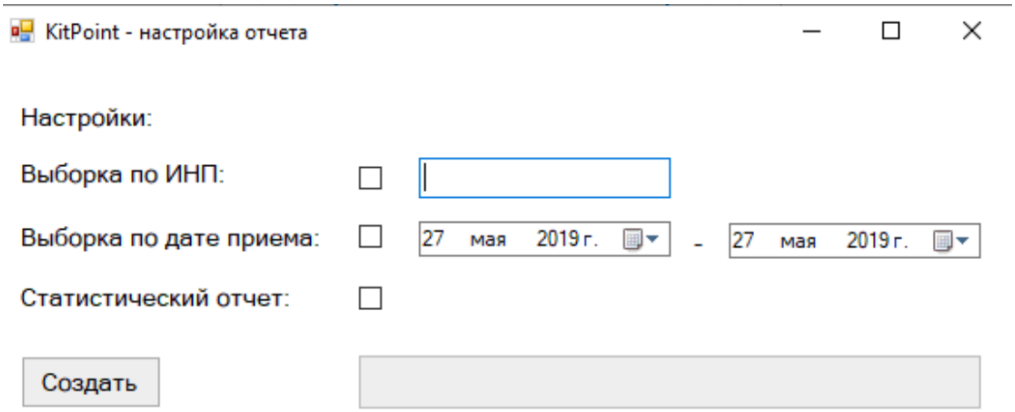

Рисунок 5.27 – Экранная форма настройки отчета

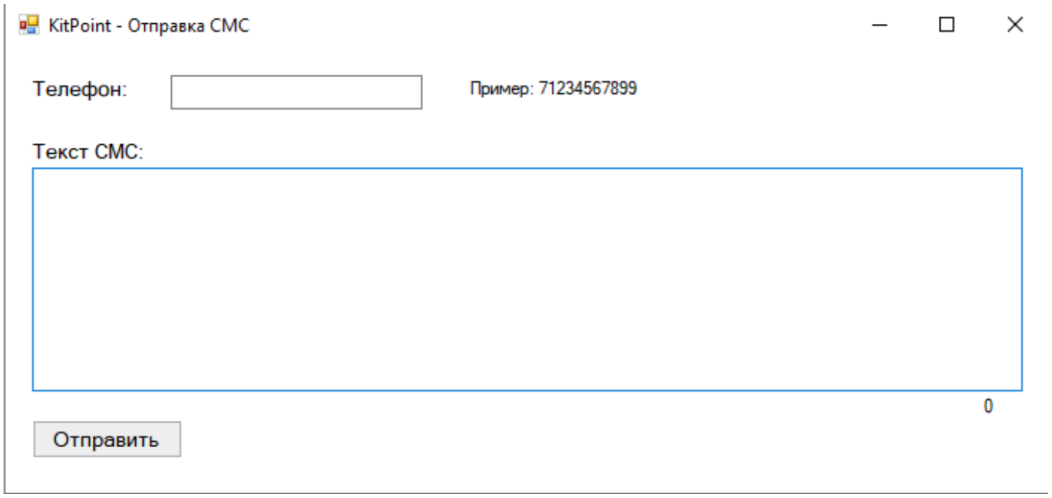

Рисунок 2.28 – Экранная форма отправки СМС

ЗАКЛЮЧЕНИЕ

В ходе выполнения выпускной квалификационной работы были решены все основные задачи, связанные с проектированием, разработкой и вводом в эксплуатацию системы отчетности и информирования клиентов для CRM-системы стоматологической клиники.

Было выполнено следующее:

- выбран сервис отправки СМС – Sms.ru;
- выбран язык программирования – C#;
- спроектирована структура базы данных;
- спроектирована архитектура системы;
- реализован модуль обработки данных;
- реализован модуль отправки СМС;
- реализован модуль администрирования и отчетности;
- проведено тестирование работы модуля обработки данных, модуля отправки СМС и модуля администрирования и отчетности.

В настоящее время система введена в эксплуатацию в сети стоматологических клиник «Белый кит» и обеспечивает высокий уровень автоматизации всех связанных с ней процессов.

БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1. Веб-сайт «SMS Aero». [Электронный ресурс]. URL: https://smsaero.ru. Дата обращения: 16.05.2019.

2. Веб-сайт «SMSC». [Электронный ресурс]. URL: https://smsc.ru. Дата обращения: 16.05.2019.

3. Веб-сайт «SMS.ru». [Электронный ресурс]. URL: https://sms.ru. Дата обращения: 09.06.2019.

4. Веб-сайт «SMS GOROD». [Электронный ресурс]. URL: http://smsgorod.ru. Дата обращения: 16.05.2019.

5. Веб-сайт «WEBSMS». [Электронный ресурс]. URL: http://websms.ru. Дата обращения: 16.05.2019.

6. Веб-сайт «REDSMS». [Электронный ресурс]. URL: https://redsms.ru. Дата обращения: 16.05.2019.

ПРИЛОЖЕНИЕ А

ОПИСАНИЕ ТАБЛИЦ БАЗЫ ДАННЫХ

Таблица А.1 – Описание полей таблицы «template_text_sms»

Наименование	**Тип данных**	**Назначение**
typeSms	smallint	Тип СМС уведомления (первичный ключ)
textSms	varchar	Текст СМС уведомления

Таблица А.2 – Описание полей таблицы «test_status_send_sms»

Наименование	**Тип данных**	**Назначение**
idSms	varchar	Идентификатор СМС уведомления (первичный ключ)
timeTest	datetime	Время когда необходимо проверить статус доставки на SMS.ru
Processed	smallint	Статус обработки
idInf	bigint	Идентификатор в Инфоденте
Id	int	Идентификатор записи (первичный ключ)
typeSms	smallint	Тип СМС уведомления (внешний ключ)

Таблица А.3 – Описание полей таблицы «configuration_care»

Наименование	**Тип данных**	**Назначение**
idConfiguration	int	Идентификатор конфигурации (первичный ключ)
dateOfIncorporation	datetime	Дата включения модуля (первичный ключ)
powerInputProcessing	tinyint	Вкл/выкл модуль обработки данных
timeSend1Day	time	Время отправки СМС уведомления за 1 день
statusSendingSMS1	tinyint	Режим отправки СМС уведомлений первого типа
statusSendingSMS2	tinyint	Режим отправки СМС уведомлений второго типа
statusSendingSMS3	tinyint	Режим отправки СМС уведомлений третьего типа
dateStartSend1Sms	datetime	Дата начала отправки первого СМС уведомления

Таблица А.4 – Описание полей таблицы «the_confirmation_table»

Наименование	Тип данных	Назначение
idInfodent	bigint	Идентификатор в Инфоденте
Status	smallint	Статус подтверждения
Processed	tinyint	Статус обработки
Id	int	Идентификатор записи (первичный ключ)
date	datetime	Дата и время подтверждения

Таблица А.5 – Описание полей таблицы «users»

Наименование	Тип данных	Назначение
idUser	bigint	Идентификатор пользователя (первичный ключ)
fullName	varchar	ФИО пользователя
password	varchar	Пароль

Таблица А.6 – Описание полей таблицы «schedule_of_messages»

Наименование	Тип данных	Назначение
idInfodent	bigint	Идентификатор в инфоденте (первичный ключ)

Продолжение таблицы А.6

timeSendMessage2	datetime	Время отправки второго СМС уведомления
timeSendMessage3	datetime	Время отправки третьего СМС уведомления
dateOfReceipt	datetime	Дата приема (первичный ключ)
wasItSent1	tinyint	Было ли отправлено первое СМС уведомление
wasItSent2	tinyint	Было ли отправлено второе СМС уведомление
wasItSent3	tinyint	Было ли отправлено третье СМС уведомление
theConfirmation	tinyint	Подтверждение приема
idClient	bigint	Идентификатор клиента
timeSendMessage2Total	datetime	Дата и время отправки второго СМС уведомления
obsolete	tinyint	Статус актуальности приема
dateAdd	datetime	Дата добавления

Окончание таблицы А.6

idSMS1	varchar	Идентификатор первого СМС уведомления
idSMS2	varchar	Идентификатор второго СМС уведомления
idSMS3	varchar	Идентификатор третьего СМС уведомления
status1	smallint	Статус отправки первого СМС уведомления
status2	smallint	Статус отправки второго СМС уведомления
status3	smallint	Статус отправки третьего СМС уведомления
id	int	Идентификатор записи (первичный ключ)
idFilial	bigint	Идентификатор филиала (внешний ключ)
idDoctor	bigint	Идентификатор доктора (внешний ключ)

Таблица А.7 – Описание полей таблицы «schedule_of_care»

Наименование	Тип данных	Назначение
idInfodent	bigint	Идентификатор в Инфоденте (первичный ключ)
idClient	bigint	Идентификатор клиента
dateOfReceipt	datetime	Дата приема (первичный ключ)
treatcode	bigint	Идентификатор приема
primaryOrSecondary	bigint	Первичный или вторичный прием
clvisit	tinyint	Посещен ли прием
phone	varchar	Телефон клиента
statusClient	bigint	Статус клиента
refusesms	bigint	Статус отказа от СМС уведомлений
nazndis	tinyint	Статус отказа от назначения
theConfirmation	tinyint	Подтверждение приема
obsolete	tinyint	Статус актуальности приема
dateAdd	datetime	Дата добавления
id	int	Идентификатор записи (первичный ключ)

Окончание таблицы А.7

idFilial	bigint	Идентификатор филиала (внешний ключ)
idDoctor	bigint	Идентификатор доктора (внешний ключ)

Таблица А.8 – Описание полей таблицы «address_filial»

Наименование	**Тип данных**	**Назначение**
idFilial	int	Идентификатор филиала (первичный ключ)
adress	varchar	Адрес филиала

Таблица А.9 – Описание полей таблицы «doctors»

Наименование	**Тип данных**	**Назначение**
idDoctor	int	Идентификатор доктора (первичный ключ)
fullName	varchar	ФИО доктора

Таблица А.10 – Индексы

Таблица	**Имя индекса**	**Поля**
template_text_sms	typeSms	typeSms
test_status_send_sms	idSms	idSms

Продолжение таблицы А.10

test_status_send_sms	processed_timeTest	processed, timeTest
configuration_care	idConfiguration	idConfiguration
schedule_of_care	idInfodent_obsolete	idInfodent, obsolete
schedule_of_care	idClient_dateOfReceipt_obsolete	idClient, dateOfReceipt, obsolete
adress_filial	idFilial	idFilial
doctors	idDoctor	idDoctor
schedule_of_messages	dateAdd_wasItSend1_obsolete	dateAdd, wasItSend1, obsolete
schedule_of_messages	idInfodent_obsolete	idInfodent, obsolete
schedule_of_messages	wasItSend2_timeSendMessage2_obsolete	wasItSend2, timeSendMessage2, obsolete
schedule_of_messages	wasItSend3_timeSendMessage3_obsolete	wasItSend3, timeSendMessage3, obsolete
schedule_of_messages	idSms1_obsolete	idSms1, obsolete
schedule_of_messages	idSms2_obsolete	idSms2, obsolete
schedule_of_messages	idSms3_obsolete	idSms3, obsolete

Окончание таблицы А.10

schedule_of_messages	status1_idSms1_obsolete	status1, idSms1, obsolete
schedule_of_messages	status2_idSms2_obsolete	status2, idSms2, obsolete
schedule_of_messages	status3_idSms3_obsolete	status2, idSms3, obsolete
schedule_of_messages	timeSendMessage2	timeSendMessage2
schedule_of_messages	idClient_dateOfReceipt_obsolete	idClient, dateOfReceipt, obsolete

ПРИЛОЖЕНИЕ Б

ТАБЛИЦЫ С ОПИСАНИЕМ МЕТОДОВ И ПОЛЕЙ

МОДУЛЯ ОБРАБОТКИ ДАННЫХ

Таблица Б.1 – Методы класса «Program»

Название	**Назначение**
SheduleProcessingFunction()	Процесс обработки новых записей в расписании.
SheduleUpdateFunction()	Процесс обработки обновленных записей в расписании.
ClientUpdateScheduleFunction()	Процесс обработки обновлений в карточке клиента.
ClientsCheckTreat()	Процесс обработки подтверждения приемов.
DeleteScheduleFunction()	Процесс обработки удаленных записей.
Main(string[] args)	Начало программы. Запуск нитей.

Таблица Б.2 – Методы класса «ConnectionDB»

Название	**Назначение**
ConnectionDB()	Конструктор.
newfbcon_co()	Подключение к базе Инфодент ЦБД (Центральная база данных).
newfbww001_con()	Подключение к базе Инфодент БК (Белый Кит).
newfbww002_con()	Подключение к базе Инфодент КИ (Кит Имплант).
newfbww003_con()	Подключение к базе Инфодент СК (Сити Кит).
newfbww005_con()	Подключение к базе Инфодент АК (Аллея Кит).

Продолжение таблицы Б.2

newfbconn_con()	Подключение к базе системы.
TestOpenDBs()	Проверка открытия баз данных.
InsertInScheduleOfMessages (TemplateRecordShedule records)	Вставка записи в таблицу «schedule_of_messages» для отправки сообщений.
InsertInScheduleOfCare (TemplateRecordShedule records)	Вставка записи в таблицу «schedule_of_care», информацию о клиенте.
UpdateRecordsInVoronkaSchedule(long schedid)	Установка статуса обработанной записи в «voronka_schedule» в базе Инфодента.
NewRecordsInScheduleDbInf(DateTime DateAndTimeStart)	Получение списка новых не обработанных записей в графике.
SelectRecordsInInfodent(long schedid)	Выгрузка полей из Инфодента для записи в графике.
UpdateDateIncorporation()	Обновление даты начала работы модуля обработки данных.
StatusPowerInputProcessing()	Статус запуска модуля обработки данных (настраивается в модуле администрирования и отчетности).
SelectPhoneAndStatusAndOtherClientInInfodent(TemplateRecordShedule record)	Получение номера и статуса клиента из базы Инфодента.
NewRecordsInVoronkaScheduleUpdate()	Получение списка обновленных записей в Инфоденте, но еще не обработанных.
UpdateInScheduleOfCare(TemplateRecordShedule records)	Обновление записи в таблице «schedule_of_care».

Продолжение таблицы Б.2

UpdateInScheduleOfMessages(TemplateRecordShedule records)	Обновление записи в таблице «schedule_of_messages».
CheckedRecordToTableOfMessages(long schedid)	Проверка существования записи в таблице «schedule_of_messages».
CheckedRecordToTableOfSchedule(long schedid)	Проверка существования записи в таблице «schedule_of_care».
SelectStatusTwoSMS(long treatcode)	Получение статуса отправки сообщения за день до приема.
SelectStatusSms(long treatcode)	Получение статусов отправки сообщений по приему.
UpdateRecordsInVoronkaScheduleUpdate(long schedid)	Установка статуса обработанной записи в таблице Инфодента «voronka_schedule_update».
NewRecordsClientsInDbInf()	Получение списка обновленных записей, связанных с обновлением в карте клиента, которые еще не были обработаны.
SelectRecordClient(long pcode)	Получение необходимых полей из таблицы «clients».
UpdateRecordsInVoronkaUpdateClient(long pcode)	Установка статуса обработанной записи в таблице Инфодента «voronka_schedule_update».
UpdateInScheduleOfCareClientColumns(TemplateRecordClient records)	Обновление записи в таблице «schedule_of_care».

Продолжение таблицы Б.2

UpdateInScheduleOfMessages Client(TemplateRecordClient records)	Обновление записи в таблице «schedule_of_messages».
SelectTreatByClient(long pcode)	Выгрузка всех записей по клиенту из таблицы «schedule_of_messages».
SelectTreatByClientScheduleof care(long pcode)	Выгрузка всех записей по клиенту из таблицы «schedule_of_care».
AddStatusConfirmation(short status, long code)	Добавление статуса подтверждения приема или отказа в таблицы «schedule_of_messages» и «schedule_of_care».
SelectTheConfirmation(long code)	Выборка подтверждений приема.
SelectTheDateOfReceipt(long code)	Выбор даты приема.
UpdateConfirmationInfodent(long code)	Добавление статуса подтверждения в Инфодент.
TestExists(long schid)	Проверка существования записи в расписании при обновлении врача или даты и времени приема.
SelectStatusRecord(long schedid)	Проверка первичный или вторичный прием.
UpdateConfirmationInfodentStop(long code)	Обновление в расписании Инфодента при отказе от приема.
SelectIdFilialRecord(long code)	Получение идентификатора филиала, в котором будет прием.
TestObsolete(TemplateRecord Shedule newRecord)	Проверка изменений в расписании.

Окончание таблицы Б.2

SelectOldMarginsRecord(long idInf)	Выборка полей из старой записи.
ObsoleteRecord(long idInf)	Устаревание записи.
NewRecordsInVoronkaScheduleDelete()	Получение списка удаленных записей в расписании.
UpdateRecordsInVoronkaScheduleDelete(long schedid)	Установка статуса обработанной записи в таблице Инфодента «voronka_schedule_delete».
InsertTheConfirmation(long schid, int status)	Запись подтверждения или отказа от приема в таблицу «the_confirmation_table».

Таблица Б.3 – Поля класса «TemplateRecordClient»

Название	**Назначение**
idClient	Идентификатор клиента.
phone	Телефон.
statusClient	Статус клиента.
refusesms	Отказ от СМС.
nazndis	Запись на прием запрещена.

Таблица Б.4 – Поля класса «TemplateRecordShedule»

Название	**Назначение**
idRecordInfodent	Идентификатор в Инфоденте.
idClient	Идентификатор клиента.
idDoctor	Идентификатор доктора.
workDate	Дата приема.

Окончание таблицы Б.4

bhour	Время начала, час.
bmin	Время начала, мин.
fhour	Время окончания, час.
fmin	Время окончания, мин.
treatcode	Идентификатор приема.
clvisit	Был ли визит.
status	Первичный или вторичный.
filial	Филиал приема.
phone	Номер телефона клиента.
statusClient	Статус клиента.
refusesms	Отказ от СМС.
nazndis	Запись на прием запрещена.
dateAdd	Дата добавления записи.

Таблица Б.5 – Поля класса «TemplateTestRecord»

Название	**Назначение**
idRecordInfodent	Идентификатор в Инфоденте.
idDoctor	Идентификатор доктора.
dateOfReceipt	Дата и время приема.

Таблица Б.6 – Поля класса «StatusSmsModel»

Название	**Назначение**
wasItSent2	Была ли отправлена СМС за день до приема.
wasItSent3	Была ли отправлена СМС за 3 часа до приема.

Таблица Б.7 – Методы класса «Parse»

Название	**Назначение**
Parse()	Конструктор.
GetResponse(string uri)	Получение кода странички.
ParseAnswerTreat(string data)	Получение подтверждений или отказов от приема.
DeleteCodes()	Очистка подтверждений или отказов.

ПРИЛОЖЕНИЕ В
ТАБЛИЦЫ С ОПИСАНИЕ МЕТОДОВ И ПОЛЕЙ МОДУЛЯ ОТПРАВКИ СМС

Таблица В.1 – Методы класса «Program»

Название	**Назначение**
Send1SMS()	Вызов отправки СМС после записи на прием.
Send2SMS()	Вызов отправки СМС за день до приема.
Send3SMS()	Вызов отправки СМС за 3 часа до приема.
SendSmsRu(string phone, string textSms, long idInf, short typeSms)	Отправка СМС через сервис SMS.ru.
GetStatusSmsRu(string id, short type)	Получение статуса доставки СМС.
UpdateStatusSMS1()	Обновление статуса доставки СМС после записи на прием.
UpdateStatusSMS2()	Обновление статуса доставки СМС за день до приема.
UpdateStatusSMS3()	Обновление статуса доставки СМС за 3 часа до приема.
UpdateStatusSMS()	Вызов обновления статусов доставки.
TestSendSmsFunction()	Проверка отправки СМС.
Main(string[] args)	Начало программы. Запуск нитей.

Таблица В.2 – Методы класса «ConnectionDB»

Название	**Назначение**
ConnectionDB()	Конструктор.
newfbcon_co()	Подключение к базе Инфодент ЦБД.
newfbww001_con()	Подключение к базе Инфодент БК.

Продолжение таблицы В.2

newfbww002_con()	Подключение к базе Инфодент КИ.
newfbww003_con()	Подключение к базе Инфодент СИ.
newfbww005_con()	Подключение к базе Инфодент АК.
newfbconn_con()	Подключение к базе системы.
SMS1(DateTime TimeSend1Sms)	Отправка СМС о записи на прием.
UpdateWasItSent1(long idInf)	Обновление статуса отправки СМС о записи на прием.
UpdateDateSend1Sms()	Обновление даты старта отправки СМС о записи на прием.
GetDname(long idDoctor)	Получение имени врача.
SMS2()	Отправка СМС за день до приема.
UpdateWasItSent2(long idInf)	Обновление статуса отправки СМС за день до приема.
UpdateInfodentWithConfirmation(long idInf)	Отметка в базе что прием был ранее подтвержден.
SMS3()	Отправка СМС за 3 часа до приема.
UpdateWasItSent3(long idInf)	Обновление статуса отправки СМС за 3 часа до приема.
TestOpenDBs()	Проверка открытия базы данных
InsertTestStatusSendSms(string idSms, short typeSms, long idInf)	Добавление записи в таблицу «test_status_send_sms», для проверки статуса доставки через 15 минут.
InsertTextSms(string text)	Добавление текста СМС для отправки.
TestConfirmation(long idInf)	Проверка подтверждения в базе Инфодент.

Продолжение таблицы В.2

TimeSend1Day()	Получение времени отправки СМС за день до приема.
GetStatusSendingSMS1()	Получение статуса запуска СМС о записи на прием.
GetStatusSendingSMS2()	Получение статуса запуска СМС за день до приема.
GetStatusSendingSMS3()	Получение статуса запуска СМС за три часа до приема.
GetText(int type)	Выбор шаблона текста СМС.
GetFirstName(TemplateRecordShedule record)	Получение имени клиента.
GetPhoneAndIdDoctor(TemplateRecordShedule record)	Получение телефона клиента и идентификатора врача.
GetIformationClient(long idInfodent)	Получение информации о клиенте.
InsertStatusSMS(string id, long idInf, short typeSms)	Добавление идентификаторов СМС в таблицу «schedule_of_messages».
UpdateStatusSMSRu(string id, string status, short type)	Обновление статуса доставки СМС.
GetIdSms1Ru()	Получение идентификатора СМС о записи на прием.
GetIdSms2Ru()	Получение идентификатора СМС за день до приема.
GetIdSms3Ru()	Получение идентификатора СМС за три часа до приема.

Продолжение таблицы В.2

GetAdressFilial(long idFilial)	Получение адреса филиала, в котором будет прием.
CreateReport()	Создание большого отчета.
GetRuStatusSms(short ruStatus)	Расшифровка статуса доставки сообщения с сервиса SMS.ru.
GetKitStatusSms(short ruStatus)	Статусы отправки СМС в Инфоденте.
GetInformation()	Получение информации для большого отчета.
GetPhone1(long idClient)	Получение домашнего телефона.
GetPhone2(long idClient)	Получение рабочего телефона.
GetPhone3(long idClient)	Получение мобильного телефона.
GetFullname(long idClient)	Получение ФИО клиента.
GetHistnum(long idClient)	Получение номера карты клиента.
GetFilial(long idFilial)	Получение короткого имени филиала, в котором будет прием.
SelectStatusSendSms()	Выбор СМС, у которых необходимо проверить статус доставки СМС.
GetStatusSmsRu(string id, short typeSms, long idInf)	Получение статуса доставки СМС с сервиса SMS.ru
AddCommentStatusSms(string status, short typeSms, long idInf)	Добавление статуса отправки СМС в расписание в Инфоденте.
SelectIdFilialRecord(long code)	Получение филиала, в котором должен состояться прием.
UpdateTestSendSms(string idSms)	Обновление статуса процесса проверки доставки СМС.

Окончание таблицы В.2

GetBalanseRu()	Получение баланса.

Таблица В.3 – поля класса «ReportModel»

Название	**Назначение**
schedid	Идентификатор в расписании Инфодента.
dateOfReceipt	Дата и время приема.
filial	Филиал приема.
histnum	Номер карты пациента.
fullname	ФИО пациента.
dname	Имя врача.
phone1	Домашний телефон.
phone2	Рабочий телефон.
phone3	Мобильный телефон.
theConfirmation	Подтверждение приема.
kitStatusSms1	Статус доставки СМС о записи на прием в расписании Инфодента.
kitStatusSms2	Статус доставки СМС за день до приема в расписании Инфодента.
kitStatusSms3	Статус доставки СМС за три часа до приема в расписании Инфодента.
RuStatusSms1	Статус доставки СМС о записи на прием.
RuStatusSms2	Статус доставки СМС за день до приема.
RuStatusSms3	Статус доставки СМС за три часа до приема.
timeSendMessage2Total	Время отправки СМС за день до приема.
idFilial	Идентификатор филиала приема.

Окончание таблицы В.3

idClient	Идентификатор клиента.
obsolete	Актуальность записи.
idDoctor	Идентификатор врача.
dateAdd	Дата добавления в расписание Инфодента.

Таблица В.4 – поля и метод класса «TemplateRecordShedule»

Название	**Назначение**
idRecordInfodent	Идентификатор в Инфоденте.
idClient	Идентификатор клиента.
idDoctor	Идентификатор врача.
phoneNumber	Номер телефона клиента.
clientName	Имя клиента.
filial	Филиал приема.
dateOfReceipt	Дата и время приема.
AddInformation(long idInfodent)	Добавление информации о клиенте.

Таблица В.5 – поля класса «TestSendSmsModel»

Название	**Назначение**
idSms	Идентификатор СМС.
typeSms	Тип СМС.
idInf	Идентификатор в Инфоденте.

ПРИЛОЖЕНИЕ Г
ТАБЛИЦЫ С ОПИСАНИЕМ МЕТОДОВ И ПОЛЕЙ МОУЛЯ АДМИНИСТРИРОВАНИЯ И ОТЧЕТНОСТИ

Таблица Г.1 – Методы класса «Program»

Название	**Назначение**
Main()	Начало программы. Запуск Авторизации.

Таблица Г.2 – Методы класса «ConnectionDB»

Название	**Назначение**
ConnectionDB()	Конструктор.
newfbconn_con()	Подключение к базе системы.
GetStatusPowerInputProcessing()	Получение статуса запуска обработки новых приемов.
UpdateStatusCase(int numCase, int status)	Обновление статуса запуска отправки СМС и обработки новых приемов.
GetStatusSendingSMS1()	Получение статуса запуска СМС о записи на прием.
GetStatusSendingSMS2()	Получение статуса запуска СМС за день до приема.
GetStatusSendingSMS3()	Получение статуса запуска СМС за три часа до приема.
AddText(int type, string text)	Добавление текста в шаблоны СМС.
GetText(int type)	Получение текста шаблонов СМС.
GetInformation(int typeReport, string histnum, DateTime ntime, DateTime ktime)	Получение информации для отчетов.
GetInformationStatisticalReport()	Получение информации для статистического отчета.

Окончание таблицы Г.2

GetPcode(string histnum)	Получение идентификатора пациента.
GetDname(long idDoctor)	Получение ФИО доктора.
GetIdDoctor(long schedid)	Получение идентификатора врача.
GetPhone1(long idClient)	Получение домашнего телефона.
GetPhone2(long idClient)	Получение рабочего телефона.
GetPhone3(long idClient)	Получение мобильного телефона.
GetFullname(long idClient)	Получение ФИО пациента.
GetHistnum(long idClient)	Получение номера карты пациента.
GetFilial(long idFilial)	Получение филиала.
GetUsersName()	Получение списка имен пользователей.
TestPasswordUsers(string userName, string password)	Проверка пароля.

Таблица Г.3 – поля класса «ReportModel»

Название	**Назначение**
schedid	Идентификатор в расписании Инфодента.
dateOfReceipt	Дата и время приема.
filial	Филиал приема.
histnum	Номер карты пациента.
fullname	ФИО пациента.
dname	Имя врача.
phone1	Домашний телефон.
phone2	Рабочий телефон.
phone3	Мобильный телефон.
theConfirmation	Подтверждение приема.

Окончание таблицы Г.3

kitStatusSms1	Статус доставки СМС о записи на прием в расписании Инфодента.
kitStatusSms2	Статус доставки СМС за день до приема в расписании Инфодента.
kitStatusSms3	Статус доставки СМС за три часа до приема в расписании Инфодента.
RuStatusSms1	Статус доставки СМС о записи на прием.
RuStatusSms2	Статус доставки СМС за день до приема.
RuStatusSms3	Статус доставки СМС за три часа до приема.
timeSendMessage2Total	Время отправки СМС за день до приема.
idFilial	Идентификатор филиала приема.
idClient	Идентификатор клиента.
obsolete	Актуальность записи.
idDoctor	Идентификатор врача.
dateAdd	Дата добавления в расписание Инфодента.
refSMS	Отказ от СМС.
clVisit	Явка на прием.

Таблица Г.4 – поля класса «SendSmsEmail»

Название	**Назначение**
getBalanse()	Получение баланса.
sendSms(string phone, string textSms)	Отправка СМС.
getStatusSms(long id)	Получение статуса доставки.

Окончание таблицы Г.4

sendEmail(string subject, string body)	Отправление на Email.

Таблица Г.5 – поля класса «Authorization»

Название	**Назначение**
Authorization()	Конструктор. Инициализация формы. Выбор имени и ввод пароля.
button1_Click(object sender, EventArgs e)	Кнопка «Вход». Проверка пароля.

Таблица Г.6 – поля класса «GlobalForm»

Название	**Назначение**
GlobalForm(string userName)	Конструктор. Инициализация формы.
button1_Click(object sender, EventArgs e)	Кнопка «Настройка кейсов». Открытие формы настройки отправки СМС уведомлений и обработки новых приемов.
GlobalForm_FormClosed(object sender, FormClosedEventArgs e)	Закрытие главной формы.
button2_Click(object sender, EventArgs e)	Кнопка «Настройка шаблонов». Открытие формы управления шаблонами.
button3_Click(object sender, EventArgs e)	Кнопка «Универсальный отчет». Открытие формы настройки отчета.
button4_Click(object sender, EventArgs e)	Кнопка «Отправка СМС». Открытие формы отправки СМС.

Таблица Г.7 – поля класса «SendSms»

Название	**Назначение**
SendSms()	Конструктор. Инициализация формы.
button1_Click(object sender, EventArgs e)	Кнопка «Отправить». Открывает подтверждение отправки.
button3_Click(object sender, EventArgs e)	Кнопка «Нет» на вкладке «Подтвердить отправку». Закрывает вкладку «Подтвердить отправку».
button2_Click(object sender, EventArgs e)	Кнопка «Да» на вкладке «Подтвердить отправку». Отправляет СМС.

Таблица Г.8 – поля класса «SettingsCase»

Название	**Назначение**
SettingsCase()	Конструктор. Инициализация формы.
radioButton1_CheckedChanged(object sender, EventArgs e)	Включение обработки новых приемов.
radioButton2_CheckedChanged(object sender, EventArgs e)	Выключение обработки новых приемов.
radioButton3_CheckedChanged(object sender, EventArgs e)	Включение отправки СМС о записи на прием.
radioButton4_CheckedChanged(object sender, EventArgs e)	Выключение отправки СМС о записи на прием.
radioButton5_CheckedChanged(object sender, EventArgs e)	Включение отправки СМС за день до приема.
radioButton6_CheckedChanged(object sender, EventArgs e)	Выключение отправки СМС за день до приема.

Окончание таблицы Г.8

radioButton7_CheckedChanged(object sender, EventArgs e)	Включение отправки СМС за три часа до приема.
radioButton8_CheckedChanged(object sender, EventArgs e)	Выключение отправки СМС за три часа до приема.

Таблица Г.9 – поля класса «SettingsReport»

Название	**Назначение**
SettingsReport()	Конструктор. Инициализация формы.
button1_Click(object sender, EventArgs e)	Кнопка «Создать». Создание выбранного отчета.
checkBox1_Click(object sender, EventArgs e)	Установка checkbox «Выборка по ИНП».
checkBox2_Click(object sender, EventArgs e)	Установка checkbox «Выборка по дате приема».
creatStatisticalReport()	Создание статистического отчета.
createReport()	Создание универсального отчета.
checkBox3_Click(object sender, EventArgs e)	Установка checkbox «Статистический отчет».

Таблица Г.10 – поля класса «SettingsTemplate »

Название	**Назначение**
SettingsTemplate ()	Конструктор. Инициализация формы.
button1_Click(object sender, EventArgs e)	Кнопка «Редактировать». Изменяется текст шаблона СМС о записи на прием.

Окончание таблицы Г.10

button2_Click(object sender, EventArgs e)	Кнопка «Редактировать». Изменяется текст шаблона СМС за день до приема.
button3_Click(object sender, EventArgs e)	Кнопка «Редактировать». Изменяется текст шаблона СМС за три часа до приема.
textBox1_KeyUp(object sender, KeyEventArgs e)	Вызывает метод подсчета символов в шаблоне СМС о записи на прием.
textBox2_KeyUp(object sender, KeyEventArgs e)	Вызывает метод подсчета символов в шаблоне СМС за день до приема.
textBox3_KeyUp(object sender, KeyEventArgs e)	Вызывает метод подсчета символов в шаблоне СМС за три часа до приема.
characterCount(int i)	Считает количество символов в шаблоне.

ПРИЛОЖЕНИЕ Д

ИСХОДНЫЙ КОД INPUTPROCESSING.PROGRAM

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading;
using System.Threading.Tasks;
using FirebirdSql.Data.FirebirdClient;
using InputProcessingFB3.Function;
using test2Con;

namespace InputProcessingFB3
{
    class Program
    {
        static void SheduleProcessingFunction() //функция процесса обработки новых записей в
расписании
        {
            ConnectionDB dB = new ConnectionDB();
            int UpgrateTimeProcessing = 1;
            DateTime TimeProcessing = DateTime.Now;
            while (true)
            {
                try
                {
                    if (dB.testOpenDBs() == 1)
                    {
                        if (dB.StatusPowerInputProcessing() == 1)
                        {
                            if (UpgrateTimeProcessing == 1)
                            {
                                TimeProcessing = DateTime.Parse(dB.UpdateDateIncorporation());
                                UpgrateTimeProcessing = 0;
                                Console.WriteLine("Запущен:" + DateTime.Now.ToString());
                            }

                            List<long> schedids = new List<long>();
                            schedids.AddRange(dB.NewRecordsInScheduleDbInf(TimeProcessing));
                            if (schedids.Count != 0)
                            {
                                foreach (var schedid in schedids)
                                {
                                    int status = 0;
                                    try
                                    {
                                        TemplateRecordShedule tempRecord =
dB.SelectRecordsInInfodent(schedid);
                                        if (tempRecord.IdClient != 0 && tempRecord.IdDoctor != 0
&& tempRecord.IdRecordInfodent != 0)
                                        {
                                            status = dB.InsertInScheduleOfCare(tempRecord);
                                            if (status == 1)
                                            {
                                                status =
dB.InsertInScheduleOfMessages(tempRecord);
                                            }
                                        }
                                        if (status == 1)
                                        {
                                            dB.UpdateRecordsInVoronkaSchedule(schedid);
                                        }
                                    }
                                    catch (Exception ex)
                                    {
                                        Console.WriteLine($"IP1 : {ex.Message}");
```

```
                                }
                            }
                        }
                    }
                    else
                    {
                        UpgrateTimeProcessing = 1;
                    }
                }
                else
                {
                    Console.WriteLine("Нет подключения к БД!");
                    UpgrateTimeProcessing = 1;
                    Thread.Sleep(60000);
                }
                Thread.Sleep(6000);
            }
            catch (Exception ex)
            {
                Console.WriteLine($"40 : {ex.Message}");
            }
        }
    }

    static void SheduleUpdateFunction() // процесс обработки обновленных записей в расписании
    {
        ConnectionDB dB = new ConnectionDB();
        while (true)
        {
            try
            {
                if (dB.testOpenDBs() == 1)
                {
                    if (dB.StatusPowerInputProcessing() == 1)
                    {
                        List<long> schedids = new List<long>();
                        schedids.AddRange(dB.NewRecordsInVoronkaScheduleUpdate());
                        if (schedids.Count != 0)
                        {
                            foreach (var schedid in schedids)
                            {
                                int status = 0;
                                TemplateRecordShedule tempRecord =
dB.SelectRecordsInInfodent(schedid);
                                if (dB.testObsolete(tempRecord) == 0)
                                {
                                    status = dB.UpdateInScheduleOfCare(tempRecord);
                                    if (status == 1)
                                    {
                                        status = dB.UpdateInScheduleOfMessages(tempRecord);
                                    }
                                }
                                else // изменилось время, либо врач
                                {
                                    dB.obsoleteRecord(schedid);
                                    if (dB.testExists(schedid) == 1)
                                    {
                                        status = dB.InsertInScheduleOfCare(tempRecord);
                                        if (status == 1)
                                        {
                                            status =
dB.InsertInScheduleOfMessages(tempRecord);
                                        }
                                    }
                                }
                                if (status == 1)
                                {
```

```
                                    dB.UpdateRecordsInVoronkaScheduleUpdate(schedid);
                                }
                            }
                        }
                    }
                }
                else
                {
                    Console.WriteLine("Нет подключения к БД!");
                    Thread.Sleep(60000);
                }

                Thread.Sleep(6000);
            }
            catch (Exception ex)
            {
                Console.WriteLine($"41 : {ex.Message}");
            }
        }
    }

    static void ClientUpdateScheduleFunction() // Процесс обработки обновлений в карточке
клиента
    {
        ConnectionDB dB = new ConnectionDB();
        while (true)
        {
            try
            {
                if (dB.testOpenDBs() == 1)
                {
                    if (dB.StatusPowerInputProcessing() == 1)
                    {
                        List<long> pcodes = new List<long>();
                        pcodes.AddRange(dB.NewRecordsClientsInDbInf());
                        if (pcodes.Count != 0)
                        {
                            foreach (var pcode in pcodes)
                            {
                                int status = 0;
                                TemplateRecordClient tempRecord =
dB.SelectRecordClient(pcode);
                                status = dB.UpdateInScheduleOfCareClientColumns(tempRecord);
                                if (status == 1)
                                {
                                    status = dB.UpdateInScheduleOfMessagesClient(tempRecord);
                                }
                                if (status == 1)
                                {
                                    dB.UpdateRecordsInVoronkaUpdateClient(pcode);
                                }
                            }
                        }
                    }
                }
                else
                {
                    Console.WriteLine("Нет подключения к БД!");
                    Thread.Sleep(60000);
                }

                Thread.Sleep(6000);
            }
            catch (Exception ex)
            {
                Console.WriteLine($"42 : {ex.Message}");
```

```
                }
            }
        }

        static void ClientsCheckTreat() // Процесс обработки обновлений в карточке клиента - подтверждение
        {
            ConnectionDB dB = new ConnectionDB();
            Parse parse = new Parse();
            while (true)
            {
                try
                {
                    if (dB.testOpenDBs() == 1)
                    {
                        string parseText = parse.getResponse("");
                        foreach (TemplateTestConfirmation code in parse.parseAnswerTreat(parseText))//написать парсер даты и доделать вставку новых подтверждений
                        {
                            if (!dB.testExistsTheConfirmation(code))
                            {
                                dB.insertTheConfirmation(code);
                            }
                        }

                        parseText = parse.getResponse("");
                        foreach (TemplateTestConfirmation code in parse.parseAnswerTreat(parseText))
                        {
                            if (!dB.testExistsTheConfirmation(code))
                            {
                                dB.insertTheConfirmation(code);
                            }
                        }

                        List<TemplateTestConfirmation> confirmations = dB.selectConfirmation();
                        foreach (var confirmation in confirmations)
                        {
                            var result = dB.AddStatusConfirmation(confirmation.status, confirmation.schedid);

                            if (result.Item1)
                            {
                                dB.updateTheconfirmationtableProcessed(confirmation, result.Item2);
                            }
                        }

                        if (DateTime.Now.Hour > 1 && DateTime.Now.Hour < 5)
                        {
                            parse.deleteCodes();
                        }
                    }
                    else
                    {
                        Console.WriteLine("Нет подключения к БД!");
                        Thread.Sleep(60000);
                    }
                    Thread.Sleep(10000); //10 сек
                }
                catch (Exception ex)
                {
                    Console.WriteLine($"43 : {ex.Message}");
                }
            }
```

```
}


static void DeleteScheduleFunction() // Процесс обработки удалений записей
{
    ConnectionDB dB = new ConnectionDB();
    while (true)
    {
        try
        {
            if (dB.testOpenDBs() == 1)
            {
                if (dB.StatusPowerInputProcessing() == 1)
                {
                    List<long> pcodes = new List<long>();
                    pcodes.AddRange(dB.NewRecordsInVoronkaScheduleDelete());
                    if (pcodes.Count != 0)
                    {
                        foreach (var pcode in pcodes)
                        {
                            int status = 0;
                            status = dB.obsoleteRecord(pcode);
                            if (status == 1)
                            {
                                dB.UpdateRecordsInVoronkaScheduleDelete(pcode);
                            }
                        }
                    }
                }
            }
            else
            {
                Console.WriteLine("Нет подключения к БД!");
                Thread.Sleep(60000);
            }

            Thread.Sleep(120000);
        }
        catch (Exception ex)
        {
            Console.WriteLine($"44 : {ex.Message}");
        }
    }
}

static void DeleteOldInformationInTables() // Процесс удаления устаревшей информации
{
    ConnectionDB dB = new ConnectionDB();
    while (true)
    {
        try
        {
            if (dB.testOpenDBs() == 1)
            {
                dB.deleteOldInformationTableVoronkaSchedule();
                dB.deleteOldInformationTableVoronkaScheduleUpdate();
                dB.deleteOldInformationTableVoronkaScheduleDelete();
                dB.deleteOldInformationTableVoronkaUpdateClient();
                dB.deleteOldInformationTableScheduleofcare();
                dB.deleteOldInformationTableScheduleofmessages();
                dB.deleteOldInformationTableSmssend();
                dB.deleteOldInformationTableTeststatussendsms();
                dB.deleteOldInformationTableTheconfirmationtable();
            }
            else
            {
```

```
                    Console.WriteLine("Нет подключения к БД!");
                    Thread.Sleep(60000);
                }

                Thread.Sleep(6000 * 60 * 24);
            }
            catch (Exception ex)
            {
                Console.WriteLine($"44 : {ex.Message}");
                Thread.Sleep(6000);
            }
        }
    }

    static void Main(string[] args)
    {
        ConnectionDB dB = new ConnectionDB();
        dB.UpdateDateIncorporation();
        Thread ShedulеProcessing = new Thread(SheduleProcessingFunction);
        Thread SheduleUpdate = new Thread(SheduleUpdateFunction);
        Thread ClientUpdateSchedule = new Thread(ClientUpdateScheduleFunction);
        Thread CheckTreat = new Thread(ClientsCheckTreat);
        Thread ScheduleDelete = new Thread(DeleteScheduleFunction);
        //Thread DeleteOldInformation = new Thread(DeleteOldInformationInTables);

        SheduleProcessing.Start();
        SheduleUpdate.Start();
        ClientUpdateSchedule.Start();
        CheckTreat.Start();
        ScheduleDelete.Start();
        //DeleteOldInformation.Start();


        while (true)
        {
            Thread.Sleep(100);
        }
}
}
}
```

ПРИЛОЖЕНИЕ Е

ИСХОДНЫЙ КОД INPUTPROCESSING.CONNECTIONDB

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using FirebirdSql.Data.FirebirdClient;
using MySql.Data.MySqlClient;
using InputProcessingFB3.Models;
using test2Con;

namespace InputProcessingFB3
{
    class ConnectionDB
    {
        public ConnectionDB()
        {/////.....///////
        }/////.....///////

        public FbConnection newfbcon_co()
        {
            FbConnection fbco;
            FbConnectionStringBuilder fbco_con = new FbConnectionStringBuilder();
            //fbco_con.Charset = "WIN1251";
            fbco_con.UserID = "";
            fbco_con.Password = "";
            fbco_con.Database = "";
            fbco_con.ServerType = 0;
            fbco_con.ConnectionLifeTime = 60;
            fbco = new FbConnection(fbco_con.ToString());

            return fbco;
        }

        public FbConnection newfbww001_con()
        {
            FbConnection fbww001;
            // Подключение к базе ww001
            FbConnectionStringBuilder fbww001_con = new FbConnectionStringBuilder();
            //fbww001_con.Charset = "WIN1251";
            fbww001_con.UserID = "";
            fbww001_con.Password = "";
            fbww001_con.Database = "";
            fbww001_con.ServerType = 0;
            fbww001_con.ConnectionLifeTime = 60;
            fbww001 = new FbConnection(fbww001_con.ToString());

            return fbww001;
        }

        public FbConnection newfbww002_con()
        {
            FbConnection fbww002;
            // Подключение к базе ww002
            FbConnectionStringBuilder fbww002_con = new FbConnectionStringBuilder();
            // fbww002_con.Charset = "WIN1251";
            fbww002_con.UserID = "";
            fbww002_con.Password = "";
            fbww002_con.Database = "";
            fbww002_con.ServerType = 0;
            fbww002_con.ConnectionLifeTime = 60;
            fbww002 = new FbConnection(fbww002_con.ToString());

            return fbww002;
        }
```

```
public FbConnection newfbww003_con()
{
    FbConnection fbww003;
    // Подключение к базе ww003
    FbConnectionStringBuilder fbww003_con = new FbConnectionStringBuilder();
    // fbww003_con.Charset = "WIN1251";
    fbww003_con.UserID = "";
    fbww003_con.Password = "";
    fbww003_con.Database = "";
    fbww003_con.ServerType = 0;
    fbww003_con.ConnectionLifeTime = 60;
    fbww003 = new FbConnection(fbww003_con.ToString());

    return fbww003;
}

public FbConnection newfbww005_con()
{
    FbConnection fbww005;
    // Подключение к базе ww005
    FbConnectionStringBuilder fbww005_con = new FbConnectionStringBuilder();
    //fbww005_con.Charset = "UTF8";
    fbww005_con.UserID = "";
    fbww005_con.Password = "";
    fbww005_con.Database = "";
    fbww005_con.ServerType = 0;
    fbww005_con.ConnectionLifeTime = 60;
    fbww005 = new FbConnection(fbww005_con.ToString());

    return fbww005;
}

public FbConnection newfbconn_con()
{
    FbConnection fbconn;
    FbConnectionStringBuilder fbconn_con = new FbConnectionStringBuilder();
    //fbco_con.Charset = "WIN1251";
    fbconn_con.UserID = "";
    fbconn_con.Password = "";
    fbconn_con.Database = "";
    fbconn_con.ServerType = 0;
    fbconn_con.ConnectionLifeTime = 60;
    fbconn = new FbConnection(fbconn_con.ToString());

    return fbconn;
}

public int testOpenDBs() // Проверка открытия баз данных
{
    using (FbConnection fbco = newfbcon_co())
    using (FbConnection conn = newfbconn_con())
    {

        int test = 0;
        try
        {
            fbco.Open();
            conn.Open();
            test = 1;
        }
        catch (Exception ex)
        {
            Console.WriteLine($"45 : {ex.Message}");
            test = 0;
            if (ex.Message == "Error reading data from the connection.")
            {
```

```
                    FbConnection.ClearPool(fbco);
                }
            }
            finally
            {
                fbco.Close();
                conn.Close();
            }
            return test;
        }
    }

    public void deleteOldInformationTableVoronkaSchedule()// удаление устаревшей информации
из таблицы voronka_schedule
    {
        using (FbConnection fbco = newfbcon_co())
        {
            string sqlstring = String.Format($"delete from voronka_schedule where
dateandtimeinsert < dateadd(month, -6, current_date)");
            FbCommand InsertSQL =
               new FbCommand(sqlstring, fbco);// Добавить аргументы
            try
            {
                fbco.Open();
                FbTransaction fbt = fbco.BeginTransaction();
                InsertSQL.Transaction = fbt;
                int res = InsertSQL.ExecuteNonQuery();
                fbt.Commit();
            }
            catch (Exception ex)
            {
                Console.WriteLine($"Ошибка при удалении 1 : {ex.Message}");
                if (ex.Message == "Error reading data from the connection.")
                {
                    FbConnection.ClearPool(fbco);
                }
            }
            finally
            {
                InsertSQL.Dispose();
                fbco.Close();
            }
        }
    }

    public void deleteOldInformationTableVoronkaScheduleUpdate()// удаление устаревшей
информации из таблицы voronka_schedule_update
    {
        using (FbConnection fbco = newfbcon_co())
        {
            string sqlstring = String.Format($"delete from voronka_schedule_update where
dateandtimeinsert < dateadd(month, -2, current_date)");
            FbCommand InsertSQL =
               new FbCommand(sqlstring, fbco);// Добавить аргументы
            try
            {
                fbco.Open();
                FbTransaction fbt = fbco.BeginTransaction();
                InsertSQL.Transaction = fbt;
                int res = InsertSQL.ExecuteNonQuery();
                fbt.Commit();
            }
            catch (Exception ex)
            {
                Console.WriteLine($"Ошибка при удалении 2 : {ex.Message}");
                if (ex.Message == "Error reading data from the connection.")
                {
```

```
                    FbConnection.ClearPool(fbco);
                }
            }
            finally
            {
                InsertSQL.Dispose();
                fbco.Close();
            }
        }
    }

    public void deleteOldInformationTableVoronkaScheduleDelete()// удаление устаревшей
информации из таблицы voronka_schedule_delete
    {
        using (FbConnection fbco = newfbcon_co())
        {
            string sqlstring = String.Format($"delete from voronka_schedule_delete where
dateandtimeinsert < dateadd(month, -2, current_date)");
            FbCommand InsertSQL =
               new FbCommand(sqlstring, fbco);// Добавить аргументы
            try
            {
                fbco.Open();
                FbTransaction fbt = fbco.BeginTransaction();
                InsertSQL.Transaction = fbt;
                int res = InsertSQL.ExecuteNonQuery();
                fbt.Commit();
            }
            catch (Exception ex)
            {
                Console.WriteLine($"Ошибка при удалении 3 : {ex.Message}");
                if (ex.Message == "Error reading data from the connection.")
                {
                    FbConnection.ClearPool(fbco);
                }
            }
            finally
            {
                InsertSQL.Dispose();
                fbco.Close();
            }
        }
    }

    public void deleteOldInformationTableVoronkaUpdateClient()// удаление устаревшей
информации из таблицы voronka_update_client
    {
        using (FbConnection fbco = newfbcon_co())
        {
            string sqlstring = String.Format($"delete from voronka_update_client where
dateandtimeinsert < dateadd(month, -2, current_date)");
            FbCommand InsertSQL =
               new FbCommand(sqlstring, fbco);// Добавить аргументы
            try
            {
                fbco.Open();
                FbTransaction fbt = fbco.BeginTransaction();
                InsertSQL.Transaction = fbt;
                int res = InsertSQL.ExecuteNonQuery();
                fbt.Commit();
            }
            catch (Exception ex)
            {
                Console.WriteLine($"Ошибка при удалении 4 : {ex.Message}");
                if (ex.Message == "Error reading data from the connection.")
                {
                    FbConnection.ClearPool(fbco);
```

```
                }
            }
            finally
            {
                InsertSQL.Dispose();
                fbco.Close();
            }
        }
    }

    public void deleteOldInformationTableScheduleofcare()// удаление устаревшей информации из
таблицы scheduleofcare
    {
        using (FbConnection fbcon = newfbconn_con())
        {
            string sqlstring = String.Format($"delete from scheduleofcare where dateAdd <
dateadd(month, -2, current_date)");

            FbCommand InsertSQL =
               new FbCommand(sqlstring, fbcon);// Добавить аргументы
            try
            {
                fbcon.Open();
                FbTransaction fbt = fbcon.BeginTransaction();
                InsertSQL.Transaction = fbt;
                int res = InsertSQL.ExecuteNonQuery();
                fbt.Commit();
            }
            catch (Exception ex)
            {
                Console.WriteLine($"Ошибка при удалении 5 : {ex.Message}");
                if (ex.Message == "Error reading data from the connection.")
                {
                    FbConnection.ClearPool(fbcon);
                }
            }
            finally
            {
                InsertSQL.Dispose();
                fbcon.Close();
            }
        }
    }

    public void deleteOldInformationTableScheduleofmessages()// удаление устаревшей
информации из таблицы scheduleofmessages
    {
        using (FbConnection fbcon = newfbconn_con())
        {
            string sqlstring = String.Format($"delete from scheduleofmessages where dateAdd <
dateadd(month, -2, current_date)");

            FbCommand InsertSQL =
               new FbCommand(sqlstring, fbcon);// Добавить аргументы
            try
            {
                fbcon.Open();
                FbTransaction fbt = fbcon.BeginTransaction();
                InsertSQL.Transaction = fbt;
                int res = InsertSQL.ExecuteNonQuery();
                fbt.Commit();
            }
            catch (Exception ex)
            {
                Console.WriteLine($"Ошибка при удалении 6 : {ex.Message}");
                if (ex.Message == "Error reading data from the connection.")
                {
```

```
                    FbConnection.ClearPool(fbcon);
                }
            }
            finally
            {
                InsertSQL.Dispose();
                fbcon.Close();
            }
        }
    }

    public void deleteOldInformationTableSmssend()// удаление устаревшей информации из
таблицы smssend
    {
        using (FbConnection fbcon = newfbconn_con())
        {
            string sqlstring = String.Format($"delete from smssend where dateAdd <
dateadd(month, -2, current_date)");

            FbCommand InsertSQL =
               new FbCommand(sqlstring, fbcon);// Добавить аргументы
            try
            {
                fbcon.Open();
                FbTransaction fbt = fbcon.BeginTransaction();
                InsertSQL.Transaction = fbt;
                int res = InsertSQL.ExecuteNonQuery();
                fbt.Commit();
            }
            catch (Exception ex)
            {
                Console.WriteLine($"Ошибка при удалении 7 : {ex.Message}");
                if (ex.Message == "Error reading data from the connection.")
                {
                    FbConnection.ClearPool(fbcon);
                }
            }
            finally
            {
                InsertSQL.Dispose();
                fbcon.Close();
            }
        }
    }

    public void deleteOldInformationTableTeststatussendsms()// удаление устаревшей информации
из таблицы teststatussendsms
    {
        using (FbConnection fbcon = newfbconn_con())
        {
            string sqlstring = String.Format($"delete from teststatussendsms where timeTest <
dateadd(month, -2, current_date)");

            FbCommand InsertSQL =
               new FbCommand(sqlstring, fbcon);// Добавить аргументы
            try
            {
                fbcon.Open();
                FbTransaction fbt = fbcon.BeginTransaction();
                InsertSQL.Transaction = fbt;
                int res = InsertSQL.ExecuteNonQuery();
                fbt.Commit();
            }
            catch (Exception ex)
            {
                Console.WriteLine($"Ошибка при удалении 8 : {ex.Message}");
                if (ex.Message == "Error reading data from the connection.")
```

```
                {
                    FbConnection.ClearPool(fbcon);
                }
            }
            finally
            {
                InsertSQL.Dispose();
                fbcon.Close();
            }
        }
    }

    public void deleteOldInformationTableTheconfirmationtable()// удаление устаревшей
информации из таблицы theconfirmationtable
    {
        using (FbConnection fbcon = newfbconn_con())
        {
            string sqlstring = String.Format($"delete from theconfirmationtable where date <
dateadd(month, -2, current_date)");

            FbCommand InsertSQL =
               new FbCommand(sqlstring, fbcon);// Добавить аргументы
            try
            {
                fbcon.Open();
                FbTransaction fbt = fbcon.BeginTransaction();
                InsertSQL.Transaction = fbt;
                int res = InsertSQL.ExecuteNonQuery();
                fbt.Commit();
            }
            catch (Exception ex)
            {
                Console.WriteLine($"Ошибка при удалении 9 : {ex.Message}");
                if (ex.Message == "Error reading data from the connection.")
                {
                    FbConnection.ClearPool(fbcon);
                }
            }
            finally
            {
                InsertSQL.Dispose();
                fbcon.Close();
            }
        }
    }

    public int InsertInScheduleOfMessages(TemplateRecordShedule records) // Вставляю запись в
таблицу для отправки сообщений
    {
        using (FbConnection fbcon = newfbconn_con())
        {
            int timeTreat = (records.fhour - records.bhour) * 60 + records.fmin -
records.bmin;
            if ((timeTreat > 5)
                && (records.IdClient != -1) && (records.refusesms != 1) && (records.nazndis
!= 1)
                && (records.statusClient != 990146168 /*умерший*/))
            {
                if (records.phone != null)
                {
                    if (records.phone.ToString() != "-1")
                    {
                        DateTime dateofreceipt;
                        TimeSpan time = new TimeSpan(records.bhour, records.bmin, 0);
                        dateofreceipt = records.workDate.Date + time;

                        DateTime timeSendMessage2 = dateofreceipt.AddDays(-1);
```

```
                            DateTime timeSendMessage3 = dateofreceipt.AddHours(-3);
                            DateTime timeSendMessage3f;
                            TimeSpan timeSend3f = new TimeSpan(7, 30, 0);
                            if (dateofreceipt.Hour < 11) timeSendMessage3f =
timeSendMessage3.Date + timeSend3f;
                            else timeSendMessage3f = timeSendMessage3;
                            int statusSendMessage2 = 0;
                            int statusSendMessage3 = 0;

                            if (DateTime.Now.Date == timeSendMessage2.Date) statusSendMessage2 =
2;
                            if (records.workDate.Date == DateTime.Now.Date)
                            {
                                int r = (dateofreceipt.Hour * 60 + dateofreceipt.Minute) -
(DateTime.Now.Hour * 60 + DateTime.Now.Minute);
                                if (r >= 240)
                                {
                                    statusSendMessage2 = 3; // 3 - прием в день назначения
                                }
                                else
                                {
                                    statusSendMessage2 = 3; // 3 - прием в день назначения
                                    statusSendMessage3 = 3; // 3 - прием в день назначения
                                }
                            }

                            string sqlstring = String.Format("INSERT INTO
scheduleofmessages(idInfodent,idclient,idfilial,dateofreceipt,timeSendMessage2,timeSendMessage3,w
asItSent1" +
                                   ",wasItSent2,wasItSent3,idDoctor,dateAdd) " +

$"VALUES('{records.IdRecordInfodent}','{records.IdClient}','{records.filial}','{dateofreceipt.ToS
tring("yyyy-MM-dd HH:mm:ss")}','{timeSendMessage2.ToString("yyyy-MM-dd HH:mm:ss")}'," +
                                       $"'{timeSendMessage3f.ToString("yyyy-MM-dd
HH:mm:ss")}','0','{statusSendMessage2}','{statusSendMessage3}','{records.IdDoctor}','{records.dat
eAdd.ToString("yyyy-MM-dd HH:mm:ss")}')");
                            FbCommand InsertSQL = new FbCommand(sqlstring, fbcon);// Добавить
аргументы
                            try
                            {
                                fbcon.Open();
                                FbTransaction fbt = fbcon.BeginTransaction();
                                InsertSQL.Transaction = fbt;
                                int res = InsertSQL.ExecuteNonQuery();
                                fbt.Commit();
                            }
                            catch (Exception ex)
                            {
                                Console.WriteLine($"46 : {ex.Message}");
                                if (ex.Message == "Error reading data from the connection.")
                                {
                                    FbConnection.ClearPool(fbcon);
                                }
                            }
                            finally
                            {
                                InsertSQL.Dispose();
                                fbcon.Close();
                            }
                        }
                    }
                }
                return 1;
            }
        }
```

```
        public int InsertInScheduleOfCare(TemplateRecordShedule records) // Вставляю в таблицу с
дубликатами полей
        {
            using (FbConnection fbcon = newfbconn_con())
            {
                DateTime dateofreceipt;
                TimeSpan time = new TimeSpan(records.bhour, records.bmin, 0);
                dateofreceipt = records.workDate.Date + time;

                string sqlstring = String.Format("INSERT INTO
scheduleofcare(idClient,idInfodent,idDoctor,dateofreceipt,treatcode,primaryOrSecondary,clvisit" +
                       ",idFilial, phone, statusClient, refusesms, nazndis,dateAdd) " +

$"VALUES('{records.IdClient}','{records.IdRecordInfodent}','{records.IdDoctor}','{dateofreceipt.T
oString("yyyy-MM-dd HH:mm:ss")}','{records.treatcode}'," +

$"'{records.status}','{records.clvisit}','{records.filial}','{records.phone}','{records.statusCli
ent}','{records.refusesms}','{records.nazndis}','{records.dateAdd.ToString("yyyy-MM-dd
HH:mm:ss")}')");
                FbCommand InsertSQL = new FbCommand(sqlstring, fbcon);// Добавить аргументы
                try
                {
                    fbcon.Open();
                    FbTransaction fbt = fbcon.BeginTransaction();
                    InsertSQL.Transaction = fbt;
                    int res = InsertSQL.ExecuteNonQuery();
                    fbt.Commit();
                }
                catch (Exception ex)
                {
                    Console.WriteLine($"47 : {ex.Message}");
                    if (ex.Message == "Error reading data from the connection.")
                    {
                        FbConnection.ClearPool(fbcon);
                    }
                }
                finally
                {
                    InsertSQL.Dispose();
                    fbcon.Close();
                }
                return 1;
            }
        }

        public void UpdateRecordsInVoronkaSchedule(long schedid) // Установка статуса
обработанной записи в Voronka_Schedule
        {
            using (FbConnection fbco = newfbcon_co())
            {

                string sqlstring = String.Format($"UPDATE voronka_schedule set processed = '1'
where schedid = '{schedid}'");
                FbCommand InsertSQL =
                   new FbCommand(sqlstring, fbco);// Добавить аргументы
                try
                {
                    fbco.Open();
                    FbTransaction fbt = fbco.BeginTransaction();
                    InsertSQL.Transaction = fbt;
                    int res = InsertSQL.ExecuteNonQuery();
                    fbt.Commit();
                }
                catch (Exception ex)
                {
                    Console.WriteLine($"48 : {ex.Message}");
                    if (ex.Message == "Error reading data from the connection.")
```

```
                {
                    FbConnection.ClearPool(fbco);
                }
            }
            finally
            {
                InsertSQL.Dispose();
                fbco.Close();
            }
        }
    }

    public List<long> NewRecordsInScheduleDbInf(DateTime DateAndTimeStart)// Получаю список
новых необработанных записей в графике
    {
        using (FbConnection fbco = newfbcon_co())
        {
            List<long> schedids = new List<long>();
            FbCommand SelectSQL = new FbCommand(String.Format($"select schedid,
DATEANDTIMEINSERT from voronka_schedule " +
                $"where processed = '0' and dateandtimeinsert >
'{DateAndTimeStart.ToString()}'"), fbco);

            try
            {
                fbco.Open();
                FbTransaction fbt = fbco.BeginTransaction();
                SelectSQL.Transaction = fbt;
                FbDataReader reader = SelectSQL.ExecuteReader();

                try
                {
                    while (reader.Read())
                    {
                        DateTime timeadd = reader.GetDateTime(1);
                        if (DateTime.Now > timeadd)
                        {
                            if (DateTime.Now > timeadd.AddMinutes(2))
                            {
                                schedids.Add(reader.GetInt64(0));
                            }
                        }
                    }
                }
                catch (Exception ex)
                {
                    Console.WriteLine($"IP2 : {ex.Message}");
                    if (ex.Message == "Error reading data from the connection.")
                    {
                        FbConnection.ClearPool(fbco);
                    }
                }
                finally
                {
                    reader.Close();
                }

            }
            catch (Exception ex)
            {
                Console.WriteLine($"49 : {ex.Message}");
                if (ex.Message == "Error reading data from the connection.")
                {
                    FbConnection.ClearPool(fbco);
                }
            }
            Finally
```

```
                {
                    fbco.Close();
                }
                return schedids;
            }
        }

        public TemplateRecordShedule SelectRecordsInInfodent(long schedid) // Выггружаем поля из
Infodent для записи в графике
        {
            using (FbConnection fbco = newfbcon_co())
            {
                TemplateRecordShedule record = new TemplateRecordShedule();
                FbCommand SelectSQL = new FbCommand(String.Format("select pcode, dcode, workdate,
bhour, bmin, fhour, fmin, treatcode, clvisit, status, filial from schedule where schedid =
'{0}'", schedid), fbco);

                try
                {
                    fbco.Open();
                    FbTransaction fbt = fbco.BeginTransaction();
                    SelectSQL.Transaction = fbt;
                    FbDataReader reader = SelectSQL.ExecuteReader();

                    try
                    {
                        while (reader.Read())
                        {
                            record.IdRecordInfodent = schedid;
                            if (reader[0].ToString() != "") record.IdClient = reader.GetInt64(0);
else record.IdClient = -1;
                            record.IdDoctor = reader.GetInt64(1);
                            record.workDate = reader.GetDateTime(2);
                            record.bhour = reader.GetInt16(3);
                            record.bmin = reader.GetInt16(4);
                            record.fhour = reader.GetInt16(5);
                            record.fmin = reader.GetInt16(6);
                            if (reader[7].ToString() != "") record.treatcode =
reader.GetInt64(7);
                            if (reader[8].ToString() != "") record.clvisit = reader.GetInt16(8);
else record.clvisit = -1;
                            if (reader[9].ToString() != "") record.status = reader.GetInt64(9);
                            record.filial = reader.GetInt64(10);
                            record.dateAdd = DateTime.Now;
                        }
                    }
                    catch (Exception ex)
                    {
                        Console.WriteLine($"IP3 : {ex.Message}");
                        if (ex.Message == "Error reading data from the connection.")
                        {
                            FbConnection.ClearPool(fbco);
                        }
                    }
                    finally
                    {
                        reader.Close();
                    }
                }
                catch (Exception ex)
                {
                    Console.WriteLine($"50 : {ex.Message}");
                    if (ex.Message == "Error reading data from the connection.")
                    {
                        FbConnection.ClearPool(fbco);
                    }
                }
```

```
                finally
                {
                    fbco.Close();
                }

                record = SelectPhoneAndStatusAndOtherClientInInfodent(record);
                return record;
            }
        }

        public string UpdateDateIncorporation()  // Обновляю дату старта InputProcessing
        {
            using (FbConnection fbcon = newfbconn_con())
            {
                string now = DateTime.Now.ToString("yyyy-MM-dd HH:mm:ss");
                string sqlstring = String.Format("UPDATE configurationcare SET
DateOfIncorporation='" + now + "' WHERE idConfiguration='1';");
                FbCommand InsertSQL =
                   new FbCommand(sqlstring, fbcon);// Добавить аргументы
                try
                {
                    fbcon.Open();
                    FbTransaction fbt = fbcon.BeginTransaction();
                    InsertSQL.Transaction = fbt;
                    int res = InsertSQL.ExecuteNonQuery();
                    fbt.Commit();
                }
                catch (Exception ex)
                {
                    Console.WriteLine($"51 : {ex.Message}");
                    if (ex.Message == "Error reading data from the connection.")
                    {
                        FbConnection.ClearPool(fbcon);
                    }
                }
                finally
                {
                    InsertSQL.Dispose();
                    fbcon.Close();
                }
                return now;
            }
        }

        public int StatusPowerInputProcessing() // Получаю статус запуска InputProcessing
        {
            using (FbConnection fbcon = newfbconn_con())
            {
                int status = 0;
                FbCommand SelectSQL = new FbCommand(String.Format("SELECT PowerInputProcessing
FROM configurationcare WHERE idConfiguration = 1"));
                try
                {
                    fbcon.Open();
                    FbTransaction fbt = fbcon.BeginTransaction();
                    SelectSQL.Transaction = fbt;
                    FbDataReader reader = SelectSQL.ExecuteReader();
                    try
                    {
                        while (reader.Read())
                        {
                            status = reader.GetInt16(0);
                        }
                    }
                    catch (Exception ex)
                    {
                        Console.WriteLine($"IP3 : {ex.Message}");
```

```
                        if (ex.Message == "Error reading data from the connection.")
                        {
                            FbConnection.ClearPool(fbcon);
                        }
                    }
                    finally
                    {
                        reader.Close();
                    }
                }
                catch (Exception ex)
                {
                    Console.WriteLine($"52 : {ex.Message}");
                    if (ex.Message == "Error reading data from the connection.")
                    {
                        FbConnection.ClearPool(fbcon);
                    }
                }
                finally
                {
                    fbcon.Close();
                }

                return status;
            }
        }

        public TemplateRecordShedule
SelectPhoneAndStatusAndOtherClientInInfodent(TemplateRecordShedule record) // Выггружаем номер и
статус клиента
        {
            using (FbConnection fbco = newfbcon_co())
            {
                FbCommand SelectSQL = new FbCommand(String.Format("select phone1,phone2,phone3,
cstatus, refusesms, nazndis from clients where pcode = '{0}'", record.IdClient), fbco);

                try
                {
                    fbco.Open();
                    FbTransaction fbt = fbco.BeginTransaction();
                    SelectSQL.Transaction = fbt;
                    FbDataReader reader = SelectSQL.ExecuteReader();

                    try
                    {
                        while (reader.Read())
                        {
                            record.phone = reader.GetString(2);
                            if (record.phone.Length != 11 || (record.phone[1] == '3' &&
record.phone[2] == '5' && record.phone[3] == '1'))
                            {
                                record.phone = reader.GetString(1);
                                if (record.phone.Length != 11 || (record.phone[1] == '3' &&
record.phone[2] == '5' && record.phone[3] == '1'))
                                {
                                    record.phone = reader.GetString(0);
                                    if (record.phone.Length != 11 || (record.phone[1] == '3' &&
record.phone[2] == '5' && record.phone[3] == '1'))
                                    {
                                        record.phone = "-1";
                                    }
                                }
                            }

                            if (reader[3].ToString() != "") record.statusClient =
reader.GetInt64(3);
```

```
                        if (reader[4].ToString() != "") record.refusesms =
reader.GetInt16(4);
                        if (reader[5].ToString() != "") record.nazndis = reader.GetInt16(5);
                    }
                }
                catch (Exception ex)
                {
                    Console.WriteLine($"IP4 : {ex.Message}");
                    if (ex.Message == "Error reading data from the connection.")
                    {
                        FbConnection.ClearPool(fbco);
                    }
                }
                finally
                {
                    reader.Close();
                }
            }
            catch (Exception ex)
            {
                Console.WriteLine($"53 : {ex.Message}");
                if (ex.Message == "Error reading data from the connection.")
                {
                    FbConnection.ClearPool(fbco);
                }
            }
            finally
            {
                fbco.Close();
            }

            return record;
        }
    }

    public List<long> NewRecordsInVoronkaScheduleUpdate() // Получаю список обновленных
необработанных записей в графике
    {
        using (FbConnection fbco = newfbcon_co())
        {
            List<long> schedids = new List<long>();
            FbCommand SelectSQL = new FbCommand(String.Format("select schedid from
voronka_schedule_update where processed = '0'"), fbco);

            try
            {
                fbco.Open();
                FbTransaction fbt = fbco.BeginTransaction();
                SelectSQL.Transaction = fbt;
                FbDataReader reader = SelectSQL.ExecuteReader();

                try
                {
                    while (reader.Read())
                    {
                        schedids.Add(reader.GetInt64(0));
                    }
                }
                catch (Exception ex)
                {
                    Console.WriteLine($"IP5 : {ex.Message}");
                    if (ex.Message == "Error reading data from the connection.")
                    {
                        FbConnection.ClearPool(fbco);
                    }
                }
                Finally
```

```
                {
                    reader.Close();
                }
            }
            catch (Exception ex)
            {
                Console.WriteLine($"54 : {ex.Message}");
                if (ex.Message == "Error reading data from the connection.")
                {
                    FbConnection.ClearPool(fbco);
                }
            }
            finally
            {
                fbco.Close();
            }

            return schedids;
        }
    }

    public int UpdateInScheduleOfCare(TemplateRecordShedule records)  // Обновляю запись в
таблице ScheduleOfCare
    {
        using (FbConnection fbcon = newfbconn_con())
        {
            DateTime dateofreceipt;
            TimeSpan time = new TimeSpan(records.bhour, records.bmin, 0);
            dateofreceipt = records.workDate.Date + time;
            string sqlstring = String.Format($"UPDATE scheduleofcare SET dateOfReceipt =
'{dateofreceipt.ToString("yyyy-MM-dd HH:mm:ss")}', " +
                    $"treatcode = '{records.treatcode}', clvisit = '{records.clvisit}' WHERE
idInfodent='{records.IdRecordInfodent}' and obsolete = '0';");
            FbCommand InsertSQL =
               new FbCommand(sqlstring, fbcon);// Добавить аргументы
            try
            {
                fbcon.Open();
                FbTransaction fbt = fbcon.BeginTransaction();
                InsertSQL.Transaction = fbt;
                int res = InsertSQL.ExecuteNonQuery();
                fbt.Commit();
            }
            catch (Exception ex)
            {
                Console.WriteLine($"51 : {ex.Message}");
                if (ex.Message == "Error reading data from the connection.")
                {
                    FbConnection.ClearPool(fbcon);
                }
            }
            finally
            {
                InsertSQL.Dispose();
                fbcon.Close();
            }
            return 1;
        }
    }

    public int UpdateInScheduleOfMessages(TemplateRecordShedule records)  // Обновляю запись
в таблице ScheduleOfMessages
    {
        using (FbConnection fbcon = newfbconn_con())
        {
            DateTime dateofreceipt;
            TimeSpan time = new TimeSpan(records.bhour, records.bmin, 0);
```

```
                dateofreceipt = records.workDate.Date + time;

                StatusSmsModel model = SelectStatusSms(records.IdRecordInfodent);

                if (CheckedRecordToTableOfMessages(records.IdRecordInfodent) == 1)
                {

                    if ((records.fhour - records.bhour == 0) && (records.fmin - records.bmin <=
5))
                    {
                        if (model.wasItSent2 != 1 && model.wasItSent2 != 2 && model.wasItSent2 !=
3 && model.wasItSent2 != 4) model.wasItSent2 = -1;
                        if (model.wasItSent3 != 1 && model.wasItSent3 != 2 && model.wasItSent3 !=
3 && model.wasItSent3 != 4) model.wasItSent3 = -1;
                        string sqlstring = String.Format($"UPDATE scheduleofmessages SET
dateOfReceipt = '{dateofreceipt.ToString("yyyy-MM-dd HH:mm:ss")}', " +
                                $"wasItSent2 = '{model.wasItSent2}', wasItSent3 =
'{model.wasItSent3}' WHERE idInfodent='{records.IdRecordInfodent}' and obsolete = '0';");
                        FbCommand InsertSQL =
                           new FbCommand(sqlstring, fbcon);// Добавить аргументы
                        try
                        {
                            fbcon.Open();
                            FbTransaction fbt = fbcon.BeginTransaction();
                            InsertSQL.Transaction = fbt;
                            int res = InsertSQL.ExecuteNonQuery();
                            fbt.Commit();
                        }
                        catch (Exception ex)
                        {
                            Console.WriteLine($"56 : {ex.Message}");
                            if (ex.Message == "Error reading data from the connection.")
                            {
                                FbConnection.ClearPool(fbcon);
                            }
                        }
                        finally
                        {
                            InsertSQL.Dispose();
                            fbcon.Close();
                        }
                    }
                }
                else
                {
                    if (CheckedRecordToTableOfSchedule(records.IdRecordInfodent) == 1)
InsertInScheduleOfMessages(records);
                }
                return 1;
            }
        }

        public int CheckedRecordToTableOfMessages(long schedid) // Проверка существования записи
в таблице с сообщениями
        {
            using (FbConnection fbcon = newfbconn_con())
            {
                int test = 0;
                FbCommand SelectSQL = new FbCommand(String.Format($"SELECT * FROM
scheduleofmessages WHERE idInfodent = '{schedid}' and obsolete = '0'"), fbcon);
                try
                {
                    fbcon.Open();
                    FbTransaction fbt = fbcon.BeginTransaction();
                    SelectSQL.Transaction = fbt;
                    FbDataReader reader = SelectSQL.ExecuteReader();
```

```
                try
                {
                    while (reader.Read())
                    {
                        test = 1;
                    }
                }
                catch (Exception ex)
                {
                    Console.WriteLine($"IP6 : {ex.Message}");
                    if (ex.Message == "Error reading data from the connection.")
                    {
                        FbConnection.ClearPool(fbcon);
                    }
                }
                finally
                {
                    reader.Close();
                }
            }
            catch (Exception ex)
            {
                Console.WriteLine($"57 : {ex.Message}");
                if (ex.Message == "Error reading data from the connection.")
                {
                    FbConnection.ClearPool(fbcon);
                }
            }
            finally
            {
                fbcon.Close();
            }
            return test;
        }
    }

    public int CheckedRecordToTableOfSchedule(long schedid) // Проверка существования записи
в таблице с записями в графике
    {
        using (FbConnection fbcon = newfbconn_con())
        {
            int test = 0;
            FbCommand SelectSQL = new FbCommand(String.Format($"SELECT * FROM scheduleofcare
WHERE idInfodent = '{schedid}' and obsolete = '0'"), fbcon);
            try
            {
                fbcon.Open();
                FbTransaction fbt = fbcon.BeginTransaction();
                SelectSQL.Transaction = fbt;
                FbDataReader reader = SelectSQL.ExecuteReader();

                try
                {
                    while (reader.Read())
                    {
                        test = 1;
                    }
                }
                catch (Exception ex)
                {
                    Console.WriteLine($"IP7 : {ex.Message}");
                    if (ex.Message == "Error reading data from the connection.")
                    {
                        FbConnection.ClearPool(fbcon);
                    }
```

```
                }
                Finally
                {
                    reader.Close();
                }
            }
            catch (Exception ex)
            {
                Console.WriteLine($"58 : {ex.Message}");
                if (ex.Message == "Error reading data from the connection.")
                {
                    FbConnection.ClearPool(fbcon);
                }
            }
            finally
            {
                fbcon.Close();
            }

            return test;
        }
    }

    public int SelectStatusTwoSMS(long treatcode) //Достаю статус 2 смс
    {
        using (FbConnection fbcon = newfbconn_con())
        {
            int status = 0;
            FbCommand SelectSQL = new FbCommand(String.Format($"SELECT wasItSent2 FROM
scheduleofmessages WHERE idInfodent = '{treatcode}' and obsolete = '0'"), fbcon);
            try
            {
                fbcon.Open();
                FbTransaction fbt = fbcon.BeginTransaction();
                SelectSQL.Transaction = fbt;
                FbDataReader reader = SelectSQL.ExecuteReader();

                try
                {
                    while (reader.Read())
                    {
                        status = reader.GetInt16(0);
                    }
                }
                catch (Exception ex)
                {
                    Console.WriteLine($"IP8 : {ex.Message}");
                    if (ex.Message == "Error reading data from the connection.")
                    {
                        FbConnection.ClearPool(fbcon);
                    }
                }
                finally
                {
                    reader.Close();
                }
            }
            catch (Exception ex)
            {
                Console.WriteLine($"59 : {ex.Message}");
                if (ex.Message == "Error reading data from the connection.")
                {
                    FbConnection.ClearPool(fbcon);
                }
            }
            Finally
```

```
                {
                    fbcon.Close();
                }

                return status;
            }
        }

        public StatusSmsModel SelectStatusSms(long treatcode) //Достаю статусы отправки СМС по
приему
        {
            using (FbConnection fbcon = newfbconn_con())
            {
                StatusSmsModel model = new StatusSmsModel();
                FbCommand SelectSQL = new FbCommand(String.Format($"SELECT wasItSent2, wasItSent3
FROM scheduleofmessages WHERE idInfodent = '{treatcode}' and obsolete = '0'"));

                try
                {
                    fbcon.Open();
                    FbTransaction fbt = fbcon.BeginTransaction();
                    SelectSQL.Transaction = fbt;
                    FbDataReader reader = SelectSQL.ExecuteReader();

                    try
                    {
                        while (reader.Read())
                        {
                            model.wasItSent2 = reader.GetInt16(0);
                            model.wasItSent3 = reader.GetInt16(1);
                        }
                    }
                    catch (Exception ex)
                    {
                        Console.WriteLine($"IP9 : {ex.Message}");
                        if (ex.Message == "Error reading data from the connection.")
                        {
                            FbConnection.ClearPool(fbcon);
                        }
                    }
                    finally
                    {
                        reader.Close();
                    }
                }
                catch (Exception ex)
                {
                    Console.WriteLine($"60 : {ex.Message}");
                    if (ex.Message == "Error reading data from the connection.")
                    {
                        FbConnection.ClearPool(fbcon);
                    }
                }
                finally
                {
                    fbcon.Close();
                }

                return model;
            }
        }

        public void UpdateRecordsInVoronkaScheduleUpdate(long schedid) // Установка статуса
обработанной записи в Voronka_Schedule_Update
        {
            using (FbConnection fbco = newfbcon_co())
```

```
            {
                string sqlstring = String.Format("UPDATE voronka_schedule_update set processed =
'1' where schedid = '{0}'", schedid);
                FbCommand InsertSQL =
                   new FbCommand(sqlstring, fbco);// Добавить аргументы

                try
                {
                    fbco.Open();
                    FbTransaction fbt = fbco.BeginTransaction();
                    InsertSQL.Transaction = fbt;
                    int res = InsertSQL.ExecuteNonQuery();
                    fbt.Commit();
                }
                catch (Exception ex)
                {
                    Console.WriteLine($"61 : {ex.Message}");
                    if (ex.Message == "Error reading data from the connection.")
                    {
                        FbConnection.ClearPool(fbco);
                    }
                }
                finally
                {
                    InsertSQL.Dispose();
                    fbco.Close();
                }
            }
        }

        public List<long> NewRecordsClientsInDbInf()// Получаю список обновленных необработанных
записей в таблице клиентов
        {
            using (FbConnection fbco = newfbcon_co())
            {
                List<long> pcodes = new List<long>();
                FbCommand SelectSQL = new FbCommand(String.Format("select pcode from
VORONKA_UPDATE_CLIENT where processed = '0'"), fbco);

                try
                {
                    fbco.Open();
                    FbTransaction fbt = fbco.BeginTransaction();
                    SelectSQL.Transaction = fbt;
                    FbDataReader reader = SelectSQL.ExecuteReader();

                    try
                    {
                        while (reader.Read())
                        {
                            pcodes.Add(reader.GetInt64(0));
                        }
                    }
                    catch (Exception ex)
                    {
                        Console.WriteLine($"IP10 : {ex.Message}");
                        if (ex.Message == "Error reading data from the connection.")
                        {
                            FbConnection.ClearPool(fbco);
                        }
                    }
                    finally
                    {
                        reader.Close();
                    }
                }
                catch (Exception ex)
```

```
                {

                    Console.WriteLine($"62 : {ex.Message}");
                    if (ex.Message == "Error reading data from the connection.")
                    {
                        FbConnection.ClearPool(fbco);
                    }
                }
                finally
                {
                    fbco.Close();
                }

                return pcodes;
            }
        }

        public TemplateRecordClient SelectRecordClient(long pcode) // Получаю необходимые поля из
таблицы клиентов
        {
            using (FbConnection fbco = newfbcon_co())
            {
                TemplateRecordClient record = new TemplateRecordClient();
                FbCommand SelectSQL = new FbCommand(String.Format("select phone1,phone2,phone3,
cstatus, refusesms, nazndis from clients where pcode = '{0}'", pcode), fbco);

                try
                {
                    fbco.Open();
                    FbTransaction fbt = fbco.BeginTransaction();
                    SelectSQL.Transaction = fbt;
                    FbDataReader reader = SelectSQL.ExecuteReader();

                    try
                    {
                        while (reader.Read())
                        {
                            record.IdClient = pcode;
                            record.phone = reader.GetString(2);
                            if (record.phone.Length != 11 || (record.phone[1] == '3' &&
record.phone[2] == '5' && record.phone[3] == '1'))
                            {
                                record.phone = reader.GetString(1);
                                if (record.phone.Length != 11 || (record.phone[1] == '3' &&
record.phone[2] == '5' && record.phone[3] == '1'))
                                {
                                    record.phone = reader.GetString(0);
                                    if (record.phone.Length != 11 || (record.phone[1] == '3' &&
record.phone[2] == '5' && record.phone[3] == '1'))
                                    {
                                        record.phone = "-1";
                                    }
                                }
                            }
                            if (reader[3].ToString() != "") record.statusClient =
reader.GetInt64(3);
                            if (reader[4].ToString() != "") record.refusesms =
reader.GetInt16(4);
                            if (reader[5].ToString() != "") record.nazndis = reader.GetInt16(5);
                        }
                    }
                    catch (Exception ex)
                    {
                        Console.WriteLine($"IP11 : {ex.Message}");
                        if (ex.Message == "Error reading data from the connection.")
                        {
                            FbConnection.ClearPool(fbco);
```

```
                    }

                }
                finally
                {
                    reader.Close();
                }
            }
            catch (Exception ex)
            {
                Console.WriteLine($"63 : {ex.Message}");
                if (ex.Message == "Error reading data from the connection.")
                {
                    FbConnection.ClearPool(fbco);
                }
            }
            finally
            {
                fbco.Close();
            }

            return record;
        }
    }
    public void UpdateRecordsInVoronkaUpdateClient(long pcode) // Установка статуса
обработанной записи в Voronka_Update_Client
    {
        using (FbConnection fbco = newfbcon_co())
        {
            string sqlstring = String.Format("UPDATE voronka_update_client set processed =
'1' where pcode = '{0}'", pcode);
            FbCommand InsertSQL =
               new FbCommand(sqlstring, fbco);// Добавить аргументы

            try
            {
                fbco.Open();
                FbTransaction fbt = fbco.BeginTransaction();
                InsertSQL.Transaction = fbt;

                int res = InsertSQL.ExecuteNonQuery();
                fbt.Commit();
            }
            catch (Exception ex)
            {
                Console.WriteLine($"64 : {ex.Message}");
                if (ex.Message == "Error reading data from the connection.")
                {
                    FbConnection.ClearPool(fbco);
                }
            }
            finally
            {
                InsertSQL.Dispose();
                fbco.Close();
            }
        }
    }

    public int UpdateInScheduleOfCareClientColumns(TemplateRecordClient records)  // Обновляю
запись в таблице ScheduleOfCare
    {
        using (FbConnection fbcon = newfbconn_con())
        {
            string sqlstring = String.Format($"UPDATE scheduleofcare SET phone =
'{records.phone}', " +
```

```
                    $"statusClient = '{records.statusClient}', refusesms =
'{records.refusesms}', nazndis = '{records.nazndis}' WHERE " +

                    $"dateOfReceipt >= '{DateTime.Now.ToString("yyyy-MM-dd HH:mm:ss")}' and
idClient = '{records.IdClient}' " +
                    $"and obsolete = '0';");
                FbCommand InsertSQL =
                   new FbCommand(sqlstring, fbcon);// Добавить аргументы

                try
                {
                    fbcon.Open();
                    FbTransaction fbt = fbcon.BeginTransaction();
                    InsertSQL.Transaction = fbt;

                    int res = InsertSQL.ExecuteNonQuery();
                    fbt.Commit();
                }
                catch (Exception ex)
                {
                    Console.WriteLine($"65 : {ex.Message}");
                    if (ex.Message == "Error reading data from the connection.")
                    {
                        FbConnection.ClearPool(fbcon);
                    }
                }
                finally
                {
                    InsertSQL.Dispose();
                    fbcon.Close();
                }
                return 1;
            }
        }

        public int UpdateInScheduleOfMessagesClient(TemplateRecordClient records)  // Обновляю
запись в таблице ScheduleOfMessages
        {
            using (FbConnection fbcon = newfbconn_con())
            {
                if ((records.refusesms == 1) || (records.nazndis == 1) || (records.statusClient
== 990146168 /*умерший*/))
                {
                    foreach (var record in SelectTreatByClient(records.IdClient))
                    {
                        StatusSmsModel model = SelectStatusSms(record);
                        if (model.wasItSent2 != 1 && model.wasItSent2 != 2 && model.wasItSent2 !=
3 && model.wasItSent2 != 4) model.wasItSent2 = -1;
                        if (model.wasItSent3 != 1 && model.wasItSent3 != 2 && model.wasItSent3 !=
3 && model.wasItSent3 != 4) model.wasItSent3 = -1;
                        string sqlstring = String.Format($"UPDATE scheduleofmessages SET
wasItSent2 = '{model.wasItSent2}', wasItSent3 = '{model.wasItSent3}'" +
                                $" WHERE dateOfReceipt >= '{DateTime.Now.ToString("yyyy-MM-dd
HH:mm:ss")}' and idClient = '{records.IdClient}' " +
                                $"and obsolete = '0';");
                        FbCommand InsertSQL =
                           new FbCommand(sqlstring, fbcon);// Добавить аргументы

                        try
                        {
                            fbcon.Open();
                            FbTransaction fbt = fbcon.BeginTransaction();
                            InsertSQL.Transaction = fbt;

                            int res = InsertSQL.ExecuteNonQuery();
                            fbt.Commit();
```

```
                        }
                        catch (Exception ex)
                        {

                            Console.WriteLine($"66 : {ex.Message}");
                            if (ex.Message == "Error reading data from the connection.")
                            {
                                FbConnection.ClearPool(fbcon);
                            }
                        }
                        finally
                        {
                            InsertSQL.Dispose();
                            fbcon.Close();
                        }
                    }
                }
                else
                {
                    foreach (var record in SelectTreatByClientScheduleofcare(records.IdClient))
                    {
                        if (CheckedRecordToTableOfMessages(record) == 0)
                        {
                            TemplateRecordShedule tempRecord = SelectRecordsInInfodent(record);
                            InsertInScheduleOfMessages(tempRecord);
                        }
                        else
                        {
                            StatusSmsModel model = SelectStatusSms(record);
                            if (model.wasItSent2 == -1) model.wasItSent2 = 0;
                            if (model.wasItSent3 == -1) model.wasItSent3 = 0;
                            string sqlstring = String.Format($"UPDATE scheduleofmessages SET
wasItSent2 = '{model.wasItSent2}', wasItSent3 = '{model.wasItSent3}' WHERE idInfodent =
'{record}' " +
                                   $"and obsolete = '0';");
                            FbCommand InsertSQL =
                           new FbCommand(sqlstring, fbcon);// Добавить аргументы

                            try
                            {
                                fbcon.Open();
                                FbTransaction fbt = fbcon.BeginTransaction();
                                InsertSQL.Transaction = fbt;

                                int res = InsertSQL.ExecuteNonQuery();
                                fbt.Commit();
                            }
                            catch (Exception ex)
                            {
                                Console.WriteLine($"67 : {ex.Message}");
                                if (ex.Message == "Error reading data from the connection.")
                                {
                                    FbConnection.ClearPool(fbcon);
                                }
                            }
                            finally
                            {
                                InsertSQL.Dispose();
                                fbcon.Close();
                            }
                        }
                    }
                }
                return 1;
            }
        }
```

```
        public List<long> SelectTreatByClient(long pcode)
        {
            using (FbConnection fbcon = newfbconn_con())


            {
                List<long> idIndafents = new List<long>();
                DateTime now = DateTime.Now;
                FbCommand SelectSQL = new FbCommand(String.Format($"SELECT idInfodent FROM
scheduleofmessages WHERE dateOfReceipt > '{now.ToString("yyyy-MM-dd HH:mm:ss")}' and idClient =
'{pcode}' and obsolete = '0'"), fbcon);
                try
                {
                    fbcon.Open();
                    FbTransaction fbt = fbcon.BeginTransaction();
                    SelectSQL.Transaction = fbt;
                    FbDataReader reader = SelectSQL.ExecuteReader();

                    try
                    {
                        while (reader.Read())
                        {
                            idIndafents.Add(reader.GetInt64(0));
                        }
                    }
                    catch (Exception ex)
                    {
                        Console.WriteLine($"IP12 : {ex.Message}");
                        if (ex.Message == "Error reading data from the connection.")
                        {
                            FbConnection.ClearPool(fbcon);
                        }
                    }
                    finally
                    {
                        reader.Close();
                    }
                }
                catch (Exception ex)
                {
                    Console.WriteLine($"68 : {ex.Message}");
                    if (ex.Message == "Error reading data from the connection.")
                    {
                        FbConnection.ClearPool(fbcon);
                    }
                }
                finally
                {
                    fbcon.Close();
                }

                return idIndafents;
            }
        } // выгружаю все записи по клиенту из таблицы scheduleofmessages

        public List<long> SelectTreatByClientScheduleofcare(long pcode)// выгружаю все записи по
клиенту из таблицы scheduleofcare
        {
            using (FbConnection fbcon = newfbconn_con())
            {
                List<long> idIndafents = new List<long>();
                DateTime now = DateTime.Now;
                FbCommand SelectSQL = new FbCommand(String.Format($"SELECT idInfodent FROM
scheduleofcare WHERE dateOfReceipt > '{now.ToString("yyyy-MM-dd HH:mm:ss")}' and idClient =
'{pcode}' and obsolete = '0'"), fbcon);

                Try
```

```
            {
                fbcon.Open();
                FbTransaction fbt = fbcon.BeginTransaction();

                SelectSQL.Transaction = fbt;
                FbDataReader reader = SelectSQL.ExecuteReader();

                try
                {
                    while (reader.Read())
                    {
                        idIndafents.Add(reader.GetInt64(0));
                    }
                }
                catch (Exception ex)
                {
                    Console.WriteLine($"IP13 : {ex.Message}");
                    if (ex.Message == "Error reading data from the connection.")
                    {
                        FbConnection.ClearPool(fbcon);
                    }
                }
                finally
                {
                    reader.Close();
                }
            }
            catch (Exception ex)
            {
                Console.WriteLine($"69 : {ex.Message}");
                if (ex.Message == "Error reading data from the connection.")
                {
                    FbConnection.ClearPool(fbcon);
                }
            }
            finally
            {
                fbcon.Close();
            }

            return idIndafents;
        }
    } // выгружаю все записи по клиенту из таблицы scheduleofmessages

    public Tuple<bool, int> AddStatusConfirmation(short status, long code)
    {
        bool test = false;
        int oldTreat = 0;

        using (FbConnection fbcon = newfbconn_con())
        {
            short nowStatus = selectTheConfirmation(code);
            if (selectTheDateOfReceipt(code) >= DateTime.Now)
            {
                switch (status)
                {
                    case 0: // отказ
                        if (nowStatus != 0)
                        {
                            string sqlstring = String.Format($"UPDATE scheduleofcare SET
theConfirmation = '0' WHERE idInfodent = '{code}' and obsolete = '0';");
                            FbCommand InsertSQL =
                           new FbCommand(sqlstring, fbcon);// Добавить аргументы
                            try
                            {
                                fbcon.Open();
                                FbTransaction fbt = fbcon.BeginTransaction();
```

```
                                    InsertSQL.Transaction = fbt;
                                    int res = InsertSQL.ExecuteNonQuery();
                                    fbt.Commit();

                                    sqlstring = String.Format($"UPDATE scheduleofmessages SET
theConfirmation = '0' WHERE idInfodent = '{code}' and obsolete = '0';");
                                    InsertSQL = new FbCommand(sqlstring, fbcon);// Добавить
аргументы
                                    fbt = fbcon.BeginTransaction();
                                    InsertSQL.Transaction = fbt;
                                    res = InsertSQL.ExecuteNonQuery();
                                    fbt.Commit();
                                }
                                catch (Exception ex)
                                {
                                    Console.WriteLine($"70 : {ex.Message}");
                                    if (ex.Message == "Error reading data from the connection.")
                                    {
                                        FbConnection.ClearPool(fbcon);
                                    }
                                }
                                finally
                                {
                                    InsertSQL.Dispose();
                                    fbcon.Close();
                                }

                                if (test)
                                {
                                    test = UpdateConfirmationInfodentStop(code);
                                }
                            }
                            break;

                        case 1: // подтверждение
                            if (nowStatus != 1 && nowStatus != 0)
                            {
                                string sqlstring = String.Format($"UPDATE scheduleofcare SET
theConfirmation = '1' WHERE idInfodent = '{code}' and obsolete = '0';");
                                FbCommand InsertSQL =
                               new FbCommand(sqlstring, fbcon);// Добавить аргументы
                                try
                                {
                                    fbcon.Open();
                                    FbTransaction fbt = fbcon.BeginTransaction();
                                    InsertSQL.Transaction = fbt;
                                    int res = InsertSQL.ExecuteNonQuery();
                                    fbt.Commit();

                                    sqlstring = String.Format($"UPDATE scheduleofmessages SET
theConfirmation = '1' WHERE idInfodent = '{code}' and obsolete = '0';");
                                    InsertSQL = new FbCommand(sqlstring, fbcon);// Добавить
аргументы
                                    fbt = fbcon.BeginTransaction();
                                    InsertSQL.Transaction = fbt;
                                    res = InsertSQL.ExecuteNonQuery();
                                    fbt.Commit();
                                }
                                catch (Exception ex)
                                {
                                    Console.WriteLine($"71 : {ex.Message}");
                                    if (ex.Message == "Error reading data from the connection.")
                                    {
                                        FbConnection.ClearPool(fbcon);
                                    }
                                }
```

```
                                finally
                                {
                                    InsertSQL.Dispose();

                                    fbcon.Close();
                                }

                                if (test)
                                {
                                    test = UpdateConfirmationInfodent(code);
                                }
                            }
                            break;
                       }
                    }
                    else
                    {
                        test = true;
                        oldTreat = 1;
                    }
                }
                return Tuple.Create(test, oldTreat);
            } // добавить статус подтверждения либо отказа приема в базу Voronka

            public short selectTheConfirmation(long code)
            {
                using (FbConnection fbcon = newfbconn_con())
                {
                    short status = -1;
                    FbCommand SelectSQL = new FbCommand(String.Format($"SELECT theConfirmation FROM
scheduleofcare WHERE idInfodent = '{code}' and obsolete = '0'"), fbcon);

                    try
                    {
                        fbcon.Open();
                        FbTransaction fbt = fbcon.BeginTransaction();
                        SelectSQL.Transaction = fbt;
                        FbDataReader reader = SelectSQL.ExecuteReader();

                        try
                        {
                            while (reader.Read())
                            {
                                if (reader[0].ToString() != "") status = reader.GetInt16(0);
                            }
                        }
                        catch (Exception ex)
                        {
                            Console.WriteLine($"IP14 : {ex.Message}");
                            if (ex.Message == "Error reading data from the connection.")
                            {
                                FbConnection.ClearPool(fbcon);
                            }
                        }
                        finally
                        {
                            reader.Close();
                        }
                    }
                    catch (Exception ex)
                    {
                        Console.WriteLine($"72 : {ex.Message}");
                        if (ex.Message == "Error reading data from the connection.")
                        {
                            FbConnection.ClearPool(fbcon);
                        }
                    }
```

```
                finally
                {
                    fbcon.Close();

                }

                return status;
            }
        } // выборка подтверждения приема

        public DateTime selectTheDateOfReceipt(long code)
        {
            using (FbConnection fbcon = newfbconn_con())
            {
                DateTime dateOfReceipt = DateTime.Now;
                FbCommand SelectSQL = new FbCommand(String.Format($"SELECT dateOfReceipt FROM
scheduleofcare WHERE idInfodent = '{code}' and obsolete = '0'"), fbcon);

                try
                {
                    fbcon.Open();
                    FbTransaction fbt = fbcon.BeginTransaction();
                    SelectSQL.Transaction = fbt;
                    FbDataReader reader = SelectSQL.ExecuteReader();

                    try
                    {
                        while (reader.Read())
                        {
                            if (reader[0].ToString() != "") dateOfReceipt =
reader.GetDateTime(0);
                        }
                    }
                    catch (Exception ex)
                    {
                        Console.WriteLine($"IP15 : {ex.Message}");
                        if (ex.Message == "Error reading data from the connection.")
                        {
                            FbConnection.ClearPool(fbcon);
                        }
                    }
                    finally
                    {
                        reader.Close();
                    }
                }
                catch (Exception ex)
                {
                    Console.WriteLine($"73 : {ex.Message}");
                    if (ex.Message == "Error reading data from the connection.")
                    {
                        FbConnection.ClearPool(fbcon);
                    }
                }
                finally
                {
                    fbcon.Close();
                }

                return dateOfReceipt;
            }
        } // выборка даты приема

        public bool UpdateConfirmationInfodent(long code)
        {
            bool test = false;
```

```
            for (int i = 0; i < 2; i++)
            {
                using (FbConnection fbww001 = newfbww001_con())

                using (FbConnection fbww002 = newfbww002_con())
                using (FbConnection fbww003 = newfbww003_con())
                using (FbConnection fbww005 = newfbww005_con())
                {
                    long idFilial = selectIdFilialRecord(code);
                    string sqlstring = "";
                    if (i == 0)
                    {
                        sqlstring = String.Format($"UPDATE schedule set clcall = '990001026',
calltime = '{DateTime.Now.ToString()}', calltype = '1', status = '{selectStatusRecord(code)}'
where schedid = '{code}'");
                    }
                    else
                    {
                        sqlstring = String.Format($"UPDATE schedule set comment = iif(comment is
null, 'КР3:Подт. пр.', comment || '-' || 'КР3:Подт. пр.') where schedid = '{code}'");
                    }
                    FbTransaction fbt;
                    switch (idFilial)
                    {
                        case 1:
                            FbCommand InsertSQL =
                               new FbCommand(sqlstring, fbww001);// Добавить аргументы

                            try
                            {
                                fbww001.Open();
                                fbt = fbww001.BeginTransaction();
                                InsertSQL.Transaction = fbt;

                                int res = InsertSQL.ExecuteNonQuery();
                                fbt.Commit();

                                test = true;
                            }
                            catch (Exception ex)
                            {
                                Console.WriteLine($"74 : {ex.Message}");
                                if (ex.Message == "Error reading data from the connection.")
                                {
                                    FbConnection.ClearPool(fbww001);
                                }
                            }
                            finally
                            {
                                InsertSQL.Dispose();
                                fbww001.Close();
                            }
                            break;

                        case 2:
                            InsertSQL =
                               new FbCommand(sqlstring, fbww002);// Добавить аргументы

                            try
                            {
                                fbww002.Open();
                                fbt = fbww002.BeginTransaction();
                                InsertSQL.Transaction = fbt;

                                int res = InsertSQL.ExecuteNonQuery();
                                fbt.Commit();
```

```
            test = true;
        }
        catch (Exception ex)

        {
            Console.WriteLine($"75 : {ex.Message}");
            if (ex.Message == "Error reading data from the connection.")
            {
                FbConnection.ClearPool(fbww002);
            }
        }
        finally
        {
            InsertSQL.Dispose();
            fbww002.Close();
        }
        break;

    case 3:
        InsertSQL =
           new FbCommand(sqlstring, fbww003);// Добавить аргументы

        try
        {
            fbww003.Open();
            fbt = fbww003.BeginTransaction();
            InsertSQL.Transaction = fbt;

            int res = InsertSQL.ExecuteNonQuery();
            fbt.Commit();

            test = true;
        }
        catch (Exception ex)
        {
            Console.WriteLine($"76 : {ex.Message}");
            if (ex.Message == "Error reading data from the connection.")
            {
                FbConnection.ClearPool(fbww003);
            }
        }
        finally
        {
            InsertSQL.Dispose();
            fbww003.Close();
        }
        break;

    case 5:
        InsertSQL =
           new FbCommand(sqlstring, fbww005);// Добавить аргументы

        try
        {
            fbww005.Open();
            fbt = fbww005.BeginTransaction();
            InsertSQL.Transaction = fbt;

            int res = InsertSQL.ExecuteNonQuery();
            fbt.Commit();

            test = true;
        }
        catch (Exception ex)
        {
            Console.WriteLine($"77 : {ex.Message}");
```

```
                                if (ex.Message == "Error reading data from the connection.")
                                {
                                    FbConnection.ClearPool(fbww005);

                                }
                            }
                            finally
                            {
                                InsertSQL.Dispose();
                                fbww005.Close();
                            }
                            break;
                        }
                    }
                }

                return test;
            }// добавление статуса подтверждение в Infodent

            public int testExists(long schid)
            {
                using (FbConnection fbco = newfbcon_co())
                {
                    int test = 0;
                    FbCommand SelectSQL = new FbCommand(String.Format($"select * from schedule where
schedid = '{schid}'"), fbco);

                    try
                    {
                        fbco.Open();
                        FbTransaction fbt = fbco.BeginTransaction();
                        SelectSQL.Transaction = fbt;
                        FbDataReader reader = SelectSQL.ExecuteReader();

                        try
                        {
                            while (reader.Read())
                            {
                                test = 1;
                            }
                        }
                        catch (Exception ex)
                        {
                            if (ex.Message == "Error reading data from the connection.")
                            {
                                FbConnection.ClearPool(fbco);
                            }
                        }
                        finally
                        {
                            reader.Close();
                        }
                    }
                    catch (Exception ex)
                    {
                        Console.WriteLine($"78 : {ex.Message}");
                        if (ex.Message == "Error reading data from the connection.")
                        {
                            FbConnection.ClearPool(fbco);
                        }
                    }
                    finally
                    {
                        fbco.Close();
                    }
```

```
                return test;
            }
        }

        public long selectStatusRecord(long schedid)
        {
            using (FbConnection fbcon = newfbconn_con())
            {
                long status = 0;
                FbCommand SelectSQL = new FbCommand(String.Format($"SELECT primaryOrSecondary
FROM scheduleofcare WHERE idInfodent = '{schedid}' and obsolete = '0'"));

                try
                {
                    fbcon.Open();
                    FbTransaction fbt = fbcon.BeginTransaction();
                    SelectSQL.Transaction = fbt;
                    FbDataReader reader = SelectSQL.ExecuteReader();

                    try
                    {
                        while (reader.Read())
                        {
                            if (reader[0].ToString() != "") status = reader.GetInt64(0);
                        }
                    }
                    catch (Exception ex)
                    {
                        Console.WriteLine($"IP17 : {ex.Message}");
                        if (ex.Message == "Error reading data from the connection.")
                        {
                            FbConnection.ClearPool(fbcon);
                        }
                    }
                    finally
                    {
                        reader.Close();
                    }
                }
                catch (Exception ex)
                {
                    Console.WriteLine($"79 : {ex.Message}");
                    if (ex.Message == "Error reading data from the connection.")
                    {
                        FbConnection.ClearPool(fbcon);
                    }
                }
                finally
                {
                    fbcon.Close();
                }

                return status;
            }
        }

        public bool UpdateConfirmationInfodentStop(long code)
        {
            bool test = false;

            for (int i = 0; i < 2; i++)
            {
                using (FbConnection fbww001 = newfbww001_con())
                using (FbConnection fbww002 = newfbww002_con())
                using (FbConnection fbww003 = newfbww003_con())
                using (FbConnection fbww005 = newfbww005_con())
```

```
            {

                long idFilial = selectIdFilialRecord(code);
                string sqlstring = "";

                if (i == 0)
                {
                    sqlstring = String.Format($"UPDATE schedule set clcall = null, calltime =
null, calltype = null, status = '990150313' where schedid = '{code}'");
                }
                else
                {
                    sqlstring = String.Format($"UPDATE schedule set comment = iif(comment is
null, 'КР3:Отк. от пр.', comment || '-' || 'КР3:Отк. от пр.') where schedid = '{code}'");
                }
                FbTransaction fbt;

                switch (idFilial)
                {
                    case 1:
                        FbCommand InsertSQL =
                           new FbCommand(sqlstring, fbww001);// Добавить аргументы

                        try
                        {
                            fbww001.Open();
                            fbt = fbww001.BeginTransaction();
                            InsertSQL.Transaction = fbt;

                            int res = InsertSQL.ExecuteNonQuery();
                            fbt.Commit();

                            test = true;
                        }
                        catch (Exception ex)
                        {
                            Console.WriteLine($"80 : {ex.Message}");
                            if (ex.Message == "Error reading data from the connection.")
                            {
                                FbConnection.ClearPool(fbww001);
                            }
                        }
                        finally
                        {
                            InsertSQL.Dispose();
                            fbww001.Close();
                        }
                        break;

                    case 2:
                        InsertSQL =
                           new FbCommand(sqlstring, fbww002);// Добавить аргументы

                        try
                        {
                            fbww002.Open();
                            fbt = fbww002.BeginTransaction();
                            InsertSQL.Transaction = fbt;

                            int res = InsertSQL.ExecuteNonQuery();
                            fbt.Commit();

                            test = true;
                        }
                        catch (Exception ex)
                        {
```

```
            Console.WriteLine($"81 : {ex.Message}");
            if (ex.Message == "Error reading data from the connection.")
            {
                FbConnection.ClearPool(fbww002);
            }

        }
        finally
        {
            InsertSQL.Dispose();
            fbww002.Close();
        }
        break;

    case 3:
        InsertSQL =
           new FbCommand(sqlstring, fbww003);// Добавить аргументы

        try
        {
            fbww003.Open();
            fbt = fbww003.BeginTransaction();
            InsertSQL.Transaction = fbt;

            int res = InsertSQL.ExecuteNonQuery();
            fbt.Commit();

            test = true;
        }
        catch (Exception ex)
        {
            Console.WriteLine($"82 : {ex.Message}");
            if (ex.Message == "Error reading data from the connection.")
            {
                FbConnection.ClearPool(fbww003);
            }
        }
        finally
        {
            InsertSQL.Dispose();
            fbww003.Close();
        }
        break;

    case 5:
        InsertSQL =
           new FbCommand(sqlstring, fbww005);// Добавить аргументы

        try
        {
            fbww005.Open();
            fbt = fbww005.BeginTransaction();
            InsertSQL.Transaction = fbt;

            int res = InsertSQL.ExecuteNonQuery();
            fbt.Commit();

            test = true;
        }
        catch (Exception ex)
        {
            Console.WriteLine($"83 : {ex.Message}");
            if (ex.Message == "Error reading data from the connection.")
            {
                FbConnection.ClearPool(fbww005);
            }
```

```
                        }
                        finally
                        {
                            InsertSQL.Dispose();
                            fbww005.Close();
                        }

                        break;
                    }
                }
            }

            return test;
        }// Отказ от приема

        public long selectIdFilialRecord(long code)
        {
            using (FbConnection fbco = newfbcon_co())
            {
                long idFilial = 0;
                FbCommand SelectSQL = new FbCommand(String.Format("select filial from schedule
where schedid = '{0}'", code), fbco);

                Try
                {
                    fbco.Open();
                    FbTransaction fbt = fbco.BeginTransaction();
                    SelectSQL.Transaction = fbt;
                    FbDataReader reader = SelectSQL.ExecuteReader();

                    try
                    {
                        while (reader.Read())
                        {
                            idFilial = reader.GetInt64(0);
                        }
                    }
                    catch (Exception ex)
                    {
                        Console.WriteLine($"IP18 : {ex.Message}");
                        if (ex.Message == "Error reading data from the connection.")
                        {
                            FbConnection.ClearPool(fbco);
                        }
                    }
                    finally
                    {
                        reader.Close();
                    }
                }
                catch (Exception ex)
                {
                    Console.WriteLine($"84 : {ex.Message}");
                    if (ex.Message == "Error reading data from the connection.")
                    {
                        FbConnection.ClearPool(fbco);
                    }
                }
                finally
                {
                    fbco.Close();
                }

                return idFilial;
            }
        } // получение idFilial по записи в расписании
```

```
        public short testObsolete(TemplateRecordShedule newRecord)
        {
            short test = 0;

            TemplateTestRecord oldRecord = selectOldMarginsRecord(newRecord.IdRecordInfodent);


            DateTime dateofreceiptNew;
            TimeSpan time = new TimeSpan(newRecord.bhour, newRecord.bmin, 0);
            dateofreceiptNew = newRecord.workDate.Date + time;

            if (oldRecord.dateOfReceipt != dateofreceiptNew || oldRecord.IdDoctor !=
newRecord.IdDoctor) test = 1;
            if (oldRecord.dateOfReceipt == DateTime.Parse("01.01.0001 0:00:00") &&
oldRecord.IdDoctor == 0 && oldRecord.IdRecordInfodent == 0) test = 0;

            return test;
        } // тестирование изменений в записи

        public TemplateTestRecord selectOldMarginsRecord(long idInf)
        {
            using (FbConnection fbcon = newfbconn_con())
            {
                TemplateTestRecord record = new TemplateTestRecord();
                FbCommand SelectSQL = new FbCommand(String.Format($"SELECT idDoctor,
dateOfReceipt FROM scheduleofcare WHERE idInfodent = '{idInf}' and obsolete = '0'"));

                try
                {
                    fbcon.Open();
                    FbTransaction fbt = fbcon.BeginTransaction();
                    SelectSQL.Transaction = fbt;
                    FbDataReader reader = SelectSQL.ExecuteReader();

                    try
                    {
                        while (reader.Read())
                        {
                            record.IdRecordInfodent = idInf;
                            record.IdDoctor = reader.GetInt64(0);
                            record.dateOfReceipt = reader.GetDateTime(1);
                        }
                    }
                    catch (Exception ex)
                    {
                        Console.WriteLine($"IP19 : {ex.Message}");
                        if (ex.Message == "Error reading data from the connection.")
                        {
                            FbConnection.ClearPool(fbcon);
                        }
                    }
                    finally
                    {
                        reader.Close();
                    }
                }
                catch (Exception ex)
                {
                    Console.WriteLine($"82 : {ex.Message}");
                    if (ex.Message == "Error reading data from the connection.")
                    {
                        FbConnection.ClearPool(fbcon);
                    }
                }
                Finally
                {
```

```
                fbcon.Close();
            }

            return record;
        }
    } // достаем нужные поля из старой записи приема


    public int obsoleteRecord(long idInf)
    {
        using (FbConnection fbcon = newfbconn_con())
        {
            string sqlstring = String.Format($"UPDATE scheduleofcare SET obsolete = '1' WHERE
idInfodent = '{idInf}' and obsolete = '0';");
            FbCommand InsertSQL =
           new FbCommand(sqlstring, fbcon);// Добавить аргументы
            try
            {
                fbcon.Open();
                FbTransaction fbt = fbcon.BeginTransaction();
                InsertSQL.Transaction = fbt;
                int res = InsertSQL.ExecuteNonQuery();
                fbt.Commit();

                sqlstring = String.Format($"UPDATE scheduleofmessages SET obsolete = '1'
WHERE idInfodent = '{idInf}' and obsolete = '0';");
                InsertSQL = new FbCommand(sqlstring, fbcon);// Добавить аргументы
                fbt = fbcon.BeginTransaction();
                InsertSQL.Transaction = fbt;
                res = InsertSQL.ExecuteNonQuery();
                fbt.Commit();
            }
            catch (Exception ex)
            {
                Console.WriteLine($"IP20 : {ex.Message}");
                if (ex.Message == "Error reading data from the connection.")
                {
                    FbConnection.ClearPool(fbcon);
                }
            }
            finally
            {
                InsertSQL.Dispose();
                fbcon.Close();
            }

            return 1;
        }
    } // устаревание записи

    public List<long> NewRecordsInVoronkaScheduleDelete() // Получаю список удаленных
необработанных записей в графике
    {
        using (FbConnection fbco = newfbcon_co())
        {
            List<long> schedids = new List<long>();
            FbCommand SelectSQL = new FbCommand(String.Format("select schedid from
voronka_schedule_delete where processed = '0'"), fbco);

            try
            {
                fbco.Open();
                FbTransaction fbt = fbco.BeginTransaction();
                SelectSQL.Transaction = fbt;
                FbDataReader reader = SelectSQL.ExecuteReader();
```

```
                    try
                    {
                        while (reader.Read())
                        {
                            schedids.Add(reader.GetInt64(0));
                        }
                    }

                    catch (Exception ex)
                    {
                        Console.WriteLine($"IP21 : {ex.Message}");
                        if (ex.Message == "Error reading data from the connection.")
                        {
                            FbConnection.ClearPool(fbco);
                        }
                    }
                    finally
                    {
                        reader.Close();
                    }
                }
                catch (Exception ex)
                {
                    Console.WriteLine($"86 : {ex.Message}");
                    if (ex.Message == "Error reading data from the connection.")
                    {
                        FbConnection.ClearPool(fbco);
                    }

                }
                finally
                {
                    fbco.Close();
                }

                return schedids;
            }
        }
        public void UpdateRecordsInVoronkaScheduleDelete(long schedid) // Установка статуса
обработанной записи в Voronka_Update_Client
        {
            using (FbConnection fbco = newfbcon_co())
            {
                string sqlstring = String.Format("UPDATE VORONKA_SCHEDULE_DELETE set processed =
'1' where schedid = '{0}'", schedid);
                FbCommand InsertSQL =
                   new FbCommand(sqlstring, fbco);// Добавить аргументы

                try
                {
                    fbco.Open();
                    FbTransaction fbt = fbco.BeginTransaction();
                    InsertSQL.Transaction = fbt;

                    int res = InsertSQL.ExecuteNonQuery();
                    fbt.Commit();
                }
                catch (Exception ex)
                {
                    Console.WriteLine($"87 : {ex.Message}");
                    if (ex.Message == "Error reading data from the connection.")
                    {
                        FbConnection.ClearPool(fbco);
                    }
                }
                Finally
```

```
                {
                    InsertSQL.Dispose();
                    fbco.Close();
                }
            }
        }

        public void insertTheConfirmation(TemplateTestConfirmation confirmation)

        {
            using (FbConnection fbcon = newfbconn_con())
            {
                string sqlstring = String.Format("INSERT INTO theconfirmationtable(idInfodent,
date, status) " +
                                        $"VALUES('{confirmation.schedid}',
'{confirmation.createAt}','{confirmation.status}')");
                FbCommand InsertSQL = new FbCommand(sqlstring, fbcon);// Добавить аргументы
                try
                {
                    fbcon.Open();
                    FbTransaction fbt = fbcon.BeginTransaction();
                    InsertSQL.Transaction = fbt;
                    int res = InsertSQL.ExecuteNonQuery();
                    fbt.Commit();
                }
                catch (Exception ex)
                {
                    Console.WriteLine($"100 : {ex.Message}");
                    if (ex.Message == "Error reading data from the connection.")

                        FbConnection.ClearPool(fbcon);
                    }
                }
                finally
                {
                    InsertSQL.Dispose();
                    fbcon.Close();
                }
            }
        }

        public bool testExistsTheConfirmation(TemplateTestConfirmation confirmation)
        {
            using (FbConnection fbcon = newfbconn_con())
            {
                bool status = false;
                FbCommand SelectSQL = new FbCommand(String.Format($"SELECT * FROM
theconfirmationtable WHERE idInfodent = '{confirmation.schedid}' and date =
'{confirmation.createAt}' and status = '{confirmation.status}'"));

                try
                {
                    fbcon.Open();
                    FbTransaction fbt = fbcon.BeginTransaction();
                    SelectSQL.Transaction = fbt;
                    FbDataReader reader = SelectSQL.ExecuteReader();

                    try
                    {
                        while (reader.Read())
                        {
                            status = true;
                        }
                    }
                    catch (Exception ex)
                    {
                        Console.WriteLine($"IP14 : {ex.Message}");
```

```
                    if (ex.Message == "Error reading data from the connection.")
                    {
                        FbConnection.ClearPool(fbcon);
                    }
                }
                finally
                {
                    reader.Close();

                }
            }
            catch (Exception ex)
            {
                Console.WriteLine($"72 : {ex.Message}");
                if (ex.Message == "Error reading data from the connection.")
                {
                    FbConnection.ClearPool(fbcon);
                }
            }
            finally
            {
                fbcon.Close();
            }

            return status;
        }
    } // проверка существования подтверждения в таблице
    public List<TemplateTestConfirmation> selectConfirmation()
    {
        using (FbConnection fbcon = newfbconn_con())
        {
            List<TemplateTestConfirmation> confirmations = new
List<TemplateTestConfirmation>();
            FbCommand SelectSQL = new FbCommand(String.Format($"SELECT idInfodent, date,
status FROM theconfirmationtable WHERE processed = '0'"));
            try
            {
                fbcon.Open();
                FbTransaction fbt = fbcon.BeginTransaction();
                SelectSQL.Transaction = fbt;
                FbDataReader reader = SelectSQL.ExecuteReader();

                try
                {
                    while (reader.Read())
                    {
                        TemplateTestConfirmation confirmation = new
TemplateTestConfirmation();
                        confirmation.schedid = reader.GetInt64(0);
                        confirmation.createAt = reader.GetDateTime(1);
                        confirmation.status = reader.GetInt16(2);
                        confirmations.Add(confirmation);
                    }
                }
                catch (Exception ex)
                {
                    Console.WriteLine($"IP14 : {ex.Message}");
                    if (ex.Message == "Error reading data from the connection.")
                    {
                        FbConnection.ClearPool(fbcon);
                    }
                }
                Finally
                {
                    reader.Close();
                }
            }
```

```
                catch (Exception ex)
                {
                    Console.WriteLine($"72 : {ex.Message}");
                    if (ex.Message == "Error reading data from the connection.")
                    {

                        FbConnection.ClearPool(fbcon);
                    }
                }
                finally
                {
                    fbcon.Close();
                }
                return confirmations;
            }
        } // выборка необработанных подтверждений

        public bool updateTheconfirmationtableProcessed(TemplateTestConfirmation confirmation,
int oldTreat)  // Обновляю статус обработанной записи в таблице подтверждений
        {
            bool status = false;
            string sqlstring = "";
            switch (oldTreat)
            {
                case 0:
                    sqlstring = String.Format($"UPDATE theconfirmationtable SET processed= '1'
WHERE idInfodent ='{confirmation.schedid}' and date = '{confirmation.createAt}' and status =
'{confirmation.status}'");
                    break;
                case 1:
                    sqlstring = String.Format($"UPDATE theconfirmationtable SET processed= '1',
oldTreat = '1' WHERE idInfodent ='{confirmation.schedid}' and date = '{confirmation.createAt}'
and status = '{confirmation.status}'");
                    break;
            }
            using (FbConnection fbcon = newfbconn_con())
            {
                FbCommand InsertSQL =
                   new FbCommand(sqlstring, fbcon);// Добавить аргументы
                try
                {
                    fbcon.Open();
                    FbTransaction fbt = fbcon.BeginTransaction();
                    InsertSQL.Transaction = fbt;

                    int res = InsertSQL.ExecuteNonQuery();
                    fbt.Commit();
                }
                catch (Exception ex)
                {
                    Console.WriteLine($"51 : {ex.Message}");
                    if (ex.Message == "Error reading data from the connection.")
                    {
                        FbConnection.ClearPool(fbcon);
                    }
                }
                finally
                {
                    InsertSQL.Dispose();
                    fbcon.Close();
                }

                return status;
            }
        }
    }
}
```

ПРИЛОЖЕНИЕ Ж

ИСХОДНЫЙ КОД MESSAGE.PROGRAM

```
using Newtonsoft.Json;
using System;
using System.Collections.Generic;
using System.IO;
using System.Linq;
using System.Net;
using System.Text;
using System.Text.RegularExpressions;
using System.Threading;
using System.Threading.Tasks;
using static KitPoint___Message.TemplateResponsSmsRuGetStatus;

namespace KitPoint___Message
{
    class Program
    {
        static void Send1SMS()//вызов отправки 1 смс ожидание 30 сек
        {
            ConnectionDB db = new ConnectionDB();
            int UpgrateTimeSend1Sms = 1;
            DateTime TimeSend1Sms = DateTime.Now;
            while (true)
            {
                if (DateTime.Now.Hour < 23 && DateTime.Now.Hour > 6)
                {
                    if (db.testOpenDBs() == 1)
                    {
                        if (db.getBalanseRu() > 20)
                        {
                            try
                            {
                                if ((db.getStatusSendingSMS1()) == 1)
                                {
                                    if (UpgrateTimeSend1Sms == 1)
                                    {
                                        TimeSend1Sms = DateTime.Parse(db.UpdateDateSend1Sms());
                                        UpgrateTimeSend1Sms = 0;
                                        Console.WriteLine("Запущен:" + DateTime.Now.ToString());
                                    }

                                    db.SMS1(TimeSend1Sms);
                                    Thread.Sleep(30000);
                                }
                                else
                                {
                                    Thread.Sleep(60000);
                                    UpgrateTimeSend1Sms = 1;
                                }
                            }
                            catch (Exception ex)
                            {
                                Console.WriteLine($"1 : {ex.Message}");
                            }
                        }
                        else
                        {
                            Console.WriteLine("Не хватает денег на балансе для отправки 1 СМС!");
                            UpgrateTimeSend1Sms = 1;
                            Thread.Sleep(60000);
                        }
                    }
                    else
                    {
                        Console.WriteLine("Нет подключения к БД!");
```

```
                    UpgrateTimeSend1Sms = 1;
                    Thread.Sleep(60000);
                }
            }
            else Thread.Sleep(60000);
        }
    }
    static void Send2SMS()//вызов отправки 2 смс за день до приема, раз в сутки
    {
        ConnectionDB db = new ConnectionDB();
        bool flag = true;
        while (true)
        {
            if (DateTime.Now.Hour < 22 && DateTime.Now.Hour > 8)
            {
                if (db.testOpenDBs() == 1)
                {
                    if (db.getBalanseRu() > 700)
                    {
                        try
                        {
                            if ((db.getStatusSendingSMS2()) == 1)
                            {
                                TimeSpan time = db.timeSend1Day();

                                if (flag == true)
                                {
                                    if (DateTime.Now.TimeOfDay < time)
                                    {
                                        DateTime now = DateTime.Now;
                                        int timeSleepp = (time.Hours - now.Hour) * 3600000 +
(time.Minutes - now.Minute) * 60000 +
                                            (time.Seconds - now.Second) * 1000 +
now.Millisecond;
                                        Console.WriteLine("Жду до 12: " + DateTime.Now + " "
+ timeSleepp);
                                        Thread.Sleep(timeSleepp);
                                        Console.WriteLine("Проснулся: " + DateTime.Now);
                                    }
                                    else
                                    {
                                        DateTime now = DateTime.Now;
                                        int timeSleepp = (now.Hour - time.Hours) * 3600000 +
(now.Minute - time.Minutes) * 60000 +
                                    (now.Second - time.Seconds) * 1000 + now.Millisecond;
                                        int days = 86400000;
                                        Console.WriteLine("Жду 12 завтра: " + DateTime.Now +
" " + (days - timeSleepp));
                                        Thread.Sleep(days - timeSleepp);
                                        Console.WriteLine("Проснулся: " + DateTime.Now);
                                    }
                                    flag = false;
                                }

                                if (db.getBalanseRu() > 700)
                                {
                                    db.SMS2();

                                    Console.WriteLine("Отправил 2СМС: " + DateTime.Now);

                                    Thread.Sleep(3600000); // для формирования отчета 1 час
времени

                                    db.createReport();
                                    Console.WriteLine("Создал отчет: " + DateTime.Now);
                                }
```

```
                                    else
                                    {
                                        Console.WriteLine("Не хватает денег на балансе для
отправки 2 СМС!");
                                    }

                                    DateTime sendDateTime = DateTime.Now;
                                    Console.WriteLine(sendDateTime.AddHours(-1));

                                    int timeSleep = (sendDateTime.Hour - time.Hours) * 3600000 +
(sendDateTime.Minute - time.Minutes) * 60000 +
                                        (sendDateTime.Second - time.Seconds) * 1000 +
sendDateTime.Millisecond;
                                    int day = 86400000;

                                    Console.WriteLine("Засёк время: " + (day - timeSleep));

                                    Thread.Sleep(day - timeSleep);
                                }
                                else Thread.Sleep(60000);
                            }
                            catch (Exception ex)
                            {
                                Console.WriteLine($"2 : {ex.Message}");
                            }
                        }
                        else
                        {
                            Console.WriteLine("Не хватает денег на балансе для отправки 2 СМС!");
                            flag = true;
                            Thread.Sleep(60000);
                        }
                    }
                    else
                    {
                        Console.WriteLine("Нет подключения к БД!");
                        flag = true;
                        Thread.Sleep(60000);
                    }
                }
                else
                {
                    flag = true;
                    Thread.Sleep(60000);
                }
            }

        }

        static void Send3SMS()//вызов отправки 3 смс в день приема, каждые 15 минут
        {
            ConnectionDB db = new ConnectionDB();
            while (true)
            {
                if (db.testOpenDBs() == 1)
                {
                    if (db.getBalanseRu() > 40)
                    {
                        if (DateTime.Now.Hour < 22 && DateTime.Now.Hour > 6)
                        {
                            try
                            {
                                if ((db.getStatusSendingSMS3()) == 1)
                                {
                                    db.SMS3();
                                    Thread.Sleep(600000);
```

```
                            }
                            else Thread.Sleep(60000);
                        }
                        catch (Exception ex)
                        {
                            Console.WriteLine($"3 : {ex.Message}");
                        }
                    }
                    else Thread.Sleep(60000);
                }
                else
                {
                    Console.WriteLine("Не хватает денег на балансе для отправки 3 СМС!");
                    Thread.Sleep(60000);
                }
            }
            else
            {
                Console.WriteLine("Нет подключения к БД!");
                Thread.Sleep(60000);
            }
        }
    }

    public void sendSms(string phone, string textSms, long idInf, short typeSms) // отправка
sms
    {
        ConnectionDB db = new ConnectionDB();
        System.Net.HttpWebRequest reqGET =
(System.Net.HttpWebRequest)System.Net.WebRequest.Create("" + phone + "&text=" + textSms);

        WebProxy myProxy = new WebProxy("", 3128);
        myProxy.Credentials = new NetworkCredential("", "");
        reqGET.Proxy = HttpWebRequest.DefaultWebProxy;
        reqGET.Proxy.Credentials = CredentialCache.DefaultNetworkCredentials;
        reqGET.PreAuthenticate = true;
        reqGET.Proxy = myProxy;
        var resp = (HttpWebResponse)reqGET.GetResponse();
        using (var sr = new StreamReader(resp.GetResponseStream()))
        {
            string respons = sr.ReadToEnd();

            string pattern1 = @";([0-9A-F]+)";
            Regex regex = new Regex(pattern1);
            var codes = regex.Match(respons);
            string id = "";
            while (codes.Success)
            {
                id = codes.Groups[1].Value;
                db.insertStatusSMS(id, idInf, typeSms);
                codes = codes.NextMatch();
            }
        }
    }

    public void sendSmsRu(string phone, string textSms, long idInf, short typeSms) //
отправка sms
    {
        ConnectionDB db = new ConnectionDB();
        System.Net.HttpWebRequest reqGET =
(System.Net.HttpWebRequest)System.Net.WebRequest.Create("" + phone + "&msg=" + textSms +
"&json=1");

        WebProxy myProxy = new WebProxy("", );
        myProxy.Credentials = new NetworkCredential("", "");
        reqGET.Proxy = HttpWebRequest.DefaultWebProxy;
```

```
            reqGET.Proxy.Credentials = CredentialCache.DefaultNetworkCredentials;
            reqGET.PreAuthenticate = true;
            reqGET.Proxy = myProxy;
            var resp = (HttpWebResponse)reqGET.GetResponse();
            using (var sr = new StreamReader(resp.GetResponseStream()))
            {
                string respons = sr.ReadToEnd();

                string pattern1 = "\"sms_id\": \"([0-9-]+)\"";
                Regex regex = new Regex(pattern1);
                var codes = regex.Match(respons);
                string id = "";
                while (codes.Success)
                {
                    id = codes.Groups[1].Value;
                    db.insertStatusSMS(id, idInf, typeSms);
                    db.insertTestStatusSendSms(id, typeSms, idInf);
                    codes = codes.NextMatch();
                }
            }
        }


        static double getBalanseDiscont() // получение баланса
        {
            double totlaBalanse = 0;
            System.Net.HttpWebRequest reqGET =
(System.Net.HttpWebRequest)System.Net.WebRequest.Create("");

            WebProxy myProxy = new WebProxy("", );
            myProxy.Credentials = new NetworkCredential("", "");
            reqGET.Proxy = HttpWebRequest.DefaultWebProxy;
            reqGET.Proxy.Credentials = CredentialCache.DefaultNetworkCredentials;
            reqGET.PreAuthenticate = true;
            reqGET.Proxy = myProxy;
            var resp = (HttpWebResponse)reqGET.GetResponse();
            using (var sr = new StreamReader(resp.GetResponseStream()))
            {
                var result = sr.ReadToEnd();
                string pattern1 = "RUB;([0-9]+.[0-9]+);";
                Regex regex = new Regex(pattern1);
                Match balanse = regex.Match(result);
                while (balanse.Success)
                {
                    string temp = balanse.Groups[1].Value;
                    temp = temp.Replace('.', ',');
                    totlaBalanse = Convert.ToDouble(temp);
                    balanse = balanse.NextMatch();
                }
            }
            return totlaBalanse;
        }

        static void getStatusSms(string id, short type) // получения статуса sms
        {
            try
            {
                ConnectionDB db = new ConnectionDB();
                System.Net.HttpWebRequest reqGET =
(System.Net.HttpWebRequest)System.Net.WebRequest.Create("" + id);

                WebProxy myProxy = new WebProxy("", );
                myProxy.Credentials = new NetworkCredential("", "");
                reqGET.Proxy = HttpWebRequest.DefaultWebProxy;
                reqGET.Proxy.Credentials = CredentialCache.DefaultNetworkCredentials;
                reqGET.PreAuthenticate = true;
```

```
                reqGET.Proxy = myProxy;
                var resp = (HttpWebResponse)reqGET.GetResponse();
                using (var sr = new StreamReader(resp.GetResponseStream()))
                {
                    string respons = sr.ReadToEnd();

                    string pattern1 = @";([a-z ]+)";
                    Regex regex = new Regex(pattern1);
                    var status = regex.Match(respons);
                    string idStr = "";
                    while (status.Success)
                    {
                        idStr = status.Groups[1].Value;
                        db.updateStatusSMS(id, idStr, type);
                        status = status.NextMatch();
                    }
                }
            }
            catch (Exception ex)
            {
                Console.WriteLine($"4 : {ex.Message}");
            }
        }

        static void getStatusSmsRu(string id, short type) // получения статуса sms
        {
            try
            {
                ConnectionDB db = new ConnectionDB();
                System.Net.HttpWebRequest reqGET =
(System.Net.HttpWebRequest)System.Net.WebRequest.Create("" + id + "&json=1");

                WebProxy myProxy = new WebProxy("", );
                myProxy.Credentials = new NetworkCredential("", "");
                reqGET.Proxy = HttpWebRequest.DefaultWebProxy;
                reqGET.Proxy.Credentials = CredentialCache.DefaultNetworkCredentials;
                reqGET.PreAuthenticate = true;
                reqGET.Proxy = myProxy;
                var resp = (HttpWebResponse)reqGET.GetResponse();
                using (var sr = new StreamReader(resp.GetResponseStream()))
                {
                    string respons = sr.ReadToEnd();
                    var obj = JsonConvert.DeserializeObject<RootObject>(respons);
                    string idStr = obj.sms[id].status_code;

                    if (idStr != null)
                    {
                        db.updateStatusSMSRu(id, idStr, type);
                    }
                }
            }
            catch (Exception ex) { Console.WriteLine($"5 : {ex.Message}"); }
        }

        static void updateStatusSMS1()
        {
            ConnectionDB db = new ConnectionDB();

            List<string> ids = new List<string>();
            ids.AddRange(db.getIdSms1Ru());
            foreach (var id in ids)
            {
                getStatusSmsRu(id, 1);
            }
        }
```

```
static void updateStatusSMS2()
{
    ConnectionDB db = new ConnectionDB();
    List<string> ids = new List<string>();
    ids.AddRange(db.getIdSms2Ru());
    foreach (var id in ids)
    {
        getStatusSmsRu(id, 2);
    }
}

static void updateStatusSMS3()
{
    ConnectionDB db = new ConnectionDB();
    List<string> ids = new List<string>();
    ids.AddRange(db.getIdSms3Ru());
    foreach (var id in ids)
    {
        getStatusSmsRu(id, 3);
    }
}

static void updateStatusSMS()
{
    ConnectionDB db = new ConnectionDB();
    while (true)
    {
        if (db.testOpenDBs() == 1)
        {
            updateStatusSMS1();
            updateStatusSMS2();
            updateStatusSMS3();
            Thread.Sleep(1000);
        }
        else
        {
            Console.WriteLine("Нет подключения к БД!");
            Thread.Sleep(60000);
        }
    }
}

static void testSendSmsFunction()
{
    while (true)
    {
        try
        {
            ConnectionDB db = new ConnectionDB();
            if (db.testOpenDBs() == 1)
            {
                List<testSendSmsModel> list = db.selectStatusSendSms();
                if (list.Count != 0)
                {
                    foreach (var sms in list)
                    {
                        db.getStatusSmsRu(sms.idSms, sms.typeSms, sms.idInf);
                        db.updateTestSendSms(sms.idSms);
                    }
                }
                Thread.Sleep(1000);
            }
            else
            {
                Console.WriteLine("Нет подключения к БД!");
                Thread.Sleep(60000);
```

```
                }
            }
            catch (Exception ex)
            {
                Console.WriteLine($"6 : {ex.Message}");
            }
        }
    }

    static void Main(string[] args)
    {
        ConnectionDB db = new ConnectionDB();
        Thread thread1 = new Thread(Send1SMS);
        Thread thread2 = new Thread(Send2SMS);
        Thread thread3 = new Thread(Send3SMS);
        Thread thread4 = new Thread(updateStatusSMS);
        Thread thread5 = new Thread(testSendSmsFunction);

        thread1.Start();
        thread2.Start();
        thread3.Start();
        thread4.Start();
        thread5.Start();

        //db.testOpenDBs();

        while (true)
        {
            Thread.Sleep(100);
        }
    }
  }
}
```

ПРИЛОЖЕНИЕ 3

ИСХОДНЫЙ КОД MESSAGE.CONNECTIONDB

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using MySql.Data.MySqlClient;
using FirebirdSql.Data.FirebirdClient;
using System.Drawing;
using Excel = Microsoft.Office.Interop.Excel;
using System.Net;
using System.IO;
using System.Text.RegularExpressions;
using static KitPoint___Message.TemplateResponsSmsRuGetStatus;
using Newtonsoft.Json;

namespace KitPoint___Message
{
    class ConnectionDB
    {
        public ConnectionDB()//подключение к базе
        { }

        public FbConnection newfbcon_co()
        {
            FbConnection fbco;
            FbConnectionStringBuilder fbco_con = new FbConnectionStringBuilder();
            //fbco_con.Charset = "WIN1251";
            fbco_con.UserID = "";
            fbco_con.Password = "";
            fbco_con.Database = "";
            fbco_con.ServerType = 0;
            fbco_con.ConnectionLifeTime = 60;
            fbco = new FbConnection(fbco_con.ToString());

            return fbco;
        }

        public FbConnection newfbww001_con()
        {
            FbConnection fbww001;
            // Подключение к базе ww001
            FbConnectionStringBuilder fbww001_con = new FbConnectionStringBuilder();
            // fbww001_con.Charset = "WIN1251";
            fbww001_con.UserID = "";
            fbww001_con.Password = "";
            fbww001_con.Database = "";
            fbww001_con.ServerType = 0;
            fbww001_con.ConnectionLifeTime = 60;
            fbww001 = new FbConnection(fbww001_con.ToString());

            return fbww001;
        }

        public FbConnection newfbww002_con()
        {
            FbConnection fbww002;
            // Подключение к базе ww002
            FbConnectionStringBuilder fbww002_con = new FbConnectionStringBuilder();
            //fbww002_con.Charset = "WIN1251";
            fbww002_con.UserID = "";
            fbww002_con.Password = "";
            fbww002_con.Database = "";
            fbww002_con.ServerType = 0;
            fbww002_con.ConnectionLifeTime = 60;
```

```
    fbww002 = new FbConnection(fbww002_con.ToString());

    return fbww002;
}

public FbConnection newfbww003_con()
{
    FbConnection fbww003;
    // Подключение к базе ww003
    FbConnectionStringBuilder fbww003_con = new FbConnectionStringBuilder();
    //fbww003_con.Charset = "WIN1251";
    fbww003_con.UserID = "";
    fbww003_con.Password = "";
    fbww003_con.Database = "";
    fbww003_con.ServerType = 0;
    fbww003_con.ConnectionLifeTime = 60;
    fbww003 = new FbConnection(fbww003_con.ToString());

    return fbww003;
}

public FbConnection newfbww005_con()
{
    FbConnection fbww005;
    // Подключение к базе ww005
    FbConnectionStringBuilder fbww005_con = new FbConnectionStringBuilder();
    // fbww005_con.Charset = "WIN1251";
    fbww005_con.UserID = "";
    fbww005_con.Password = "";
    fbww005_con.Database = "";
    fbww005_con.ServerType = 0;
    fbww005_con.ConnectionLifeTime = 60;
    fbww005 = new FbConnection(fbww005_con.ToString());

    return fbww005;
}

public FbConnection newfbconn_con()
{
    FbConnection fbconn;
    FbConnectionStringBuilder fbconn_con = new FbConnectionStringBuilder();
    //fbco_con.Charset = "WIN1251";
    fbconn_con.UserID = "";
    fbconn_con.Password = "";
    fbconn_con.Database = "";
    fbconn_con.ServerType = 0;
    fbconn_con.ConnectionLifeTime = 60;
    fbconn = new FbConnection(fbconn_con.ToString());

    return fbconn;
}

public int testOpenDBs() // Проверка открытия баз данных
{
    using (FbConnection conn = newfbconn_con())
    using (FbConnection fbco = newfbcon_co())
    {
        int test = 0;
        try
        {
            fbco.Open();
            conn.Open();
            test = 1;
        }
        catch (Exception ex)
        {
```

```
                    Console.WriteLine($"12 : {ex.Message}");
                    test = 3;
                    if (ex.Message == "Error reading data from the connection.")
                    {
                        FbConnection.ClearPool(fbco);
                    }
                }
                finally
                {
                    fbco.Close();
                    conn.Close();
                }

                return test;
            }
        }

        public void SMS1(DateTime TimeSend1Sms)//отправка смс 1
        {
            using (FbConnection fbcon = newfbconn_con())
            {
                try
                {
                    List<long> idInfodent = new List<long>();
                    FbCommand SelectSQL = new FbCommand(String.Format($"select idInfodent from
kitpointdb.scheduleofmessages where dateAdd > (select DateStartSend1Sms from
kitpointdb.configurationcare where idConfiguration = 1) and wasItSent1 = 0 and obsolete = '0' " +
                            $"and dateOfReceipt > current_timestamp()"), fbcon);

                    try
                    {
                        fbcon.Open();
                        FbTransaction fbt = fbcon.BeginTransaction();
                        SelectSQL.Transaction = fbt;
                        FbDataReader reader = SelectSQL.ExecuteReader();

                        try
                        {
                            while (reader.Read())
                            {
                                idInfodent.Add(reader.GetInt64(0));
                            }
                        }
                        catch (Exception ex)
                        {
                            Console.WriteLine($"MES_1 : {ex.Message}");
                            if (ex.Message == "Error reading data from the connection.")
                            {
                                FbConnection.ClearPool(fbcon);
                            }
                        }
                        finally
                        {
                            reader.Close();
                        }

                    }
                    catch (Exception ex)
                    {
                        Console.WriteLine($"7.1 : {ex.Message}");
                        if (ex.Message == "Error reading data from the connection.")
                        {
                            FbConnection.ClearPool(fbcon);
                        }
                    }
                    Finally
```

```
                    {
                        fbcon.Close();
                    }

                    TemplateRecordShedule record = new TemplateRecordShedule();
                    foreach (var idInf in idInfodent)
                    {
                        try
                        {
                            record = record.AddInformation(idInf);
                            string str = getText(1);
                            string adress = getAdressFilial(record.filial);
                            string dname = getDname(record.idDoctor);
                            if (dname[2] == ' ') dname = dname.Remove(0, 3);
                            if (dname == "Томограф") str = str.Replace(" Врач [dname]", "");
                            dname = dname.Replace("(врач)", "").Replace("(Врач)", "");
                            str = str.Replace("[name]", record.clientName).Replace("[date]",
record.dateOfReceipt.ToShortDateString().Remove(5, 5)).Replace("[filial]",
adress).Replace("[dname]", dname).Replace("[time]", record.dateOfReceipt.ToShortTimeString());
                            Program model = new Program();
                            model.sendSmsRu(record.phoneNumber, str, idInf, 1);
                            insertTextSms(idInf + str);

                            updateWasItSent1(idInf);
                        }
                        catch (Exception ex)
                        {
                            Console.WriteLine($"7.2 : {idInf} : {ex.Message}");
                        }
                    }
                }
                catch (Exception ex)
                {
                    Console.WriteLine($"7 : {ex.Message}");
                }
            }
        }

        public void updateWasItSent1(long idInf)
        {
            using (FbConnection fbcon = newfbconn_con())
            {
                string sqlstring = String.Format($"UPDATE scheduleofmessages SET wasItSent1 = 1
WHERE idInfodent = {idInf}  and obsolete = '0'");
                FbCommand InsertSQL =
                   new FbCommand(sqlstring, fbcon);// Добавить аргументы
                try
                {
                    fbcon.Open();
                    FbTransaction fbt = fbcon.BeginTransaction();
                    InsertSQL.Transaction = fbt;
                    int res = InsertSQL.ExecuteNonQuery();
                    fbt.Commit();
                }
                catch (Exception ex)
                {
                    Console.WriteLine($"7.3 : {ex.Message}");
                    if (ex.Message == "Error reading data from the connection.")
                    {
                        FbConnection.ClearPool(fbcon);
                    }
                }
                finally
                {
                    InsertSQL.Dispose();
                    fbcon.Close();
```

```
                }
            }
        }

        public string UpdateDateSend1Sms()  // Обновляю дату старта отправки 1 смс
        {
            using (FbConnection fbcon = newfbconn_con())
            {
                string now = DateTime.Now.ToString("yyyy-MM-dd HH:mm:ss");
                string sqlstring = String.Format("UPDATE configurationcare SET
DateStartSend1Sms='" + now + "' WHERE idConfiguration='1';");
                FbCommand InsertSQL =
                   new FbCommand(sqlstring, fbcon);// Добавить аргументы
                try
                {
                    fbcon.Open();
                    FbTransaction fbt = fbcon.BeginTransaction();
                    InsertSQL.Transaction = fbt;
                    int res = InsertSQL.ExecuteNonQuery();
                    fbt.Commit();
                }
                catch (Exception ex)
                {
                    Console.WriteLine($"8 : {ex.Message}");
                    if (ex.Message == "Error reading data from the connection.")
                    {
                        FbConnection.ClearPool(fbcon);
                    }
                }
                finally
                {
                    InsertSQL.Dispose();
                    fbcon.Close();
                }

                return now;
            }
        }

        public string getDname(long idDoctor)
        {
            using (FbConnection fbco = newfbcon_co())
            {
                string dname = "";
                FbCommand SelectSQL = new FbCommand(String.Format($"select dname from doctor
where dcode = '{idDoctor}'"), fbco);

                try
                {
                    fbco.Open();
                    FbTransaction fbt = fbco.BeginTransaction();
                    SelectSQL.Transaction = fbt;
                    FbDataReader reader = SelectSQL.ExecuteReader();

                    try
                    {
                        while (reader.Read())
                        {
                            dname = reader.GetString(0);
                        }
                    }
                    catch (Exception ex)
                    {
                        Console.WriteLine($"MES_2 : {ex.Message}");
                        if (ex.Message == "Error reading data from the connection.")
                        {
```

```
                        FbConnection.ClearPool(fbco);
                    }
                }
                finally
                {
                    reader.Close();
                }
            }
            catch (Exception ex)
            {
                Console.WriteLine($"9 : {ex.Message}");
                if (ex.Message == "Error reading data from the connection.")
                {
                    FbConnection.ClearPool(fbco);
                }
            }
            finally
            {
                fbco.Close();
            }

            return dname;
        }
    }

    public void SMS2()//отправка смс 2
    {
        using (FbConnection fbcon = newfbconn_con())
        {
            try
            {
                List<long> idInfodent = new List<long>();
                List<long> idInfodentWithNoConfirmation = new List<long>();
                FbCommand SelectSQL = new FbCommand(String.Format($"select idInfodent from
scheduleofmessages where wasItSent2 = '0' and date(timeSendMessage2) = CURDATE() and obsolete =
'0'"));
                try
                {
                    fbcon.Open();
                    FbTransaction fbt = fbcon.BeginTransaction();
                    SelectSQL.Transaction = fbt;
                    FbDataReader reader = SelectSQL.ExecuteReader();
                    try
                    {
                        while (reader.Read())
                        {
                            idInfodent.Add(reader.GetInt64(0));
                        }
                    }
                    catch (Exception ex)
                    {
                        Console.WriteLine($"MES_3 : {ex.Message}");
                        if (ex.Message == "Error reading data from the connection.")
                        {
                            FbConnection.ClearPool(fbcon);
                        }
                    }
                    finally
                    {
                        reader.Close();
                    }
                }
                catch (Exception ex)
                {
                    Console.WriteLine($"10.1 : {ex.Message}");
                    if (ex.Message == "Error reading data from the connection.")
```

```
                    {
                        FbConnection.ClearPool(fbcon);
                    }
                }
                finally
                {
                    fbcon.Close();
                }

                try
                {
                    foreach (var idInf in idInfodent) // проверка на раннее подтверждение
                    {
                        if (testConfirmation(idInf) == 1)
                        {
                            updateInfodentWithConfirmation(idInf);
                        }
                        else
                        {
                            idInfodentWithNoConfirmation.Add(idInf);
                        }
                    }
                }
                catch (Exception ex)
                {
                    Console.WriteLine($"10.2 : {ex.Message}");
                }


                TemplateRecordShedule record = new TemplateRecordShedule();
                foreach (var idInf in idInfodentWithNoConfirmation)
                {
                    try
                    {
                        record = record.AddInformation(idInf);
                        string str = getText(2);
                        string adress = getAdressFilial(record.filial);
                        string dname = getDname(record.idDoctor);
                        if (dname[2] == ' ') dname = dname.Remove(0, 3);
                        if (dname == "Томограф") str = str.Replace(" Врач [dname]", "");
                        dname = dname.Replace("(врач)", "").Replace("(Врач)", "");
                        string url = "" + idInf;
                        str = str.Replace("[name]", record.clientName).Replace("[date]",
record.dateOfReceipt.ToShortDateString().Remove(5, 5)).Replace("[filial]",
adress).Replace("[dname]", dname).Replace("[time]",
record.dateOfReceipt.ToShortTimeString()).Replace("[url]", url);
                        Program model = new Program();
                        model.sendSmsRu(record.phoneNumber, str, idInf, 2);
                        insertTextSms(idInf + str);

                        updateWasItSent2(idInf);
                    }
                    catch (Exception ex)
                    {
                        Console.WriteLine($"10.3 : {idInf} : {ex.Message}");
                    }
                }
            }
            catch (Exception ex)
            {
                Console.WriteLine($"10 : {ex.Message}");
            }
        }
    }

    public void updateWasItSent2(long idInf) // отметка что 2 смс была отправлена
```

```
        {
            using (FbConnection fbcon = newfbconn_con())
            {
                string sqlstring = String.Format($"UPDATE scheduleofmessages SET wasItSent2 = 1,
" +
                    $"timeSendMessage2Total = '{DateTime.Now.ToString("yyyy-MM-dd HH:mm:ss")}'
WHERE idInfodent = {idInf} and obsolete = '0'");
                FbCommand InsertSQL =
                   new FbCommand(sqlstring, fbcon);// Добавить аргументы
                try
                {
                    fbcon.Open();
                    FbTransaction fbt = fbcon.BeginTransaction();
                    InsertSQL.Transaction = fbt;
                    int res = InsertSQL.ExecuteNonQuery();
                    fbt.Commit();
                }
                catch (Exception ex)
                {
                    Console.WriteLine($"10.4 : {ex.Message}");
                    if (ex.Message == "Error reading data from the connection.")
                    {
                        FbConnection.ClearPool(fbcon);
                    }
                }
                finally
                {
                    InsertSQL.Dispose();
                    fbcon.Close();
                }
            }
        }

        public void updateInfodentWithConfirmation(long idInf) // отметить в базе что прием был
ранее подтвержден
        {
            using (FbConnection fbcon = newfbconn_con())
            {
                string sqlstring = String.Format($"UPDATE scheduleofmessages SET wasItSent2 = 4
WHERE idInfodent = {idInf} and obsolete = '0'"); // был ранее подтвержден 4
                FbCommand InsertSQL =
                   new FbCommand(sqlstring, fbcon);// Добавить аргументы
                try
                {
                    fbcon.Open();
                    FbTransaction fbt = fbcon.BeginTransaction();
                    InsertSQL.Transaction = fbt;
                    int res = InsertSQL.ExecuteNonQuery();
                    fbt.Commit();
                }
                catch (Exception ex)
                {
                    Console.WriteLine($"10.5 : {ex.Message}");
                    if (ex.Message == "Error reading data from the connection.")
                    {
                        FbConnection.ClearPool(fbcon);
                    }
                }
                finally
                {
                    InsertSQL.Dispose();
                    fbcon.Close();
                }
            }
        }
```

```
        public void SMS3()//отправка смс 3
        {
            using (FbConnection fbcon = newfbconn_con())
            {
                try
                {
                    List<long> idInfodent = new List<long>();
                    FbCommand SelectSQL = new FbCommand(String.Format($"select idInfodent from
scheduleofmessages where wasItSent3 = '0' and date(timeSendMessage3) = CURDATE() and" +
                                $" time(timeSendMessage3) between time(now()) and
time(DATE_ADD(NOW(),Interval 10 MINUTE)) and obsolete = '0'"), fbcon);
                    try
                    {
                        fbcon.Open();
                        FbTransaction fbt = fbcon.BeginTransaction();
                        SelectSQL.Transaction = fbt;
                        FbDataReader reader = SelectSQL.ExecuteReader();

                        try
                        {
                            while (reader.Read())
                            {
                                idInfodent.Add(reader.GetInt64(0));
                            }
                        }
                        catch (Exception ex)
                        {
                            Console.WriteLine($"MES_4 : {ex.Message}");
                            if (ex.Message == "Error reading data from the connection.")
                            {
                                FbConnection.ClearPool(fbcon);
                            }
                        }
                        finally
                        {
                            reader.Close();
                        }
                    }
                    catch (Exception ex)
                    {
                        Console.WriteLine($"11.1 : {ex.Message}");
                        if (ex.Message == "Error reading data from the connection.")
                        {
                            FbConnection.ClearPool(fbcon);
                        }
                    }
                    finally
                    {
                        fbcon.Close();
                    }

                    TemplateRecordShedule record = new TemplateRecordShedule();
                    foreach (var idInf in idInfodent)
                    {
                        try
                        {
                            record = record.AddInformation(idInf);
                            string str = getText(3);
                            string adress = getAdressFilial(record.filial);
                            string dname = getDname(record.idDoctor);
                            if (dname[2] == ' ') dname = dname.Remove(0, 3);
                            dname = dname.Replace("(врач)", "").Replace("(Врач)", "");
                            str = str.Replace("[name]", record.clientName).Replace("[filial]",
adress).Replace("[time]", record.dateOfReceipt.ToShortTimeString());
                            Program model = new Program();
                            model.sendSmsRu(record.phoneNumber, str, idInf, 3);
```

```
                            insertTextSms(idInf + str);

                            updateWasItSent3(idInf);
                        }
                        catch (Exception ex)
                        {
                            Console.WriteLine($"11.2 : {idInf} : {ex.Message}");
                        }
                    }
                }
                catch (Exception ex)
                {
                    Console.WriteLine($"11 : {ex.Message}");
                }
            }
        }

        public void updateWasItSent3(long idInf) // отметка что 3  была отправлена
        {
            using (FbConnection fbcon = newfbconn_con())
            {
                string sqlstring = String.Format($"UPDATE scheduleofmessages SET wasItSent3 = 1
WHERE idInfodent = {idInf} and obsolete = '0'"); // был ранее подтвержден 4
                FbCommand InsertSQL =
                   new FbCommand(sqlstring, fbcon);
                try
                {
                    fbcon.Open();
                    FbTransaction fbt = fbcon.BeginTransaction();
                    InsertSQL.Transaction = fbt;
                    int res = InsertSQL.ExecuteNonQuery();
                    fbt.Commit();
                }
                catch (Exception ex)
                {
                    Console.WriteLine($"11.3 : {ex.Message}");
                    if (ex.Message == "Error reading data from the connection.")
                    {
                        FbConnection.ClearPool(fbcon);
                    }
                }
                finally
                {
                    InsertSQL.Dispose();
                    fbcon.Close();
                }
            }
        }

        public void insertTestStatusSendSms(string idSms, short typeSms, long idInf)
        {
            using (FbConnection fbcon = newfbconn_con())
            {
                DateTime dateTest = DateTime.Now.AddMinutes(15);
                string sqlstring = String.Format("INSERT INTO
teststatussendsms(idSms,typeSms,idInf,timeTest) " +
                            $"VALUES('{idSms}', '{typeSms}', '{idInf}',
'{dateTest.ToString("yyyy-MM-dd HH:mm:ss")}');");

                FbCommand InsertSQL = new FbCommand(sqlstring, fbcon);
                try
                {
                    fbcon.Open();
                    FbTransaction fbt = fbcon.BeginTransaction();
                    InsertSQL.Transaction = fbt;
                    int res = InsertSQL.ExecuteNonQuery();
```

```
                    fbt.Commit();
                }
                catch (Exception ex)
                {
                    Console.WriteLine($"13 : {ex.Message}");
                    if (ex.Message == "Error reading data from the connection.")
                    {
                        FbConnection.ClearPool(fbcon);
                    }
                }
                finally
                {
                    InsertSQL.Dispose();
                    fbcon.Close();
                }
            }
        }

        public void insertTextSms(string text)
        {
            using (FbConnection fbcon = newfbconn_con())
            {
                DateTime dateTest = DateTime.Now.AddMinutes(15);
                string sqlstring = String.Format($"INSERT INTO smssend(textSms)
VALUES('{text}');");

                FbCommand InsertSQL = new FbCommand(sqlstring, fbcon);
                try
                {
                    fbcon.Open();
                    FbTransaction fbt = fbcon.BeginTransaction();
                    InsertSQL.Transaction = fbt;
                    int res = InsertSQL.ExecuteNonQuery();
                    fbt.Commit();
                }
                catch (Exception ex)
                {
                    Console.WriteLine($"14 : {ex.Message}");
                    if (ex.Message == "Error reading data from the connection.")
                    {
                        FbConnection.ClearPool(fbcon);
                    }
                }
                finally
                {
                    InsertSQL.Dispose();
                    fbcon.Close();
                }
            }
        }

        public short testConfirmation(long idInf)
        {
            using (FbConnection fbco = newfbcon_co())
            {
                short test = 0;
                FbCommand SelectSQL = new FbCommand(String.Format($"select * from schedule where
schedid = '{idInf}' and calltype = '1'"), fbco);

                try
                {
                    fbco.Open();
                    FbTransaction fbt = fbco.BeginTransaction();
                    SelectSQL.Transaction = fbt;
                    FbDataReader reader = SelectSQL.ExecuteReader();
```

```
                    try
                    {
                        while (reader.Read())
                        {
                            test = 1;
                        }
                    }
                    catch (Exception ex)
                    {
                        Console.WriteLine($"MES_5 : {ex.Message}");
                        if (ex.Message == "Error reading data from the connection.")
                        {
                            FbConnection.ClearPool(fbco);
                        }
                    }
                    finally
                    {
                        reader.Close();
                    }
                }
                catch (Exception ex)
                {
                    Console.WriteLine($"15 : {ex.Message}");
                    if (ex.Message == "Error reading data from the connection.")
                    {
                        FbConnection.ClearPool(fbco);
                    }
                }
                finally
                {
                    fbco.Close();
                }

                return test;
            }
        }

        public TimeSpan timeSend1Day()//получение во сколько будут отправляться смс 2
        {
            using (FbConnection fbcon = newfbconn_con())
            {
                TimeSpan time = new TimeSpan();
                FbCommand SelectSQL = new FbCommand(String.Format("select timeSend1Day from
configurationcare"), fbcon);
                try
                {
                    fbcon.Open();
                    FbTransaction fbt = fbcon.BeginTransaction();
                    SelectSQL.Transaction = fbt;
                    FbDataReader reader = SelectSQL.ExecuteReader();

                    try
                    {
                        while (reader.Read())
                        {
                            time = TimeSpan.Parse(reader[0].ToString());
                        }
                    }
                    catch (Exception ex)
                    {
                        Console.WriteLine($"MES_6 : {ex.Message}");
                        if (ex.Message == "Error reading data from the connection.")
                        {
                            FbConnection.ClearPool(fbcon);
                        }
                    }
```

```
                finally
                {
                    reader.Close();
                }
            }
            catch (Exception ex)
            {
                Console.WriteLine($"16 : {ex.Message}");
                if (ex.Message == "Error reading data from the connection.")
                {
                    FbConnection.ClearPool(fbcon);
                }
            }
            finally
            {
                fbcon.Close();
            }

            return time;
        }
    }

    public int getStatusSendingSMS1() // Получаю статус запуска отправки смс 1
    {
        using (FbConnection fbcon = newfbconn_con())
        {
            int status = 0;
            FbCommand SelectSQL = new FbCommand(String.Format("SELECT statusSendingSMS1 FROM
configurationcare WHERE idConfiguration = 1"), fbcon);
            try
            {
                fbcon.Open();
                FbTransaction fbt = fbcon.BeginTransaction();
                SelectSQL.Transaction = fbt;
                FbDataReader reader = SelectSQL.ExecuteReader();

                try
                {
                    while (reader.Read())
                    {
                        status = reader.GetInt16(0);
                    }
                }
                catch (Exception ex)
                {
                    Console.WriteLine($"MES_6 : {ex.Message}");
                    if (ex.Message == "Error reading data from the connection.")
                    {
                        FbConnection.ClearPool(fbcon);
                    }
                }
                finally
                {
                    reader.Close();
                }
            }
            catch (Exception ex)
            {
                Console.WriteLine($"17 : {ex.Message}");
                if (ex.Message == "Error reading data from the connection.")
                {
                    FbConnection.ClearPool(fbcon);
                }
                status = 0;
            }
            Finally
```

```
                {
                    fbcon.Close();
                }

                return status;
            }
        }

        public int getStatusSendingSMS2() // Получаю статус запуска отправки смс 2
        {
            using (FbConnection fbcon = newfbconn_con())
            {
                int status = 0;
                FbCommand SelectSQL = new FbCommand(String.Format("SELECT statusSendingSMS2 FROM
configurationcare WHERE idConfiguration = 1"), fbcon);
                try
                {
                    fbcon.Open();
                    FbTransaction fbt = fbcon.BeginTransaction();
                    SelectSQL.Transaction = fbt;
                    FbDataReader reader = SelectSQL.ExecuteReader();

                    try
                    {
                        while (reader.Read())
                        {
                            status = reader.GetInt16(0);
                        }
                    }
                    catch (Exception ex)
                    {
                        Console.WriteLine($"MES_6 : {ex.Message}");
                        if (ex.Message == "Error reading data from the connection.")
                        {
                            FbConnection.ClearPool(fbcon);
                        }
                    }
                    finally
                    {
                        reader.Close();
                    }
                }
                catch (Exception ex)
                {
                    Console.WriteLine($"18 : {ex.Message}");
                    if (ex.Message == "Error reading data from the connection.")
                    {
                        FbConnection.ClearPool(fbcon);
                    }
                    status = 0;
                }
                finally
                {
                    fbcon.Close();
                }

                return status;
            }
        }

        public int getStatusSendingSMS3() // Получаю статус запуска отправки смс 3
        {
            using (FbConnection fbcon = newfbconn_con())
            {
                int status = 0;
```

```
                FbCommand SelectSQL = new FbCommand(String.Format("SELECT statusSendingSMS3 FROM
configurationcare WHERE idConfiguration = 1"), fbcon);
                try
                {
                    fbcon.Open();
                    FbTransaction fbt = fbcon.BeginTransaction();
                    SelectSQL.Transaction = fbt;
                    FbDataReader reader = SelectSQL.ExecuteReader();

                    try
                    {
                        while (reader.Read())
                        {
                            status = reader.GetInt16(0);
                        }
                    }
                    catch (Exception ex)
                    {
                        Console.WriteLine($"MES_6 : {ex.Message}");
                        if (ex.Message == "Error reading data from the connection.")
                        {
                            FbConnection.ClearPool(fbcon);
                        }
                    }
                    finally
                    {
                        reader.Close();
                    }
                }
                catch (Exception ex)
                {
                    Console.WriteLine($"19 : {ex.Message}");
                    if (ex.Message == "Error reading data from the connection.")
                    {
                        FbConnection.ClearPool(fbcon);
                    }
                    status = 0;
                }
                finally
                {
                    fbcon.Close();
                }

                return status;
            }
        }

        public string getText(int type)
        {
            using (FbConnection fbcon = newfbconn_con())
            {
                string str = null;
                FbCommand SelectSQL = new FbCommand(String.Format($"Select textSMS FROM
templatetextsms WHERE typeSMS = {type}"), fbcon);
                try
                {
                    fbcon.Open();
                    FbTransaction fbt = fbcon.BeginTransaction();
                    SelectSQL.Transaction = fbt;
                    FbDataReader reader = SelectSQL.ExecuteReader();

                    try
                    {
                        while (reader.Read())
                        {
                            str = reader.GetString(0);
```

```
                        }
                    }
                    catch (Exception ex)
                    {
                        Console.WriteLine($"MES_7 : {ex.Message}");
                        if (ex.Message == "Error reading data from the connection.")
                        {
                            FbConnection.ClearPool(fbcon);
                        }
                    }
                    finally
                    {
                        reader.Close();
                    }
                }
                catch (Exception ex)
                {
                    Console.WriteLine($"19 : {ex.Message}");
                    if (ex.Message == "Error reading data from the connection.")
                    {
                        FbConnection.ClearPool(fbcon);
                    }
                }
                finally
                {
                    fbcon.Close();
                }

                return str;
            }
        }

        public TemplateRecordShedule getFullName(TemplateRecordShedule record)
        {
            using (FbConnection fbco = newfbcon_co())
            {
                FbCommand SelectSQL = new FbCommand(String.Format($"select firstname from clients
where pcode = '{record.idClient}'"), fbco);

                try
                {
                    fbco.Open();
                    FbTransaction fbt = fbco.BeginTransaction();
                    SelectSQL.Transaction = fbt;
                    FbDataReader reader = SelectSQL.ExecuteReader();

                    try
                    {
                        while (reader.Read())
                        {
                            record.clientName = reader.GetString(0);
                        }
                    }
                    catch (Exception ex)
                    {
                        Console.WriteLine($"MES_8 : {ex.Message}");
                        if (ex.Message == "Error reading data from the connection.")
                        {
                            FbConnection.ClearPool(fbco);
                        }
                    }
                    finally
                    {
                        reader.Close();
                    }
                }
```

```
                catch (Exception ex)
                {
                    Console.WriteLine($"20 : {ex.Message}");
                    if (ex.Message == "Error reading data from the connection.")
                    {
                        FbConnection.ClearPool(fbco);
                    }
                }
                finally
                {
                    fbco.Close();
                }

                return record;
            }
        }

        public TemplateRecordShedule getPhoneAndIdDoctor(TemplateRecordShedule record)
        {
            using (FbConnection fbcon = newfbconn_con())
            {
                FbCommand SelectSQL = new FbCommand(String.Format($"Select phone, idDoctor FROM
scheduleofcare WHERE idInfodent = {record.idRecordInfodent}" +
                    $" and obsolete = '0'"), fbcon);

                try
                {
                    fbcon.Open();
                    FbTransaction fbt = fbcon.BeginTransaction();
                    SelectSQL.Transaction = fbt;
                    FbDataReader reader = SelectSQL.ExecuteReader();

                    try
                    {
                        while (reader.Read())
                        {
                            record.phoneNumber = reader[0].ToString();
                            record.idDoctor = reader.GetInt64(1);
                        }
                    }
                    catch (Exception ex)
                    {
                        Console.WriteLine($"MES_9 : {ex.Message}");
                        if (ex.Message == "Error reading data from the connection.")
                        {
                            FbConnection.ClearPool(fbcon);
                        }
                    }
                    finally
                    {
                        reader.Close();
                    }
                }
                catch (Exception ex)
                {
                    Console.WriteLine($"21 : {ex.Message}");
                    if (ex.Message == "Error reading data from the connection.")
                    {
                        FbConnection.ClearPool(fbcon);
                    }
                }
                finally
                {
                    fbcon.Close();
                }
```

```
                return record;
            }
        }

        public TemplateRecordShedule getIformationClient(long idInfodent)
        {
            using (FbConnection fbcon = newfbconn_con())
            {
                TemplateRecordShedule record = new TemplateRecordShedule();
                FbCommand SelectSQL = new FbCommand(String.Format($"Select  idFilial,  idClient,
dateOfReceipt FROM scheduleofmessages Where " +
                    $"idInfodent = '{idInfodent}' and obsolete = '0'"), fbcon);

                try
                {
                    fbcon.Open();
                    FbTransaction fbt = fbcon.BeginTransaction();
                    SelectSQL.Transaction = fbt;
                    FbDataReader reader = SelectSQL.ExecuteReader();

                    try
                    {
                        while (reader.Read())
                        {
                            record.idRecordInfodent = idInfodent;
                            record.filial = reader.GetInt64(0);
                            record.idClient = reader.GetInt64(1);
                            record.dateOfReceipt = reader.GetDateTime(2);
                        }
                    }
                    catch (Exception ex)
                    {
                        Console.WriteLine($"MES_10 : {ex.Message}");
                        if (ex.Message == "Error reading data from the connection.")
                        {
                            FbConnection.ClearPool(fbcon);
                        }
                    }
                    finally
                    {
                        reader.Close();
                    }
                }
                catch (Exception ex)
                {
                    Console.WriteLine($"22 : {ex.Message}");
                    if (ex.Message == "Error reading data from the connection.")
                    {
                        FbConnection.ClearPool(fbcon);
                    }
                }
                finally
                {
                    fbcon.Close();
                }

                return record;
            }
        }

        public void insertStatusSMS(string id, long idInf, short typeSms)
        {
            using (FbConnection fbcon = newfbconn_con())
            {
                string sqlstring = "";
```

```
                switch (typeSms)
                {
                    case 1:
                        sqlstring = String.Format($"UPDATE scheduleofmessages set idSMS1 = '{id}'
where idInfodent = '{idInf}' and obsolete = '0'");
                        break;
                    case 2:
                        sqlstring = String.Format($"UPDATE scheduleofmessages set idSMS2 = '{id}'
where idInfodent = '{idInf}' and obsolete = '0'");
                        break;
                    case 3:
                        sqlstring = String.Format($"UPDATE scheduleofmessages set idSMS3 = '{id}'
where idInfodent = '{idInf}' and obsolete = '0'");
                        break;
                }
                FbCommand InsertSQL = new FbCommand(sqlstring, fbcon);
                try
                {
                    fbcon.Open();
                    FbTransaction fbt = fbcon.BeginTransaction();
                    InsertSQL.Transaction = fbt;
                    int res = InsertSQL.ExecuteNonQuery();
                    fbt.Commit();
                }
                catch (Exception ex)
                {
                    Console.WriteLine($"23 : {ex.Message}");
                    if (ex.Message == "Error reading data from the connection.")
                    {
                        FbConnection.ClearPool(fbcon);
                    }
                }
                finally
                {
                    InsertSQL.Dispose();
                    fbcon.Close();
                }
            }
        }

        public void updateStatusSMS(string id, string status, short type)
        {
            using (FbConnection fbcon = newfbconn_con())
            {
                short totStatus = 0;
                switch (status)
                {
                    case "queued":
                        totStatus = 2;
                        break;
                    case "delivered":
                        totStatus = 1;
                        break;
                    case "delivery error":
                        totStatus = 3;
                        break;
                    case "smsc submit":
                        totStatus = 4;
                        break;
                    case "smsc reject":
                        totStatus = 5;
                        break;
                    case "incorrect id":
                        totStatus = 6;
                        break;
                }
```

```
                string sqlstring = "";
                switch (type)
                {
                    case 1:
                        sqlstring = String.Format($"UPDATE scheduleofmessages SET status1 =
'{totStatus}' WHERE idSMS1 = '{id}' and obsolete = '0'");
                        break;
                    case 2:
                        sqlstring = String.Format($"UPDATE scheduleofmessages SET status2 =
'{totStatus}' WHERE idSMS2 = '{id}' and obsolete = '0'");
                        break;
                    case 3:
                        sqlstring = String.Format($"UPDATE scheduleofmessages SET status3 =
'{totStatus}' WHERE idSMS3 = '{id}' and obsolete = '0'");
                        break;
                }
                FbCommand InsertSQL = new FbCommand(sqlstring, fbcon);
                try
                {
                    fbcon.Open();
                    FbTransaction fbt = fbcon.BeginTransaction();
                    InsertSQL.Transaction = fbt;
                    int res = InsertSQL.ExecuteNonQuery();
                    fbt.Commit();
                }
                catch (Exception ex)
                {
                    Console.WriteLine($"MES_11 : {ex.Message}");
                    if (ex.Message == "Error reading data from the connection.")
                    {
                        FbConnection.ClearPool(fbcon);
                    }
                }
                finally
                {
                    InsertSQL.Dispose();
                    fbcon.Close();
                }
            }
        }

        public void updateStatusSMSRu(string id, string status, short type)
        {
            using (FbConnection fbcon = newfbconn_con())
            {
                short totStatus = 0;

                switch (status)
                {
                    case "100":
                        totStatus = 2;
                        break;
                    case "101":
                        totStatus = 1;
                        break;
                    case "102":
                        totStatus = 3;
                        break;
                    case "103":
                        totStatus = 4;
                        break;
                    case "104":
                        totStatus = 5;
                        break;
                    case "105":
                        totStatus = 6;
```

```
                        break;
                    case "106":
                        totStatus = 7;
                        break;
                    case "107":
                        totStatus = 8;
                        break;
                    case "108":
                        totStatus = 9;
                        break;
                    case "110":
                        totStatus = 10;
                        break;

                }
                string sqlstring = "";
                switch (type)
                {
                    case 1:
                        sqlstring = $"UPDATE scheduleofmessages SET status1 = '{totStatus}' WHERE
idSMS1 = '{id}' and obsolete = '0'";
                        break;
                    case 2:
                        sqlstring = $"UPDATE scheduleofmessages SET status2 = '{totStatus}' WHERE
idSMS2 = '{id}' and obsolete = '0'";
                        break;
                    case 3:
                        sqlstring = $"UPDATE scheduleofmessages SET status3 = '{totStatus}' WHERE
idSMS3 = '{id}' and obsolete = '0'";
                        break;
                }

                FbCommand InsertSQL = new FbCommand(sqlstring, fbcon);
                try
                {
                    fbcon.Open();
                    FbTransaction fbt = fbcon.BeginTransaction();
                    InsertSQL.Transaction = fbt;
                    int res = InsertSQL.ExecuteNonQuery();
                    fbt.Commit();
                }
                catch (Exception ex)
                {
                    Console.WriteLine($"MES_11 : {ex.Message}");
                    if (ex.Message == "Error reading data from the connection.")
                    {
                        FbConnection.ClearPool(fbcon);
                    }
                }
                finally
                {
                    InsertSQL.Dispose();
                    fbcon.Close();
                }
            }
        }

        public List<string> getIdSms1()
        {
            using (FbConnection fbcon = newfbconn_con())
            {
                List<string> id = new List<string>();
                FbCommand SelectSQL = new FbCommand(String.Format($"select idSMS1 from
scheduleofmessages where status1 != '1' and idSMS1 != '' and obsolete = '0'"), fbcon);

                Try
```

```
                {
                    fbcon.Open();
                    FbTransaction fbt = fbcon.BeginTransaction();
                    SelectSQL.Transaction = fbt;
                    FbDataReader reader = SelectSQL.ExecuteReader();

                    try
                    {
                        while (reader.Read())
                        {
                            if (reader[0].ToString() != "") id.Add(reader.GetString(0));
                        }
                    }
                    catch (Exception ex)
                    {
                        Console.WriteLine($"MES_14 : {ex.Message}");
                        if (ex.Message == "Error reading data from the connection.")
                        {
                            FbConnection.ClearPool(fbcon);
                        }
                    }
                    finally
                    {
                        reader.Close();
                    }
                }
                catch (Exception ex)
                {
                    Console.WriteLine($"MES_13 : {ex.Message}");
                    if (ex.Message == "Error reading data from the connection.")
                    {
                        FbConnection.ClearPool(fbcon);
                    }
                }
                finally
                {
                    fbcon.Close();
                }

                return id;
            }
        }

        public List<string> getIdSms1Ru()
        {
            using (FbConnection fbcon = newfbconn_con())
            {
                List<string> id = new List<string>();
                FbCommand SelectSQL = new FbCommand(String.Format($"select idSMS1 from
scheduleofmessages where status1 != '4' and idSMS1 != '' and obsolete = '0'"), fbcon);

                try
                {
                    fbcon.Open();
                    FbTransaction fbt = fbcon.BeginTransaction();
                    SelectSQL.Transaction = fbt;
                    FbDataReader reader = SelectSQL.ExecuteReader();

                    try
                    {
                        while (reader.Read())
                        {
                            if (reader[0].ToString() != "") id.Add(reader.GetString(0));
                        }
                    }
                    catch (Exception ex)
```

```
                    {
                        Console.WriteLine($"MES_16 : {ex.Message}");
                        if (ex.Message == "Error reading data from the connection.")
                        {
                            FbConnection.ClearPool(fbcon);
                        }
                    }
                    finally
                    {
                        reader.Close();
                    }
                }
                catch (Exception ex)
                {
                    Console.WriteLine($"MES_15 : {ex.Message}");
                    if (ex.Message == "Error reading data from the connection.")
                    {
                        FbConnection.ClearPool(fbcon);
                    }
                }
                finally
                {
                    fbcon.Close();
                }

                return id;
            }
        }

        public List<string> getIdSms2()
        {
            using (FbConnection fbcon = newfbconn_con())
            {
                List<string> id = new List<string>();
                FbCommand SelectSQL = new FbCommand(String.Format($"select idSMS2 from
scheduleofmessages where status2 != '1' and idSMS2 != '' and obsolete = '0'"), fbcon);

                try
                {
                    fbcon.Open();
                    FbTransaction fbt = fbcon.BeginTransaction();
                    SelectSQL.Transaction = fbt;
                    FbDataReader reader = SelectSQL.ExecuteReader();

                    try
                    {
                        while (reader.Read())
                        {
                            if (reader[0].ToString() != "") id.Add(reader.GetString(0));
                        }
                    }
                    catch (Exception ex)
                    {
                        Console.WriteLine($"MES_18 : {ex.Message}");
                        if (ex.Message == "Error reading data from the connection.")
                        {
                            FbConnection.ClearPool(fbcon);
                        }
                    }
                    finally
                    {
                        reader.Close();
                    }
                }
                catch (Exception ex)
                {
```

```
                    Console.WriteLine($"MES_17 : {ex.Message}");
                    if (ex.Message == "Error reading data from the connection.")
                    {
                        FbConnection.ClearPool(fbcon);
                    }
                }
                finally
                {
                    fbcon.Close();
                }

                return id;
            }
        }

        public List<string> getIdSms2Ru()
        {
            using (FbConnection fbcon = newfbconn_con())
            {
                List<string> id = new List<string>();
                FbCommand SelectSQL = new FbCommand(String.Format($"select idSMS2 from
scheduleofmessages where status2 != '4' and idSMS2 != '' and obsolete = '0'"), fbcon);

                try
                {
                    fbcon.Open();
                    FbTransaction fbt = fbcon.BeginTransaction();
                    SelectSQL.Transaction = fbt;
                    FbDataReader reader = SelectSQL.ExecuteReader();

                    try
                    {
                        while (reader.Read())
                        {
                            if (reader[0].ToString() != "") id.Add(reader.GetString(0));
                        }
                    }
                    catch (Exception ex)
                    {
                        Console.WriteLine($"MES_20 : {ex.Message}");
                        if (ex.Message == "Error reading data from the connection.")
                        {
                            FbConnection.ClearPool(fbcon);
                        }
                    }
                    finally
                    {
                        reader.Close();
                    }
                }
                catch (Exception ex)
                {
                    Console.WriteLine($"MES_19 : {ex.Message}");
                    if (ex.Message == "Error reading data from the connection.")
                    {
                        FbConnection.ClearPool(fbcon);
                    }
                }
                finally
                {
                    fbcon.Close();
                }

                return id;
            }
        }
```

```
        public List<string> getIdSms3()
        {
            using (FbConnection fbcon = newfbconn_con())
            {
                List<string> id = new List<string>();
                FbCommand SelectSQL = new FbCommand(String.Format($"select idSMS3 from
scheduleofmessages where status3 != '1' and idSMS3 != '' and obsolete = '0'"), fbcon);

                try
                {
                    fbcon.Open();
                    FbTransaction fbt = fbcon.BeginTransaction();
                    SelectSQL.Transaction = fbt;
                    FbDataReader reader = SelectSQL.ExecuteReader();

                    try
                    {
                        while (reader.Read())
                        {
                            if (reader[0].ToString() != "") id.Add(reader.GetString(0));
                        }
                    }
                    catch (Exception ex)
                    {
                        Console.WriteLine($"MES_22 : {ex.Message}");
                        if (ex.Message == "Error reading data from the connection.")
                        {
                            FbConnection.ClearPool(fbcon);
                        }
                    }
                    finally
                    {
                        reader.Close();
                    }
                }
                catch (Exception ex)
                {
                    Console.WriteLine($"MES_21 : {ex.Message}");
                    if (ex.Message == "Error reading data from the connection.")
                    {
                        FbConnection.ClearPool(fbcon);
                    }
                }
                finally
                {
                    fbcon.Close();
                }

                return id;
            }
        }

        public List<string> getIdSms3Ru()
        {
            using (FbConnection fbcon = newfbconn_con())
            {
                List<string> id = new List<string>();
                FbCommand SelectSQL = new FbCommand(String.Format($"select idSMS3 from
scheduleofmessages where status3 != '4' and idSMS3 != '' and obsolete = '0'"), fbcon);

                try
                {
                    fbcon.Open();
                    FbTransaction fbt = fbcon.BeginTransaction();
                    SelectSQL.Transaction = fbt;
```

```
                    FbDataReader reader = SelectSQL.ExecuteReader();

                    try
                    {
                        while (reader.Read())
                        {
                            if (reader[0].ToString() != "") id.Add(reader.GetString(0));
                        }
                    }
                    catch (Exception ex)
                    {
                        Console.WriteLine($"MES_24 : {ex.Message}");
                        if (ex.Message == "Error reading data from the connection.")
                        {
                            FbConnection.ClearPool(fbcon);
                        }
                    }
                    finally
                    {
                        reader.Close();
                    }
                }
                catch (Exception ex)
                {
                    Console.WriteLine($"MES_23 : {ex.Message}");
                    if (ex.Message == "Error reading data from the connection.")
                    {
                        FbConnection.ClearPool(fbcon);
                    }
                }
                finally
                {
                    fbcon.Close();
                }

                return id;
            }
        }

        public string getAdressFilial(long idFilial)
        {
            using (FbConnection fbcon = newfbconn_con())
            {
                string adress = null;
                FbCommand SelectSQL = new FbCommand(String.Format($"Select adress FROM
adressfilial Where idFilial = '{idFilial}'"), fbcon);

                try
                {
                    fbcon.Open();
                    FbTransaction fbt = fbcon.BeginTransaction();
                    SelectSQL.Transaction = fbt;
                    FbDataReader reader = SelectSQL.ExecuteReader();

                    try
                    {
                        while (reader.Read())
                        {
                            adress = reader.GetString(0);
                        }
                    }
                    catch (Exception ex)
                    {
                        Console.WriteLine($"MES_25 : {ex.Message}");
                        if (ex.Message == "Error reading data from the connection.")
                        {
```

```
                        FbConnection.ClearPool(fbcon);
                    }
                }
                finally
                {
                    reader.Close();
                }
            }
            catch (Exception ex)
            {
                Console.WriteLine($"MES_25 : {ex.Message}");
                if (ex.Message == "Error reading data from the connection.")
                {
                    FbConnection.ClearPool(fbcon);
                }
            }
            finally
            {
                fbcon.Close();
            }

            return adress;
        }
    }



    public void createReport() // создание отчет Excel
    {
        List<reportModel> reports = getInformation();

        string[] columName = new[] { "IdInfodent", "Время добавления", "Дата приема", "Время
приема", "Филиал", "ИНП", "ФИО", "Доктор",
            "Телефон1", "Телефон2", "Телефон3", "Подтверждение", "Дата отправки СМС2", "Время
отправки СМС2", "Статус записи", "Статус СМС1", "Статус СМС2", "Статус СМС3","Статус СМС4" };

        string pathname = @"";//куда выгружать



        Excel.Application ex = new Microsoft.Office.Interop.Excel.Application();
        //Отобразить Excel
        //ex.Visible = true;
        //Количество листов в рабочей книге
        //ex.SheetsInNewWorkbook = 1;

        try
        {
            //Добавить рабочую книгу
            Excel.Workbook workBook = ex.Workbooks.Open(pathname, 0, false, 5, "", "", false,
Microsoft.Office.Interop.Excel.XlPlatform.xlWindows, "", true, false, 0, true, false, false);
            //Отключить отображение окон с сообщениями
            ex.DisplayAlerts = false;
            //Получаем первый лист документа (счет начинается с 1)
            Excel.Worksheet sheet = (Excel.Worksheet)ex.Worksheets.get_Item(1);
            //Название листа (вкладки снизу)
            sheet.Name = "Ежедневный отчёт " + DateTime.Now.ToShortDateString();

            Excel.Range r;
            sheet.Cells.ClearContents();

            //Объединение
            for (int i = 0; i < 15; i++)
            {
                r = sheet.Range[sheet.Cells[1, i + 1], sheet.Cells[2, i + 1]];
                r.Select();
```

```
    r.Merge(Type.Missing);
    r.Value = columName[i];
    r.Interior.Color = Color.Yellow;
}

Excel.Range rk = sheet.Range[sheet.Cells[1, 16], sheet.Cells[1, 19]];
rk.Select();
rk.Merge(Type.Missing);
rk.Value = "Статусы KitPoint";
rk.Interior.Color = Color.Yellow;

for (int i = 0; i < 4; i++)
{
    r = sheet.Range[sheet.Cells[2, i + 16], sheet.Cells[2, i + 16]];
    r.Select();
    r.Merge(Type.Missing);
    r.Value = columName[i + 15];
    r.Interior.Color = Color.Yellow;
}

Excel.Range rd = sheet.Range[sheet.Cells[1, 20], sheet.Cells[1, 23]];
rd.Select();
rd.Merge(Type.Missing);
rd.Value = "Статусы SMSRU";
rd.Interior.Color = Color.Yellow;

for (int i = 0; i < 4; i++)
{
    r = sheet.Range[sheet.Cells[2, i + 20], sheet.Cells[2, i + 20]];
    r.Select();
    r.Merge(Type.Missing);
    r.Value = columName[i + 15];
    r.Interior.Color = Color.Yellow;
}

int j = 0;
foreach (var record in reports)
{
    for (int i = 0; i < 23; i++)
    {
        r = sheet.Range[sheet.Cells[j + 3, i + 1], sheet.Cells[j + 3, i + 1]];
        r.Select();
        switch (i)
        {
            case 0:
                r.Value = record.schedid;
                break;
            case 1:
                r.Value = record.dateAdd.ToString();
                break;
            case 2:
                r.Value = record.dateOfReceipt.ToShortDateString();
                break;
            case 3:
                r.Value = record.dateOfReceipt.ToShortTimeString(); ;
                break;
            case 4:
                r.Value = record.filial;
                break;
            case 5:
                r.Value = record.histnum;
                break;
            case 6:
                r.Value = record.fullname;
                break;
            case 7:
```

```
                                r.Value = record.dname;
                                break;
                            case 8:
                                r.Value = record.phone1;
                                break;
                            case 9:
                                r.Value = record.phone2;
                                break;
                            case 10:
                                r.Value = record.phone3;
                                break;
                            case 11:
                                r.Value = record.theConfirmation;
                                break;
                            case 12:
                                try
                                {
                                    if (r.Value != null) r.Value =
DateTime.Parse(record.timeSendMessage2Total).ToShortDateString();
                                }
                                catch (Exception exw)
                                {
                                    Console.WriteLine($"24 : {exw.Message}");
                                }
                                break;
                            case 13:
                                try
                                {
                                    if (r.Value != null) r.Value =
DateTime.Parse(record.timeSendMessage2Total).ToShortTimeString();
                                }
                                catch (Exception exw)
                                {
                                    Console.WriteLine($"25 : {exw.Message}");
                                }
                                break;
                            case 14:
                                r.Value = record.obsolete;
                                break;
                            case 15:
                                //смс визитка
                                break;
                            case 16:
                                r.Value = getKitStatusSms(record.kitStatusSms1);
                                break;
                            case 17:
                                r.Value = getKitStatusSms(record.kitStatusSms2);
                                break;
                            case 18:
                                r.Value = getKitStatusSms(record.kitStatusSms3);
                                break;
                            case 19:
                                //смс визитка
                                break;
                            case 20:
                                r.Value = getRuStatusSms(record.RuStatusSms1);
                                break;
                            case 21:
                                r.Value = getRuStatusSms(record.RuStatusSms2);
                                break;
                            case 22:
                                r.Value = getRuStatusSms(record.RuStatusSms3);
                                break;
                        }
                    }
                    j++;
```

```
                }

                sheet.Columns["A:W"].AutoFit();

                r = sheet.Range[sheet.Cells[1, 1], sheet.Cells[2, 23]];
                r.Borders.ColorIndex = 0;
                r.Borders.LineStyle = Excel.XlLineStyle.xlContinuous;
                r.Borders.Weight = Excel.XlBorderWeight.xlThin;

                workBook.SaveAs(pathname);
            }
            catch (Exception err)
            {
                Console.WriteLine($"MES_25 : Ошибка при создании отчёта : {err.Message}");
            }
            finally
            {
                ex.Quit();
                System.Runtime.InteropServices.Marshal.FinalReleaseComObject(ex);
            }
        }

        public string getRuStatusSms(short ruStatus)
        {
            string status = "";

            switch (ruStatus)
            {
                case 2: status = "Сообщение в очереди"; break;
                case 1: status = "Сообщение перед. оператору"; break;
                case 3: status = "Сообщение отправлено"; break;
                case 4: status = "Сообщение доставлено"; break;
                case 5: status = "Не доставлено, время жизни истекло"; break;
                case 6: status = "Не доставлено, удалено оператором"; break;
                case 7: status = "Не доставлено, сбой в телефоне"; break;
                case 8: status = "Не доставлено, неизветсная причина"; break;
                case 9: status = "Не доставлено, отклонено"; break;
                case 10: status = "Сообщение прочитано"; break;
            }

            return status;
        }

        public string getKitStatusSms(short ruStatus)
        {
            string status = "";

            switch (ruStatus)
            {
                case 0: status = "Не отправлено"; break;
                case 1: status = "Отправлено"; break;
                case 2: status = "Дата приема на след. день, после записи"; break;
                case 3: status = "Прием в день назначения"; break;
                case 4: status = "Прием был ранее подтвержден"; break;
            }

            return status;
        }

        public List<reportModel> getInformation()
        {
            using (FbConnection fbcon = newfbconn_con())
            {
                List<reportModel> information = new List<reportModel>();
                FbCommand SelectSQL = new FbCommand(String.Format($"select idInfodent, idFilial,
dateOfReceipt, idClient, theConfirmation, timeSendMessage2Total, wasItSent1, wasItSent2,
```

```
wasItSent3, obsolete, status1, status2, status3, idDoctor, dateAdd  FROM scheduleofmessages where
date(timeSendMessage2) = date(now())"));
                try
                {
                    fbcon.Open();
                    FbTransaction fbt = fbcon.BeginTransaction();
                    SelectSQL.Transaction = fbt;
                    FbDataReader reader = SelectSQL.ExecuteReader();
                    try
                    {
                        while (reader.Read())
                        {
                            reportModel report = new reportModel();
                            report.schedid = reader.GetInt64(0);
                            report.idFilial = reader.GetInt64(1);
                            report.dateOfReceipt = reader.GetDateTime(2);
                            report.idClient = reader.GetInt64(3);
                            if (reader[4].ToString() != "") report.theConfirmation =
reader.GetInt16(4); else report.theConfirmation = -1;
                            if (reader[5].ToString() != "") report.timeSendMessage2Total =
reader.GetString(5);
                            report.kitStatusSms1 = reader.GetInt16(6);
                            report.kitStatusSms2 = reader.GetInt16(7);
                            report.kitStatusSms3 = reader.GetInt16(8);
                            report.obsolete = reader.GetInt16(9);
                            report.RuStatusSms1 = reader.GetInt16(10);
                            report.RuStatusSms2 = reader.GetInt16(11);
                            report.RuStatusSms3 = reader.GetInt16(12);
                            report.idDoctor = reader.GetInt64(13);
                            report.dateAdd = reader.GetDateTime(14);

                            information.Add(report);
                        }
                    }
                    catch (Exception ex)
                    {
                        Console.WriteLine($"MES_25 :  {ex.Message}");
                        if (ex.Message == "Error reading data from the connection.")
                        {
                            FbConnection.ClearPool(fbcon);
                        }
                    }
                    finally
                    {
                        reader.Close();
                    }
                }
                catch (Exception ex)
                {
                    Console.WriteLine($"26 : {ex.Message}");
                    if (ex.Message == "Error reading data from the connection.")
                    {
                        FbConnection.ClearPool(fbcon);
                    }
                }
                finally
                {
                    fbcon.Close();
                }
                if (information.Count != 0)
                {
                    foreach (var report in information)
                    {
                        report.filial = getFilial(report.idFilial);
                        report.histnum = getHistnum(report.idClient);
                        report.fullname = getFullname(report.idClient);
                        report.dname = getDname(report.idDoctor);
```

```
                    report.phone1 = getPhone1(report.idClient);
                    report.phone2 = getPhone2(report.idClient);
                    report.phone3 = getPhone3(report.idClient);
                }
            }

            return information;
        }
    }

    public string getPhone1(long idClient)
    {
        using (FbConnection fbco = newfbcon_co())
        {
            string phone1 = null;
            FbCommand SelectSQL = new FbCommand(String.Format($"select phone1 from clients
where pcode = '{idClient}'"), fbco);

            try
            {
                fbco.Open();
                FbTransaction fbt = fbco.BeginTransaction();
                SelectSQL.Transaction = fbt;
                FbDataReader reader = SelectSQL.ExecuteReader();

                try
                {
                    while (reader.Read())
                    {
                        phone1 = reader.GetString(0);
                    }
                }
                catch (Exception ex)
                {
                    Console.WriteLine($"MES_26 :  {ex.Message}");
                    if (ex.Message == "Error reading data from the connection.")
                    {
                        FbConnection.ClearPool(fbco);
                    }
                }
                finally
                {
                    reader.Close();
                }
            }
            catch (Exception ex)
            {
                Console.WriteLine($"27 : {ex.Message}");
                if (ex.Message == "Error reading data from the connection.")
                {
                    FbConnection.ClearPool(fbco);
                }
            }
            finally
            {
                fbco.Close();
            }

            return phone1;
        }
    }

    public string getPhone2(long idClient)
    {
        using (FbConnection fbco = newfbcon_co())
        {
```

```
                string phone2 = null;
                FbCommand SelectSQL = new FbCommand(String.Format($"select phone2 from clients
where pcode = '{idClient}'"), fbco);

                try
                {
                    fbco.Open();
                    FbTransaction fbt = fbco.BeginTransaction();
                    SelectSQL.Transaction = fbt;
                    FbDataReader reader = SelectSQL.ExecuteReader();

                    try
                    {
                        while (reader.Read())
                        {
                            phone2 = reader.GetString(0);
                        }
                    }
                    catch (Exception ex)
                    {
                        Console.WriteLine($"MES_27 :  {ex.Message}");
                        if (ex.Message == "Error reading data from the connection.")
                        {
                            FbConnection.ClearPool(fbco);
                        }
                    }
                    finally
                    {
                        reader.Close();
                    }
                }
                catch (Exception ex)
                {
                    Console.WriteLine($"28 : {ex.Message}");
                    if (ex.Message == "Error reading data from the connection.")
                    {
                        FbConnection.ClearPool(fbco);
                    }
                }
                finally
                {
                    fbco.Close();
                }

                return phone2;
            }
        }

        public string getPhone3(long idClient)
        {
            using (FbConnection fbco = newfbcon_co())
            {
                string phone3 = null;
                FbCommand SelectSQL = new FbCommand(String.Format($"select phone3 from clients
where pcode = '{idClient}'"), fbco);

                try
                {
                    fbco.Open();
                    FbTransaction fbt = fbco.BeginTransaction();
                    SelectSQL.Transaction = fbt;
                    FbDataReader reader = SelectSQL.ExecuteReader();

                    try
                    {
                        while (reader.Read())
```

```
                        {
                            phone3 = reader.GetString(0);
                        }
                    }
                    catch (Exception ex)
                    {
                        Console.WriteLine($"MES_28 :  {ex.Message}");
                        if (ex.Message == "Error reading data from the connection.")
                        {
                            FbConnection.ClearPool(fbco);
                        }
                    }
                    finally
                    {
                        reader.Close();
                    }
                }
                catch (Exception ex)
                {
                    Console.WriteLine($"29 : {ex.Message}");
                    if (ex.Message == "Error reading data from the connection.")
                    {
                        FbConnection.ClearPool(fbco);
                    }
                }
                finally
                {
                    fbco.Close();
                }

                return phone3;
            }
        }

        public string getFullname(long idClient)
        {
            using (FbConnection fbco = newfbcon_co())
            {
                string fullname = null;
                FbCommand SelectSQL = new FbCommand(String.Format($"select fullname from clients
where pcode = '{idClient}'"), fbco);

                try
                {
                    fbco.Open();
                    FbTransaction fbt = fbco.BeginTransaction();
                    SelectSQL.Transaction = fbt;
                    FbDataReader reader = SelectSQL.ExecuteReader();

                    try
                    {
                        while (reader.Read())
                        {
                            fullname = reader.GetString(0);
                        }
                    }
                    catch (Exception ex)
                    {
                        Console.WriteLine($"MES_29 :  {ex.Message}");
                        if (ex.Message == "Error reading data from the connection.")
                        {
                            FbConnection.ClearPool(fbco);
                        }
                    }
                    finally
                    {
```

```
                    reader.Close();
                }
            }
            catch (Exception ex)
            {
                Console.WriteLine($"30 : {ex.Message}");
                if (ex.Message == "Error reading data from the connection.")
                {
                    FbConnection.ClearPool(fbco);
                }
            }
            finally
            {
                fbco.Close();
            }

            return fullname;
        }
    }

    public string getHistnum(long idClient)
    {
        using (FbConnection fbco = newfbcon_co())
        {
            string histnum = null;
            FbCommand SelectSQL = new FbCommand(String.Format($"select histnum from clients
where pcode = '{idClient}'"), fbco);

            try
            {
                fbco.Open();
                FbTransaction fbt = fbco.BeginTransaction();
                SelectSQL.Transaction = fbt;
                FbDataReader reader = SelectSQL.ExecuteReader();

                try
                {
                    while (reader.Read())
                    {
                        histnum = reader.GetString(0);
                    }
                }
                catch (Exception ex)
                {
                    Console.WriteLine($"MES_30 :  {ex.Message}");
                    if (ex.Message == "Error reading data from the connection.")
                    {
                        FbConnection.ClearPool(fbco);
                    }
                }
                finally
                {
                    reader.Close();
                }
            }
            catch (Exception ex)
            {
                Console.WriteLine($"31 : {ex.Message}");
                if (ex.Message == "Error reading data from the connection.")
                {
                    FbConnection.ClearPool(fbco);
                }
            }
            finally
            {
                fbco.Close();
```

```
                }

                return histnum;
            }
        }

        public string getFilial(long idFilial)
        {
            using (FbConnection fbco = newfbcon_co())
            {
                string filial = null;
                FbCommand SelectSQL = new FbCommand(String.Format($"select shortname from filials
where filid = '{idFilial}'"), fbco);

                try
                {
                    fbco.Open();
                    FbTransaction fbt = fbco.BeginTransaction();
                    SelectSQL.Transaction = fbt;
                    FbDataReader reader = SelectSQL.ExecuteReader();

                    try
                    {
                        while (reader.Read())
                        {
                            filial = reader.GetString(0);
                        }
                    }
                    catch (Exception ex)
                    {
                        Console.WriteLine($"MES_32 :  {ex.Message}");
                        if (ex.Message == "Error reading data from the connection.")
                        {
                            FbConnection.ClearPool(fbco);
                        }
                    }
                    finally
                    {
                        reader.Close();
                    }
                }
                catch (Exception ex)
                {
                    Console.WriteLine($"32 : {ex.Message}");
                    if (ex.Message == "Error reading data from the connection.")
                    {
                        FbConnection.ClearPool(fbco);
                    }
                }
                finally
                {
                    fbco.Close();
                }

                return filial;
            }
        }

        public List<testSendSmsModel> selectStatusSendSms()
        {
            using (FbConnection fbcon = newfbconn_con())
            {
                List<testSendSmsModel> list = new List<testSendSmsModel>();
                FbCommand SelectSQL = new FbCommand(String.Format($"select idSms, typeSms, idInf
from teststatussendsms where processed = '0' " +
```

```
                    $"and time(timeTest) between time(now()) and time(DATE_ADD(NOW(),Interval 1
MINUTE))"), fbcon);
                try
                {
                    fbcon.Open();
                    FbTransaction fbt = fbcon.BeginTransaction();
                    SelectSQL.Transaction = fbt;
                    FbDataReader reader = SelectSQL.ExecuteReader();

                    try
                    {
                        while (reader.Read())
                        {
                            testSendSmsModel model = new testSendSmsModel();
                            model.idSms = reader.GetString(0);
                            model.typeSms = reader.GetInt16(1);
                            model.idInf = reader.GetInt64(2);
                            list.Add(model);
                        }
                    }
                    catch (Exception ex)
                    {
                        Console.WriteLine($"MES_34 :  {ex.Message}");
                        if (ex.Message == "Error reading data from the connection.")
                        {
                            FbConnection.ClearPool(fbcon);
                        }
                    }
                    finally
                    {
                        reader.Close();
                    }
                }
                catch (Exception ex)
                {
                    Console.WriteLine($"MES_33 :  {ex.Message}");
                    if (ex.Message == "Error reading data from the connection.")
                    {
                        FbConnection.ClearPool(fbcon);
                    }
                }
                finally
                {
                    fbcon.Close();
                }

                return list;
            }
        }

        public void getStatusSmsRu(string id, short typeSms, long idInf) // получения статуса sms
        {
            try
            {
                ConnectionDB db = new ConnectionDB();
                System.Net.HttpWebRequest reqGET =
(System.Net.HttpWebRequest)System.Net.WebRequest.Create("" + id + "&json=1");

                WebProxy myProxy = new WebProxy();
                myProxy.Credentials = new NetworkCredential("", "");
                reqGET.Proxy = HttpWebRequest.DefaultWebProxy;
                reqGET.Proxy.Credentials = CredentialCache.DefaultNetworkCredentials;
                reqGET.PreAuthenticate = true;
                reqGET.Proxy = myProxy;
```

```
                var resp = (HttpWebResponse)reqGET.GetResponse();
                using (var sr = new StreamReader(resp.GetResponseStream()))

                {
                    string respons = sr.ReadToEnd();
                    var obj = JsonConvert.DeserializeObject<RootObject>(respons);
                    string idStr = obj.sms[id].status_code;

                    if (idStr != null)
                    {
                        switch (idStr)
                        {
                            case "100":
                                idStr = "Сообщ. в очер.";
                                break;
                            case "101":
                                idStr = "Сообщ. перед. опер.";
                                break;
                            case "102":
                                idStr = "Сообщ. отправ.";
                                break;
                            case "103":
                                idStr = "Сообщ. достав.";
                                break;
                            case "104":
                                idStr = "Не дост.: время жизни истек.";
                                break;
                            case "105":
                                idStr = "Не дост.: удал. операт.";
                                break;
                            case "106":
                                idStr = "Не дост.: сбои в тел.";
                                break;
                            case "107":
                                idStr = "Не дост.: неизв. прич.";
                                break;
                            case "108":
                                idStr = "Не дост.: откл.";
                                break;
                            case "110":
                                idStr = "Сообщ. прочит.";
                                break;
                        }

                        addCommentStatusSms(idStr, typeSms, idInf);
                    }
                }
            }
            catch (Exception ex)
            {
                Console.WriteLine($"33 : {ex.Message}");
            }
        }

        public void addCommentStatusSms(string status, short typeSms, long idInf)
        {
            using (FbConnection fbww001 = newfbww001_con())
            using (FbConnection fbww002 = newfbww002_con())
            using (FbConnection fbww003 = newfbww003_con())
            using (FbConnection fbww005 = newfbww005_con())
            {
                long idFilial = selectIdFilialRecord(idInf);
                typeSms++;
                string sqlstring = String.Format($"UPDATE schedule set comment = iif(comment is
null, 'KP{typeSms}:{status}', comment || '-' || 'KP{typeSms}:{status}') where schedid =
'{idInf}'");
                FbTransaction fbt = null;
```

```
switch (idFilial)

{
    case 1:
        FbCommand InsertSQL =
           new FbCommand(sqlstring, fbww001);

        try
        {
            fbww001.Open();
            fbt = fbww001.BeginTransaction();
            InsertSQL.Transaction = fbt;
            int res = InsertSQL.ExecuteNonQuery();
            fbt.Commit();
        }
        catch (Exception ex)
        {
            Console.WriteLine($"34 : {idInf} {ex.Message}");
            if (ex.Message == "Error reading data from the connection.")
            {
                FbConnection.ClearPool(fbww001);
            }
        }
        finally
        {
            InsertSQL.Dispose();
            fbww001.Close();
        }
        break;

    case 2:
        InsertSQL =
           new FbCommand(sqlstring, fbww002);// Добавить аргументы

        try
        {
            fbww002.Open();
            fbt = fbww002.BeginTransaction();
            InsertSQL.Transaction = fbt;
            int res = InsertSQL.ExecuteNonQuery();
            fbt.Commit();
        }
        catch (Exception ex)
        {
            Console.WriteLine($"35 :  {idInf} {ex.Message}");
            if (ex.Message == "Error reading data from the connection.")
            {
                FbConnection.ClearPool(fbww002);
            }
        }
        finally
        {
            InsertSQL.Dispose();
            fbww002.Close();
        }
        break;

    case 3:
        InsertSQL =
           new FbCommand(sqlstring, fbww003);

        try
        {
            fbww003.Open();
            fbt = fbww003.BeginTransaction();
            InsertSQL.Transaction = fbt;
            int res = InsertSQL.ExecuteNonQuery();
```

```
                    fbt.Commit();

                }
                catch (Exception ex)
                {
                    Console.WriteLine($"36 :  {idInf} {ex.Message}");
                    if (ex.Message == "Error reading data from the connection.")
                    {
                        FbConnection.ClearPool(fbww003);
                    }
                }
                finally
                {
                    InsertSQL.Dispose();
                    fbww003.Close();
                }
                break;

            case 5:
                InsertSQL =
                   new FbCommand(sqlstring, fbww005);

                try
                {
                    fbww005.Open();
                    fbt = fbww005.BeginTransaction();
                    InsertSQL.Transaction = fbt;
                    int res = InsertSQL.ExecuteNonQuery();
                    fbt.Commit();
                }
                catch (Exception ex)
                {
                    Console.WriteLine($"37 :  {idInf} {ex.Message}");
                    if (ex.Message == "Error reading data from the connection.")
                    {
                        FbConnection.ClearPool(fbww005);
                    }
                }
                finally
                {
                    InsertSQL.Dispose();
                    fbww005.Close();
                }
                break;
            }
        }
    }

    public long selectIdFilialRecord(long code)
    {
        using (FbConnection fbco = newfbcon_co())
        {
            long idFilial = 0;
            FbCommand SelectSQL = new FbCommand(String.Format("select filial from schedule
where schedid = '{0}'", code), fbco);

            try
            {
                fbco.Open();
                FbTransaction fbt = fbco.BeginTransaction();
                SelectSQL.Transaction = fbt;
                FbDataReader reader = SelectSQL.ExecuteReader();

                try
                {
                    while (reader.Read())
                    {
```

```
                        idFilial = reader.GetInt64(0);

                    }
                }
                catch (Exception ex)
                {
                    Console.WriteLine($"MES_35 :  {ex.Message}");
                    if (ex.Message == "Error reading data from the connection.")
                    {
                        FbConnection.ClearPool(fbco);
                    }
                }
                finally
                {
                    reader.Close();
                }
            }
            catch (Exception ex)
            {
                Console.WriteLine($"38 : {ex.Message}");
                if (ex.Message == "Error reading data from the connection.")
                {
                    FbConnection.ClearPool(fbco);
                }
            }
            finally
            {
                fbco.Close();
            }

            return idFilial;
        }
    } // получение idFilial по записи в расписании

    public void updateTestSendSms(string idSms)
    {
        using (FbConnection fbcon = newfbconn_con())
        {
            string sqlstring = String.Format($"UPDATE teststatussendsms SET processed = '1'
WHERE idSms = '{idSms}'");
            FbCommand InsertSQL =
               new FbCommand(sqlstring, fbcon);// Добавить аргументы
            try
            {
                fbcon.Open();
                FbTransaction fbt = fbcon.BeginTransaction();
                InsertSQL.Transaction = fbt;
                int res = InsertSQL.ExecuteNonQuery();
                fbt.Commit();
            }
            catch (Exception ex)
            {
                Console.WriteLine($"39 : {ex.Message}");
                if (ex.Message == "Error reading data from the connection.")
                {
                    FbConnection.ClearPool(fbcon);
                }
            }
            finally
            {
                InsertSQL.Dispose();
                fbcon.Close();
            }

        }
    }
```

```
        public double getBalanseRu() // получение баланса

        {
            double totlaBalanse = 0;
            System.Net.HttpWebRequest reqGET =
(System.Net.HttpWebRequest)System.Net.WebRequest.Create("");
            try
            {
                WebProxy myProxy = new WebProxy();
                myProxy.Credentials = new NetworkCredential("", "");
                reqGET.Proxy = HttpWebRequest.DefaultWebProxy;
                reqGET.Proxy.Credentials = CredentialCache.DefaultNetworkCredentials;
                reqGET.PreAuthenticate = true;
                reqGET.Proxy = myProxy;
                var resp = (HttpWebResponse)reqGET.GetResponse();
                using (var sr = new StreamReader(resp.GetResponseStream()))
                {
                    var result = sr.ReadToEnd();
                    string pattern1 = "\"balance\": ([0-9]+.[0-9]+)";
                    Regex regex = new Regex(pattern1);
                    Match balanse = regex.Match(result);
                    while (balanse.Success)
                    {
                        string temp = balanse.Groups[1].Value;
                        temp = temp.Replace('.', ',');
                        totlaBalanse = Convert.ToDouble(temp);
                        balanse = balanse.NextMatch();
                    }
                }
            }
            catch (Exception ex)
            {
                Console.WriteLine($"88 : {ex.Message}");
            }

            return totlaBalanse;
        }
    }
}
```

ПРИЛОЖЕНИЕ И

ИСХОДНЫЙ КОД CRM. PROGRAM

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace CRM
{
    static class Program
    {
        /// <summary>
        /// Главная точка входа для приложения.
        /// </summary>
        [STAThread]
        static void Main()
        {
            Application.EnableVisualStyles();
            Application.SetCompatibleTextRenderingDefault(false);
            Application.Run(new Authorization());
        }
    }
}
```

ПРИЛОЖЕНИЕ К

ИСХОДНЫЙ КОД CRM.CONNECTIONDB

```
using System;
using System.Collections.Generic;
using MySql.Data.MySqlClient;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using Excel = Microsoft.Office.Interop.Excel;
using CRM.Model;
using FirebirdSql.Data.FirebirdClient;
using System.Drawing;
using Microsoft.Office.Interop.Excel;

namespace CRM
{
    class ConnectionDB
    {

        public ConnectionDB()//подключение к базе
        {

        }

        public FbConnection newfbcon_co()
        {
            FbConnection fbco;
            FbConnectionStringBuilder fbco_con = new FbConnectionStringBuilder();
            //fbco_con.Charset = "WIN1251";
            fbco_con.UserID = "";
            fbco_con.Password = "";
            fbco_con.Database = "";
            fbco_con.ServerType = 0;
            fbco_con.ConnectionLifeTime = 60;
            fbco = new FbConnection(fbco_con.ToString());

            return fbco;
        }

        public FbConnection newfbconn_con()
        {
            FbConnection fbconn;
            FbConnectionStringBuilder fbconn_con = new FbConnectionStringBuilder();
            //fbco_con.Charset = "WIN1251";
            fbconn_con.UserID = "";
            fbconn_con.Password = "";
            fbconn_con.Database = "";
            fbconn_con.ServerType = 0;
            fbconn_con.ConnectionLifeTime = 60;
            fbconn = new FbConnection(fbconn_con.ToString());

            return fbconn;
        }
        public int getStatusPowerInputProcessing() // Получаю статус запуска обработки новых
приемов
        {
            using (FbConnection fbcon = newfbconn_con())
            {
                int status = 0;
                FbCommand SelectSQL = new FbCommand(String.Format("SELECT PowerInputProcessing
FROM configurationcare WHERE idConfiguration = 1"), fbcon);

                try
                {
                    fbcon.Open();
                    FbTransaction fbt = fbcon.BeginTransaction();
```

```
                    SelectSQL.Transaction = fbt;
                    FbDataReader reader = SelectSQL.ExecuteReader();

                    try
                    {
                        while (reader.Read())
                        {
                            status = reader.GetInt16(0);
                        }
                    }
                    catch (Exception ex)
                    {
                        Console.WriteLine($"CRM_1 :  {ex.Message}");
                        if (ex.Message == "Error reading data from the connection.")
                        {
                            FbConnection.ClearPool(fbcon);
                        }
                    }
                    finally
                    {
                        reader.Close();
                    }
                }
                catch (Exception ex)
                {
                    Console.WriteLine($"1 : {ex.Message}");
                    if (ex.Message == "Error reading data from the connection.")
                    {
                        FbConnection.ClearPool(fbcon);
                    }
                }
                finally
                {
                    fbcon.Close();
                }

                return status;
            }
        }

        public void UpdateStatusCase(int numCase, int status)  // Обновляю статус запуска
обработки новых приемов
        {
            using (FbConnection fbcon = newfbconn_con())
            {
                string sqlstring = "";
                switch (numCase)
                {
                    case 1:
                        sqlstring = String.Format($"UPDATE configurationcare SET
PowerInputProcessing='{status}' WHERE idConfiguration='1';");
                        break;
                    case 2:
                        sqlstring = String.Format($"UPDATE configurationcare SET
statusSendingSMS1='{status}' WHERE idConfiguration='1';");
                        break;
                    case 3:
                        sqlstring = String.Format($"UPDATE configurationcare SET
statusSendingSMS2='{status}' WHERE idConfiguration='1';");
                        break;
                    case 4:
                        sqlstring = String.Format($"UPDATE configurationcare SET
statusSendingSMS3='{status}' WHERE idConfiguration='1';");
                        break;
                }
```

```
                FbCommand InsertSQL = new FbCommand(sqlstring, fbcon);// Добавить аргументы
                try
                {
                    fbcon.Open();
                    FbTransaction fbt = fbcon.BeginTransaction();
                    InsertSQL.Transaction = fbt;
                    int res = InsertSQL.ExecuteNonQuery();
                    fbt.Commit();
                }
                catch (Exception ex)
                {
                    Console.WriteLine($"CRM_2 : {ex.Message}");
                    if (ex.Message == "Error reading data from the connection.")
                    {
                        FbConnection.ClearPool(fbcon);
                    }
                }
                finally
                {
                    InsertSQL.Dispose();
                    fbcon.Close();
                }
            }
        }

        public int getStatusSendingSMS1() // Получаю статус запуска отправки смс 1
        {
            using (FbConnection fbcon = newfbconn_con())
            {
                int status = 0;
                FbCommand SelectSQL = new FbCommand(String.Format("SELECT statusSendingSMS1 FROM
configurationcare WHERE idConfiguration = 1"), fbcon);

                try
                {
                    fbcon.Open();
                    FbTransaction fbt = fbcon.BeginTransaction();
                    SelectSQL.Transaction = fbt;
                    FbDataReader reader = SelectSQL.ExecuteReader();

                    try
                    {
                        while (reader.Read())
                        {
                            status = reader.GetInt16(0);
                        }
                    }
                    catch (Exception ex)
                    {
                        Console.WriteLine($"CRM_3 :  {ex.Message}");
                        if (ex.Message == "Error reading data from the connection.")
                        {
                            FbConnection.ClearPool(fbcon);
                        }
                    }
                    finally
                    {
                        reader.Close();
                    }
                }
                catch (Exception ex)
                {
                    Console.WriteLine($"3 : {ex.Message}");
                    if (ex.Message == "Error reading data from the connection.")
                    {
                        FbConnection.ClearPool(fbcon);
```

```
                }
            }
            finally
            {
                fbcon.Close();
            }

            return status;
        }
    }

    public int getStatusSendingSMS2() // Получаю статус запуска отправки смс 2
    {
        using (FbConnection fbcon = newfbconn_con())
        {
            int status = 0;
            FbCommand SelectSQL = new FbCommand(String.Format("SELECT statusSendingSMS2 FROM
configurationcare WHERE idConfiguration = 1"), fbcon);

            try
            {
                fbcon.Open();
                FbTransaction fbt = fbcon.BeginTransaction();
                SelectSQL.Transaction = fbt;
                FbDataReader reader = SelectSQL.ExecuteReader();

                try
                {
                    while (reader.Read())
                    {
                        status = reader.GetInt16(0);
                    }
                }
                catch (Exception ex)
                {
                    Console.WriteLine($"CRM_4 :  {ex.Message}");
                    if (ex.Message == "Error reading data from the connection.")
                    {
                        FbConnection.ClearPool(fbcon);
                    }
                }
                finally
                {
                    reader.Close();
                }
            }
            catch (Exception ex)
            {
                Console.WriteLine($"4 : {ex.Message}");
                if (ex.Message == "Error reading data from the connection.")
                {
                    FbConnection.ClearPool(fbcon);
                }
            }
            finally
            {
                fbcon.Close();
            }

            return status;
        }
    }

    public int getStatusSendingSMS3() // Получаю статус запуска отправки смс 3
    {
        using (FbConnection fbcon = newfbconn_con())
```

```
            {
                int status = 0;
                FbCommand SelectSQL = new FbCommand(String.Format("SELECT statusSendingSMS3 FROM
configurationcare WHERE idConfiguration = 1"), fbcon);

                try
                {
                    fbcon.Open();
                    FbTransaction fbt = fbcon.BeginTransaction();
                    SelectSQL.Transaction = fbt;
                    FbDataReader reader = SelectSQL.ExecuteReader();

                    try
                    {
                        while (reader.Read())
                        {
                            status = reader.GetInt16(0);
                        }
                    }
                    catch (Exception ex)
                    {
                        Console.WriteLine($"CRM_5 :  {ex.Message}");
                        if (ex.Message == "Error reading data from the connection.")
                        {
                            FbConnection.ClearPool(fbcon);
                        }
                    }
                    finally
                    {
                        reader.Close();
                    }
                }
                catch (Exception ex)
                {
                    Console.WriteLine($"5 : {ex.Message}");
                    if (ex.Message == "Error reading data from the connection.")
                    {
                        FbConnection.ClearPool(fbcon);
                    }
                }
                finally
                {
                    fbcon.Close();
                }

                return status;
            }
        }

        public void addText(int type, string text)
        {
            using (FbConnection fbcon = newfbconn_con())
            {
                string sqlstring = String.Format($"UPDATE templatetextsms SET textSMS =
\"{text}\" WHERE typeSMS = {type}");

                FbCommand InsertSQL = new FbCommand(sqlstring, fbcon);// Добавить аргументы
                try
                {
                    fbcon.Open();
                    FbTransaction fbt = fbcon.BeginTransaction();
                    InsertSQL.Transaction = fbt;
                    int res = InsertSQL.ExecuteNonQuery();
                    fbt.Commit();
                }
                catch (Exception ex)
```

```
                {
                    Console.WriteLine($"CRM_6 : {ex.Message}");
                    if (ex.Message == "Error reading data from the connection.")
                    {
                        FbConnection.ClearPool(fbcon);
                    }
                }
                finally
                {
                    InsertSQL.Dispose();
                    fbcon.Close();
                }
            }
        }

        public string getText(int type)
        {
            using (FbConnection fbcon = newfbconn_con())
            {
                string str = "";
                FbCommand SelectSQL = new FbCommand(String.Format($"Select textSMS FROM
templatetextsms WHERE typeSMS = {type}"), fbcon);

                try
                {
                    fbcon.Open();
                    FbTransaction fbt = fbcon.BeginTransaction();
                    SelectSQL.Transaction = fbt;
                    FbDataReader reader = SelectSQL.ExecuteReader();

                    try
                    {
                        while (reader.Read())
                        {
                            str = reader.GetString(0);
                        }
                    }
                    catch (Exception ex)
                    {
                        Console.WriteLine($"CRM_7 :  {ex.Message}");
                        if (ex.Message == "Error reading data from the connection.")
                        {
                            FbConnection.ClearPool(fbcon);
                        }
                    }
                    finally
                    {
                        reader.Close();
                    }
                }
                catch (Exception ex)
                {
                    Console.WriteLine($"7 : {ex.Message}");
                    if (ex.Message == "Error reading data from the connection.")
                    {
                        FbConnection.ClearPool(fbcon);
                    }
                }
                finally
                {
                    fbcon.Close();
                }

                return str;
            }
        }
```

```
        public List<reportModel> getInformation(int typeReport, string histnum, DateTime ntime,
DateTime ktime)
        {
            using (FbConnection fbcon = newfbconn_con())
            {
                List<reportModel> information = new List<reportModel>();
                try
                {
                    string sql = "";

                    switch (typeReport)
                    {
                        case 0:
                            sql = $"Select idInfodent, idFilial, dateOfReceipt, idClient,
theConfirmation, timeSendMessage2Total, wasItSent1, wasItSent2, wasItSent3, obsolete, status1,
status2, status3, idDoctor, dateAdd  FROM scheduleofmessages";
                            break;
                        case 1:
                            long pcode = getPcode(histnum);
                            sql = $"Select idInfodent, idFilial, dateOfReceipt, idClient,
theConfirmation, timeSendMessage2Total, wasItSent1, wasItSent2, wasItSent3, obsolete, status1,
status2, status3, idDoctor, dateAdd  FROM scheduleofmessages where idClient = '{pcode}'";
                            break;
                        case 2:
                            sql = $"Select idInfodent, idFilial, dateOfReceipt, idClient,
theConfirmation, timeSendMessage2Total, wasItSent1, wasItSent2, wasItSent3, obsolete, status1,
status2, status3, idDoctor, dateAdd  FROM scheduleofmessages where dateOfReceipt between
'{ntime.ToString("yyyy-MM-dd HH:mm:ss")}' and '{ktime.ToString("yyyy-MM-dd HH:mm:ss")}'";
                            break;
                    }

                    FbCommand SelectSQL = new FbCommand(String.Format(sql), fbcon);
                    try
                    {
                        fbcon.Open();
                        FbTransaction fbt = fbcon.BeginTransaction();
                        SelectSQL.Transaction = fbt;
                        FbDataReader reader = SelectSQL.ExecuteReader();

                        try
                        {
                            while (reader.Read())
                            {
                                reportModel report = new reportModel();
                                report.schedid = reader.GetInt64(0);
                                report.idFilial = reader.GetInt64(1);
                                report.dateOfReceipt = reader.GetDateTime(2);
                                report.idClient = reader.GetInt64(3);
                                if (reader[4].ToString() != "") report.theConfirmation =
reader.GetInt16(4); else report.theConfirmation = -1;
                                if (reader[5].ToString() != "") report.timeSendMessage2Total =
reader.GetString(5);
                                report.kitStatusSms1 = reader.GetInt16(6);
                                report.kitStatusSms2 = reader.GetInt16(7);
                                report.kitStatusSms3 = reader.GetInt16(8);
                                report.obsolete = reader.GetInt16(9);
                                report.discontStatusSms1 = reader.GetInt16(10);
                                report.discontStatusSms2 = reader.GetInt16(11);
                                report.discontStatusSms3 = reader.GetInt16(12);
                                report.idDoctor = reader.GetInt64(13);
                                report.dateAdd = reader.GetDateTime(14);

                                information.Add(report);
                            }
                        }
```

```
                    catch (Exception ex)
                    {
                        Console.WriteLine($"CRM_8 : {ex.Message}");
                        if (ex.Message == "Error reading data from the connection.")
                        {
                            FbConnection.ClearPool(fbcon);
                        }
                    }
                    finally
                    {
                        reader.Close();
                    }
                }
                catch (Exception ex)
                {
                    Console.WriteLine($"8.1 : {ex.Message}");
                    if (ex.Message == "Error reading data from the connection.")
                    {
                        FbConnection.ClearPool(fbcon);
                    }
                }
                finally
                {
                    fbcon.Close();
                }

                if (information.Count != 0)
                {
                    foreach (var report in information)
                    {
                        try
                        {
                            report.filial = getFilial(report.idFilial);
                            report.histnum = getHistnum(report.idClient);
                            report.fullname = getFullname(report.idClient);
                            report.dname = getDname(report.idDoctor);
                            report.phone1 = getPhone1(report.idClient);
                            report.phone2 = getPhone2(report.idClient);
                            report.phone3 = getPhone3(report.idClient);
                        }
                        catch (Exception ex)
                        {
                            Console.WriteLine($"8.2 : {ex.Message}");
                        }
                    }
                }
            }
            catch (Exception ex)
            {
                Console.WriteLine($"8.3 : {ex.Message}");
            }
            return information;
        }
    }

    public List<reportModel> getInformationStatisticalReport()
    {
        using (FbConnection fbcon = newfbconn_con())
        {
            List<reportModel> information = new List<reportModel>();
            try
            {
                FbCommand SelectSQL = new FbCommand(String.Format("SELECT sc.idInfodent,
sc.dateofReceipt, sc.idDoctor, " +
                    "sc.idFilial, sm.wasItSent1, sm.wasItSent2, sm.wasItSent3,
sm.theConfirmation, " +
```

```
                        "sc.clvisit, sc.obsolete," +
                        " sc.refusesms, sc.phone, sc.idClient " +
                        "FROM test.scheduleofcare sc " +
                        "left join test.scheduleofmessages sm on" +
                        " sc.idInfodent = sm.idInfodent and sc.idDoctor = sm.idDoctor and
sc.dateOfReceipt = sm.dateOfReceipt;"));

                    try
                    {
                        fbcon.Open();
                        FbTransaction fbt = fbcon.BeginTransaction();
                        SelectSQL.Transaction = fbt;
                        FbDataReader reader = SelectSQL.ExecuteReader();

                        try
                        {
                            while (reader.Read())
                            {
                                reportModel report = new reportModel();
                                report.schedid = reader.GetInt64(0);
                                report.dateOfReceipt = reader.GetDateTime(1);
                                report.idDoctor = reader.GetInt64(2);
                                report.idFilial = reader.GetInt64(3);
                                if (reader[4].ToString() != "") report.kitStatusSms1 =
reader.GetInt16(4); else report.kitStatusSms1 = -100;
                                if (reader[5].ToString() != "") report.kitStatusSms2 =
reader.GetInt16(5); else report.kitStatusSms2 = -100;
                                if (reader[6].ToString() != "") report.kitStatusSms3 =
reader.GetInt16(6); else report.kitStatusSms3 = -100;
                                if (reader[7].ToString() != "") report.theConfirmation =
reader.GetInt16(7); else report.theConfirmation = -1;
                                report.clVisit = reader.GetInt16(8);
                                report.obsolete = reader.GetInt16(9);
                                report.refSMS = reader.GetInt16(10);
                                report.phone1 = reader.GetString(11);
                                report.idClient = reader.GetInt64(12);

                                information.Add(report);
                            }
                        }
                        catch (Exception ex)
                        {
                            Console.WriteLine($"CRM_9 : {ex.Message}");
                            if (ex.Message == "Error reading data from the connection.")
                            {
                                FbConnection.ClearPool(fbcon);
                            }
                        }
                        finally
                        {
                            reader.Close();
                        }
                    }
                    catch (Exception ex)
                    {
                        Console.WriteLine($"9.1 : {ex.Message}");
                        if (ex.Message == "Error reading data from the connection.")
                        {
                            FbConnection.ClearPool(fbcon);
                        }
                    }
                    finally
                    {
                        fbcon.Close();
                    }
```

```
                    if (information.Count != 0)
                    {
                        foreach (var report in information)
                        {
                            try
                            {
                                report.filial = getFilial(report.idFilial);
                                report.histnum = getHistnum(report.idClient);
                                report.fullname = getFullname(report.idClient);
                                report.dname = getDname(report.idDoctor);
                            }
                            catch (Exception ex)
                            {
                                Console.WriteLine($"9.2 : {ex.Message}");
                            }
                        }
                    }
                }
                catch (Exception ex)
                {
                    Console.WriteLine($"9.3 : {ex.Message}");
                }
                return information;
            }
        }

        public long getPcode(string histnum)
        {
            using (FbConnection fbco = newfbcon_co())
            {
                long pcode = 0;
                FbCommand SelectSQL = new FbCommand(String.Format($"select pcode from clients
where histnum = '{histnum}'"), fbco);

                try
                {
                    fbco.Open();
                    FbTransaction fbt = fbco.BeginTransaction();
                    SelectSQL.Transaction = fbt;
                    FbDataReader reader = SelectSQL.ExecuteReader();

                    try
                    {
                        while (reader.Read())
                        {
                            pcode = reader.GetInt64(0);
                        }
                    }
                    catch (Exception ex)
                    {
                        Console.WriteLine($"CRM_10 : {ex.Message}");
                        if (ex.Message == "Error reading data from the connection.")
                        {
                            FbConnection.ClearPool(fbco);
                        }
                    }
                    finally
                    {
                        reader.Close();
                    }
                }
                catch (Exception ex)
                {
                    Console.WriteLine($"10 : {ex.Message}");
                    if (ex.Message == "Error reading data from the connection.")
                    {
```

```
                    FbConnection.ClearPool(fbco);
                }
            }
            finally
            {
                fbco.Close();
            }
            return pcode;
        }
    }

    public string getDname(long idDoctor)
    {
        using (FbConnection fbco = newfbcon_co())
        {
            string dname = "";
            FbCommand SelectSQL = new FbCommand(($"select dname from doctor where dcode =
'{idDoctor}'"), fbco);

            try
            {
                fbco.Open();
                FbTransaction fbt = fbco.BeginTransaction();
                SelectSQL.Transaction = fbt;
                FbDataReader reader = SelectSQL.ExecuteReader();

                try
                {
                    while (reader.Read())
                    {
                        dname = reader.GetString(0);
                    }
                }
                catch (Exception ex)
                {
                    Console.WriteLine($"CRM_11 : {ex.Message}");
                    if (ex.Message == "Error reading data from the connection.")
                    {
                        FbConnection.ClearPool(fbco);
                    }
                }
                finally
                {
                    reader.Close();
                }
            }
            catch (Exception ex)
            {
                Console.WriteLine($"11 : {ex.Message}");
                if (ex.Message == "Error reading data from the connection.")
                {
                    FbConnection.ClearPool(fbco);
                }
            }
            finally
            {
                fbco.Close();
            }
            return dname;
        }
    }

    public long getIdDoctor(long schedid)
    {
        using (FbConnection fbcon = newfbconn_con())
        {
```

```
                long idDoctor = 0;
                FbCommand SelectSQL = new FbCommand(($"Select idDoctor FROM scheduleofcare where
idInfodent = {schedid}"), fbcon);

                try
                {
                    fbcon.Open();
                    FbTransaction fbt = fbcon.BeginTransaction();
                    SelectSQL.Transaction = fbt;
                    FbDataReader reader = SelectSQL.ExecuteReader();

                    try
                    {
                        while (reader.Read())
                        {
                            idDoctor = reader.GetInt64(0);
                        }
                    }
                    catch (Exception ex)
                    {
                        Console.WriteLine($"CRM_12 : {ex.Message}");
                        if (ex.Message == "Error reading data from the connection.")
                        {
                            FbConnection.ClearPool(fbcon);
                        }
                    }
                    finally
                    {
                        reader.Close();
                    }
                }
                catch (Exception ex)
                {
                    Console.WriteLine($"12 : {ex.Message}");
                    if (ex.Message == "Error reading data from the connection.")
                    {
                        FbConnection.ClearPool(fbcon);
                    }
                }
                finally
                {
                    fbcon.Close();
                }
                return idDoctor;
            }
        }

        public string getPhone1(long idClient)
        {
            using (FbConnection fbco = newfbcon_co())
            {
                string phone1 = "";
                FbCommand SelectSQL = new FbCommand(($"select phone1 from clients where pcode =
'{idClient}'"), fbco);

                try
                {
                    fbco.Open();
                    FbTransaction fbt = fbco.BeginTransaction();
                    SelectSQL.Transaction = fbt;
                    FbDataReader reader = SelectSQL.ExecuteReader();

                    try
                    {
                        while (reader.Read())
                        {
```

```
                        phone1 = reader.GetString(0);
                    }
                }
                catch (Exception ex)
                {
                    Console.WriteLine($"CRM_13 : {ex.Message}");
                    if (ex.Message == "Error reading data from the connection.")
                    {
                        FbConnection.ClearPool(fbco);
                    }
                }
                finally
                {
                    reader.Close();
                }
            }
            catch (Exception ex)
            {
                Console.WriteLine($"13 : {ex.Message}");
                if (ex.Message == "Error reading data from the connection.")
                {
                    FbConnection.ClearPool(fbco);
                }
            }
            finally
            {
                fbco.Close();
            }
            return phone1;
        }
    }

    public string getPhone2(long idClient)
    {
        using (FbConnection fbco = newfbcon_co())
        {
            string phone2 = "";
            FbCommand SelectSQL = new FbCommand(($"select phone2 from clients where pcode =
'{idClient}'"), fbco);

            try
            {
                fbco.Open();
                FbTransaction fbt = fbco.BeginTransaction();
                SelectSQL.Transaction = fbt;
                FbDataReader reader = SelectSQL.ExecuteReader();

                try
                {
                    while (reader.Read())
                    {
                        phone2 = reader.GetString(0);
                    }
                }
                catch (Exception ex)
                {
                    Console.WriteLine($"CRM_14 : {ex.Message}");
                    if (ex.Message == "Error reading data from the connection.")
                    {
                        FbConnection.ClearPool(fbco);
                    }
                }
                finally
                {
                    reader.Close();
                }
```

```
                }
                catch (Exception ex)
                {
                    Console.WriteLine($"14 : {ex.Message}");
                    if (ex.Message == "Error reading data from the connection.")
                    {
                        FbConnection.ClearPool(fbco);
                    }
                }
                finally
                {
                    fbco.Close();
                }

                return phone2;
            }
        }

        public string getPhone3(long idClient)
        {
            using (FbConnection fbco = newfbcon_co())
            {
                string phone3 = "";
                FbCommand SelectSQL = new FbCommand(($"select phone3 from clients where pcode =
'{idClient}'"), fbco);

                try
                {
                    fbco.Open();
                    FbTransaction fbt = fbco.BeginTransaction();
                    SelectSQL.Transaction = fbt;
                    FbDataReader reader = SelectSQL.ExecuteReader();

                    try
                    {
                        while (reader.Read())
                        {
                            phone3 = reader.GetString(0);
                        }
                    }
                    catch (Exception ex)
                    {
                        Console.WriteLine($"CRM_15 : {ex.Message}");
                        if (ex.Message == "Error reading data from the connection.")
                        {
                            FbConnection.ClearPool(fbco);
                        }
                    }
                    finally
                    {
                        reader.Close();
                    }
                }
                catch (Exception ex)
                {
                    Console.WriteLine($"15 : {ex.Message}");
                    if (ex.Message == "Error reading data from the connection.")
                    {
                        FbConnection.ClearPool(fbco);
                    }
                }
                finally
                {
                    fbco.Close();
                }
                return phone3;
```

```
            }
        }

        public string getFullname(long idClient)
        {
            using (FbConnection fbco = newfbcon_co())
            {
                string fullname = "";
                FbCommand SelectSQL = new FbCommand(($"select fullname from clients where pcode =
'{idClient}'"), fbco);

                try
                {
                    fbco.Open();
                    FbTransaction fbt = fbco.BeginTransaction();
                    SelectSQL.Transaction = fbt;
                    FbDataReader reader = SelectSQL.ExecuteReader();

                    try
                    {
                        while (reader.Read())
                        {
                            fullname = reader.GetString(0);
                        }
                    }
                    catch (Exception ex)
                    {
                        Console.WriteLine($"CRM_16 : {ex.Message}");
                        if (ex.Message == "Error reading data from the connection.")
                        {
                            FbConnection.ClearPool(fbco);
                        }
                    }
                    finally
                    {
                        reader.Close();
                    }
                }
                catch (Exception ex)
                {
                    Console.WriteLine($"16 : {ex.Message}");
                    if (ex.Message == "Error reading data from the connection.")
                    {
                        FbConnection.ClearPool(fbco);
                    }
                }
                finally
                {
                    fbco.Close();
                }
                return fullname;
            }
        }

        public string getHistnum(long idClient)
        {
            using (FbConnection fbco = newfbcon_co())
            {
                string histnum = "";
                FbCommand SelectSQL = new FbCommand(($"select histnum from clients where pcode =
'{idClient}'"), fbco);

                try
                {
                    fbco.Open();
                    FbTransaction fbt = fbco.BeginTransaction();
```

```
                    SelectSQL.Transaction = fbt;
                    FbDataReader reader = SelectSQL.ExecuteReader();

                    try
                    {
                        while (reader.Read())
                        {
                            histnum = reader.GetString(0);
                        }
                    }
                    catch (Exception ex)
                    {
                        Console.WriteLine($"CRM_17 : {ex.Message}");
                        if (ex.Message == "Error reading data from the connection.")
                        {
                            FbConnection.ClearPool(fbco);
                        }
                    }
                    finally
                    {
                        reader.Close();
                    }
                }
                catch (Exception ex)
                {
                    Console.WriteLine($"17 : {ex.Message}");
                    if (ex.Message == "Error reading data from the connection.")
                    {
                        FbConnection.ClearPool(fbco);
                    }
                }
                finally
                {
                    fbco.Close();
                }
                return histnum;
            }
        }

        public string getFilial(long idFilial)
        {
            using (FbConnection fbco = newfbcon_co())
            {
                string filial = "";
                FbCommand SelectSQL = new FbCommand(($"select shortname from filials where filid
= '{idFilial}'"), fbco);

                try
                {
                    fbco.Open();
                    FbTransaction fbt = fbco.BeginTransaction();
                    SelectSQL.Transaction = fbt;
                    FbDataReader reader = SelectSQL.ExecuteReader();

                    try
                    {
                        while (reader.Read())
                        {
                            filial = reader.GetString(0);
                        }
                    }
                    catch (Exception ex)
                    {
                        Console.WriteLine($"CRM_18 : {ex.Message}");
                        if (ex.Message == "Error reading data from the connection.")
                        {
```

```
                    FbConnection.ClearPool(fbco);
                }
            }
            finally
            {
                reader.Close();
            }
        }
        catch (Exception ex)
        {
            Console.WriteLine($"18 : {ex.Message}");
            if (ex.Message == "Error reading data from the connection.")
            {
                FbConnection.ClearPool(fbco);
            }
        }
        finally
        {
            fbco.Close();
        }
        return filial;
    }
}

public List<string> getUsersName()
{
    using (FbConnection fbcon = newfbconn_con())
    {
        List<string> names = new List<string>();
        FbCommand SelectSQL = new FbCommand(("select fullname from users"), fbcon);

        try
        {
            fbcon.Open();
            FbTransaction fbt = fbcon.BeginTransaction();
            SelectSQL.Transaction = fbt;
            FbDataReader reader = SelectSQL.ExecuteReader();

            try
            {
                while (reader.Read())
                {
                    names.Add(reader.GetString(0));
                }
            }
            catch (Exception ex)
            {
                Console.WriteLine($"CRM_19 : {ex.Message}");
                if (ex.Message == "Error reading data from the connection.")
                {
                    FbConnection.ClearPool(fbcon);
                }
            }
            finally
            {
                reader.Close();
            }
        }
        catch (Exception ex)
        {
            Console.WriteLine($"19 : {ex.Message}");
            if (ex.Message == "Error reading data from the connection.")
            {
                FbConnection.ClearPool(fbcon);
            }
        }
```

```
                finally
                {
                    fbcon.Close();
                }

                return names;
            }
        } // получение имен пользователей

        public short testPasswordUsers(string userName, string password)
        {
            using (FbConnection fbcon = newfbconn_con())
            {
                short test = 0;
                FbCommand SelectSQL = new FbCommand(($"select * from users where fullname =
'{userName}' and password = '{password}'"), fbcon);

                try
                {
                    fbcon.Open();
                    FbTransaction fbt = fbcon.BeginTransaction();
                    SelectSQL.Transaction = fbt;
                    FbDataReader reader = SelectSQL.ExecuteReader();

                    try
                    {
                        while (reader.Read())
                        {
                            test = 1;
                        }
                    }
                    catch (Exception ex)
                    {
                        Console.WriteLine($"CRM_20 : {ex.Message}");
                        if (ex.Message == "Error reading data from the connection.")
                        {
                            FbConnection.ClearPool(fbcon);
                        }
                    }
                    finally
                    {
                        reader.Close();
                    }
                }
                catch (Exception ex)
                {
                    Console.WriteLine($"20 : {ex.Message}");
                    if (ex.Message == "Error reading data from the connection.")
                    {
                        FbConnection.ClearPool(fbcon);
                    }
                }
                finally
                {
                    fbcon.Close();
                }
                return test;
            }
        }
    }
}
```