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»

ДОПУСТИТЬ К ЗАЩИТЕ
Заведующий кафедрой ЭВМ
_____ Г.И. Радченко
«___»_____ 2020 г.

Разработка средств настраиваемой репликации для СУБД SQLite

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА
К ВЫПУСКНОЙ КВАЛИФИКАЦИОННОЙ РАБОТЕ

Руководитель работы,
к.т.н., доцент каф. ЭВМ
_____Е.С. Ярош
«___» _____2020 г.

Автор работы,
студент группы КЭ-405
_____Е.В. Анфалов
«__» _____2020 г.

Нормоконтролер,
ст. преп. каф. ЭВМ
_____ С.В. Сяськов
«___» _____2020 г.

Челябинск-2020

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
«___»_____2020 г.

ЗАДАНИЕ

на выпускную квалификационную работу бакалавра

студенту группы КЭ-405
Анфалову Егору Вадимовичу
обучающемуся по направлению
09.03.01 «Информатика и вычислительная техника»

Тема работы: «Разработка средств настраиваемой репликации для СУБД SQLite» утверждена приказом по университету от 24 апреля 2020 г. №627

1. **Срок сдачи студентом законченной работы:** 1 июня 2020 г.

2. **Исходные данные к работе:**

– тип репликации – асинхронная;
– возможность использования для мобильных устройств под управлением ОС Android;
– реализовать возможность встраивания репликации в приложения;
– разработать демонстрационный пример в виде приложения и документацию для программиста.

3. **Перечень подлежащих разработке вопросов:**

– постановка и анализ задачи;

– анализ-обзор существующих решений;
– определение требований;
– разработка архитектуры и основных проектных решений;
– программная реализация;
– оценка работоспособности разработанного программного обеспечения

4. **Дата выдачи задания:** 1 декабря 2019 г.

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

Студент _____/*Е.В. Анфалов* /

КАЛЕНДАРНЫЙ ПЛАН

Этап	Срок сдачи	Подпись руководителя
Введение и обзор литературы	01.03.2020	
Разработка модели, проектирование	01.04.2020	
Реализация системы	01.05.2020	
Тестирование, отладка, эксперименты	15.05.2020	
Компоновка текста работы и сдача на нормоконтроль	24.05.2020	
Подготовка презентации и доклада	30.05.2020	

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

Студент _____/*Е.В. Анфалов*/

АННОТАЦИЯ

Е.В. Анфалов. Разработка средств настраиваемой репликации для СУБД SQLite. – Челябинск: ФГАОУ ВО «ЮУрГУ (НИУ)», ВШЭКН; 2020, 80 с., 5 табл., 12 ил., 10 листингов, библиогр. список – 9 наим.

В рамках выпускной квалификационной работы выполнена разработка модуля репликации для встраиваемой СУБД SQLite. Репликация организована путем создания журнала транзакций. Разработка рассчитана на использование в мобильных приложениях на базе ОС Android. Для передачи данных между устройствами используется служба коротких сообщений. Цель работы – синхронизация баз данных на различных устройствах без использования хостинг сервера в качестве посредника.

ОГЛАВЛЕНИЕ

ВВЕДЕНИЕ

Каждое устройство в современном мире ежесекундно принимает, отправляет и обрабатывает огромный массив данных, предоставляя пользователю различные виды услуг, начиная от связи, заканчивая облачными вычислениями. Мы привыкли, что главным каналом передачи данных является Интернет с его огромными возможностями. Но главным вопросом остается проблема информационной безопасности, обеспечения конфиденциальности данных, доступности и целостности, а также недопущения какой-либо их компрометации. Для мобильных устройств этот вопрос стоит особенно остро, так как на них нередко хранят больше личной информации, чем на остальных устройствах пользователя.

Существует ряд задач, где требуется контроль выполнения каких-либо действий, которые должен выполнить абонент. Как правило, такие задачи имеют издателя, дистрибьютора, который может быть совмещен с издателем, а также подписчика, который должен выполнять требуемые действия и предоставлять обратную связь издателю о их выполнении. В большинстве случаев такие задачи решаются за счет централизованной базы данных. Однако при наличии мобильных устройств или наличии гетерогенной среды, состоящей из мобильных устройств и стационарных, такое решение не совсем удобно, так как требует размещения централизованной базы данных на каком-либо ресурсе.

Передача данных осуществляется с помощью разных каналов, основным является интернет трафик. Для этой цели разработчики мобильных приложений пользуются услугами хостинга. Крупные компании пользуются платными сервисами по размещению своих данных, компании поменьше, из-за недостатка средств, вынуждены использовать бесплатные сервисы, которые обладают рядом недостатков таких, как наличие рекламы на ресурсах, ограниченный

функционал и отсутствие гарантий по сохранности данных.

Актуальность создания средств передачи данных мобильных приложений без использования хостинг сервисов заключается в том, что безопасность личных данных в современном мире является важной задачей для разработчиков программного обеспечения и пользователей, но не все готовы оплачивать предоставление доступа к персональному выделенному серверу.

Целью представленной выпускной квалификационной работы является разработка средства репликации для СУБД SQLite на базе операционной системы Android с целью организации передачи изменений в базе данных по среде передачи данных без использования хостинг сервера в качестве посредника.

Для достижения поставленной цели, необходимо решить следующие задачи:

1. Рассмотреть современные способы передачи данных между мобильными устройствами.
2. Провести детальный анализ каждого из средств по критериям радиуса действия, скорости передачи данных и использования хостинг сервисов.
3. Разработать модуль для разработчиков мобильных приложений, который осуществляет репликацию базы данных на двух мобильных устройствах.
4. Разработать мобильное приложение для демонстрации функционала разработанного модуля репликации.
5. Оценить работоспособность разработанного программного комплекса.

1 АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ

Репликация (от лат. replico — повторяю) — это процесс, под которым понимается копирование данных из одного источника на другой (или на множество других), и наоборот. При репликации изменения, сделанные в одной копии объекта, могут быть распространены в другие копии.

Репликация в базах данных — это тиражирование изменений данных с главного сервера баз данных на одном или нескольких зависимых серверах. Главный сервер при этом будет называться мастером, а зависимые сервера — репликами.[6]

Существует два типа репликации: синхронная и асинхронная. При синхронной реплицируемые данные обновляются сразу после изменения исходных. Синхронная репликация для обеспечения целостности данных использует технологию двухфазовой фиксации транзакций, т.е. когда изменения исходных данных фиксируются только после получения подтверждения от всех серверов, участвующих в репликации. Такой тип репликации требует постоянной доступности всех участвующих в репликации серверов, что обычно трудно обеспечить. При асинхронной репликации реплицируемые данные обновляются после того, как зафиксировано изменение исходных, задержка может сильно отличаться в зависимости от конкретных условий.

Асинхронная репликация допускает следующие модели:

1) первичная-целевая (primary-target) – изменения происходят в первичной базе данных и реплицируются на целевую(ые);
2) потоковая (workflow) – изменения происходят в первичной базе данных и реплицируются на 1-ю целевую, оттуда на 2-ю целевую и т. д.;
3) равноранговая (update anywhere) – все базы данных равны, изменения допускаются на любом узле и реплицируются на все узлы.

Обычно отслеживание изменений основывается на одном из следующих механизмов:

1) отслеживание изменений с помощью триггеров (trigger-based data capture) – при изменении данных триггер запускает процесс репликации, такой механизм не отслеживает транзакций и не обеспечивает ссылочной целостности данных;
2) отслеживание транзакций с помощью триггеров (trigger-based transaction capture) – изменения данных группируются в транзакции, такой механизм обеспечивает ссылочную целостность, но добавляет накладные расходы;
3) отслеживание изменений с помощью журнала (log-based data capture) – изменения данных берутся из журнала транзакций, репликация не конкурирует с другими процессами за доступ к данным, такой механизм встраивается в обычный процесс журналирования и добавляет минимум работы.[7]

Для СУБД SQLite ни одно из этих решений неприемлемо. Отслеживание с помощью триггеров не позволит использовать одну из самых сильных сторон этой СУБД – хорошую поддержку аппарата транзакций в современном понимании. Накладные расходы, связанные с отслеживанием транзакций с помощью триггеров, могут «отобрать» у приложения, для которого необходима репликация, дефицитные ресурсы. Журнал транзакций СУБД могут использовать для реализации репликации только ее разработчики, имеющие доступ к внутренним механизмам. К тому же, SQLite не хранит журнал транзакций. Он создается только на время работы конкретной транзакции, обеспечивая возможность ее отката. По завершении транзакции журнал удаляется. Этим обеспечивается «легковесность» СУБД. Поэтому

единственным выходом является организация собственного процесса журналирования для целей репликации.

1.2 Обзор аналогов

Был найден единственный аналог разрабатываемой системы, библиотека LiteSync, которая обладает схожим функционалом.[3] Данный программный продукт поставляется с кодом СУБД SQLite и использует тот же интерфейс и существующие оболочки. Библиотеки LiteSync общаются друг с другом, обмениваясь данными транзакций. Стоимость коммерческой лицензии на данный программный продукт составляет 950$. Общая схема работы библиотеки представлена на рисунке 1.1.

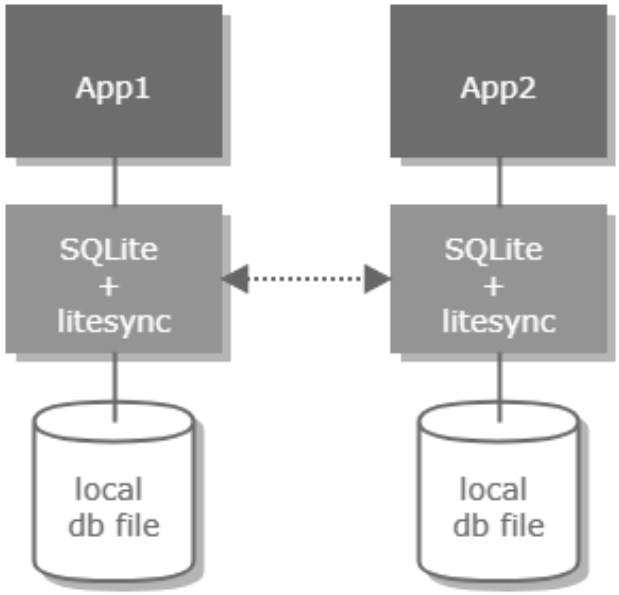

Рисунок 1.1 – Общая схема работы библиотеки LiteSync

Достоинства и недостатки готовой библиотеки и разрабатываемой представлены в таблице 1.

Таблица 1 – Достоинства и недостатки в сравнении с аналогами

Критерий	LiteSync	Разрабатываемый модуль репликации
Расширяет существующий функционал базы данных SQLite	+	+
Использует локальную базу данных при отсутствии соединения между устройствами	+	+
Передача только новых данных	+	+
Кроссплатформенность	+	-
Использует хостинг сервисы	+	-
Лицензирование	+	-

1.3 Анализ основных технологических решений

Для выполнения выпускной квалификационной работы необходимо рассмотреть технологии соединения мобильных устройств. При этом следует учесть, что для передачи файла транзакций базы данных SQLite не требуется широкого канала связи между устройствами, изменения могут затрагивать буквально нескольких строк в базе данных.

Существует несколько способов передачи данных с одного мобильного устройства на другое. Рассмотрим самые используемые. Оценивать технологии будем по критериям, которые критичны для разрабатываемой системы:

1. Радиус, в котором можно обмениваться данными между устройствами.
2. Тип соединения (интернет, непосредственное соединение устройств, передача через сети операторов).

1.3.1 Подключение мобильного устройства к сети Интернет

Данный способ является самым распространенным на сегодняшний день. Существует довольно много технологий, позволяющих устройству выйти в сеть, но самыми используемыми являются:

a) Домашние и общественные сети WiFi.

Термин Wi-Fi не является техническим, но активно применяется современными пользователями. Под аббревиатурой Wi-Fi (от английского словосочетания Wireless Fidelity, которое можно дословно перевести как высокая точность беспроводной передачи данных) в настоящее время понимается целое семейство стандартов передачи цифровых потоков данных по радиоканалам. Другими словами, под термином Wi-Fi пользователи подразумевают технологии беспроводных локальных сетей — Wireless Local Area Network (WLAN, Wireless LAN). Эти технологии позволяют объединять устройства в локальные сети без помощи проводов (т. е. используя радиоволны) и подключать их к Интернету.

Радиус действия Wi-Fi точки стандарта 802.11n в диапазоне частот 2.4 GHz составляет 50 м в прямой видимости.[8]

b) Мобильные сети

Мобильный интернет появился еще в 1991 году и с тех пор скорость передачи данных только растет. Радиус покрытия современных сотовых вышек составляет до 2-х километров. В 2000-х годах появилась системах 2G GPRS, которая обеспечивает скорость передачи данных в диапазоне 56-114 Кбит/с.

В 2001 году в Японии был представлен 3G. Это была революция: скорость передачи данных выросла до 2 Мбит/с – с 114 Кбит/с в технологии 2G. Но улучшилась не только скорость.

Существующий общий стандарт определяет 4G как сеть, в которой скорость 100 Мбит/с предоставляется для абонентов в движении и до 1 Гбит/с – в

идеальных условиях, когда абонентское устройство не движется.

Данное семейство технологий идеально подходит для передачи данных, но подключение к сети интернет вынуждает разработчиков мобильных приложений использовать хостинг сервисы для хранения данных. Выполняемая работа направлена как раз на избавление от этого. К тому же, в отдаленных районах покрытие мобильным интернетом обеспечено не всегда. Поэтому полагаться на технологии доступа к всемирной паутине мы не можем.[3]

1.3.2 Bluetooth

Bluetooth – это беспроводная технология, являющаяся стандартом, который обеспечивает беспроводную передачу данных на небольших расстояниях между мобильными персональными компьютерами, мобильными телефонами и другими устройствами в режиме реального времени. Работая в диапазоне 2,4 ГГц, два аппарата Bluetooth, находящиеся на расстоянии до 10 м, могут передавать данные со скоростью до 720 кбит/с.

Технология хорошо зарекомендовала себя для передачи данных на небольшие расстояния, что не подойдет для условий нашей задачи.[1]

1.3.3 NFC

Данная технология часто встречается в мобильных устройствах. Передача данных осуществляется на расстоянии нескольких сантиметров, максимальная скорость обмена информацией около 400 Кбит/с, рабочая частота 13,56 МГц, время установления соединения не превышает 0.1 с.[3]

1.3.4 SMS

Технология является довольно старой, но от этого ее актуальность не теряется. Передачу небольших текстовых сообщений поддерживают все

современные мобильные устройства. Максимальный размер сообщения в стандарте GSM – 140 байт (1120 бит). Таким образом, при использовании 7-битной кодировки (латинский алфавит и цифры) можно отправлять сообщения длиной до 160 символов. Радиус передачи сообщений определяется радиусом сигнала сотовой вышки, что составляет около двух километров.[9]

1.4 Вывод

После рассмотрения технологий соединения мобильных устройств для передачи данных, наиболее подходящей технологией для задач, решаемых в рамках данной работы, признана технология «Службы коротких сообщений» или SMS. При, казалось бы, обширном покрытии сигналом сотовых вышек, мобильный интернет в удаленных областях поставляется не всегда. Радиус получения SMS сообщения ограничивается силой сигнала, полученной от вышки. Количество передаваемых символов будет достаточно для отправки изменений базы данных на другое устройство. Для пользователя мобильного приложения будет необходимо иметь пакет услуг, включающий в себя отправку SMS, который подключен почти у всех пользователей мобильными сетями.

2 ОПРЕДЕЛЕНИЕ ТРЕБОВАНИЙ

2.1 Функциональные требования

Основные требования к функционалу системы:

1. Организация в виде модуля для дальнейшего встраивания в мобильное приложение.
2. Описание статей репликации.
3. Отслеживание изменения данных.
4. Формирование по расписанию выборки для отправки.
5. Передача информации об изменениях в виде файла.
6. Отправка абоненту файла изменений в базе данных.
7. Прием файла изменений в базе данных.
8. Автоматическое уведомление о получении файла и его использовании.
9. Отслеживание репликации по времени.
10. Наличие мобильного приложения для демонстрации работы модуля репликации, которое отразит основные аспекты работы модуля.
11. Возможность использования ОС Android версии 6 и выше.

2.2 Нефункциональные требования

2.2.1 Требования к пользователям

Пользователь должен иметь мобильный телефон, который работает под управлением операционной системы Android версии не ниже 6 и пакет платных услуг, предоставляющий отправку и получение SMS.

Средство репликации должно быть представлено в виде модуля. Для демонстрации работы системы необходимо создать простое приложение, роли

пользователей делятся на две группы: отправитель и получатель.

2.2.2 Требования к системе безопасности

Система должна обладать следующими средствами обеспечения безопасности:

- Для работы системы на мобильном устройстве необходимо включение разрешения Отправка SMS, Просмотр SMS в памяти смартфона, Прием SMS;
- Ограничение доступа к файлу базы данных для защиты конфиденциальных данных.

2.2.3 Требования к системе уведомлений

Система должна обладать следующим минимальным набором уведомлений:

- получение уведомления об отправленном файле;
- получение уведомления об принятом файле и об обновленной базе данных.

3 ПРОЕКТИРОВАНИЕ

3.1 Потоки данных

Для дальнейшей работы разделим пользователь на издателя и подписчика, так как набор доступных для них функций отличается.

Издатель формирует данные, которые будут доступны в других местах посредством репликации, а подписчик получает реплицированные данные, но также может передавать изменения данных издателю.

На рисунке 3.1 представлена укрупненная схема потоков данных, на которой можно просмотреть, как взаимодействуют издатель и подписчик.

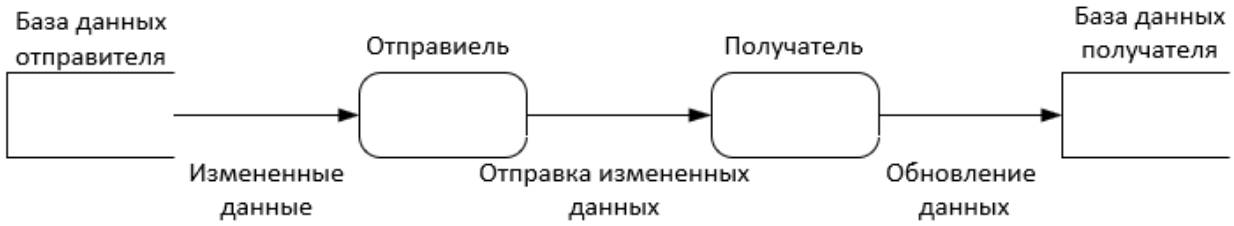

Рисунок 3.1 – Укрупненная схема потоков данных

Сценарий использования системы, на котором представлены функции издателя и подписчика, представлен на рисунке 3.2.

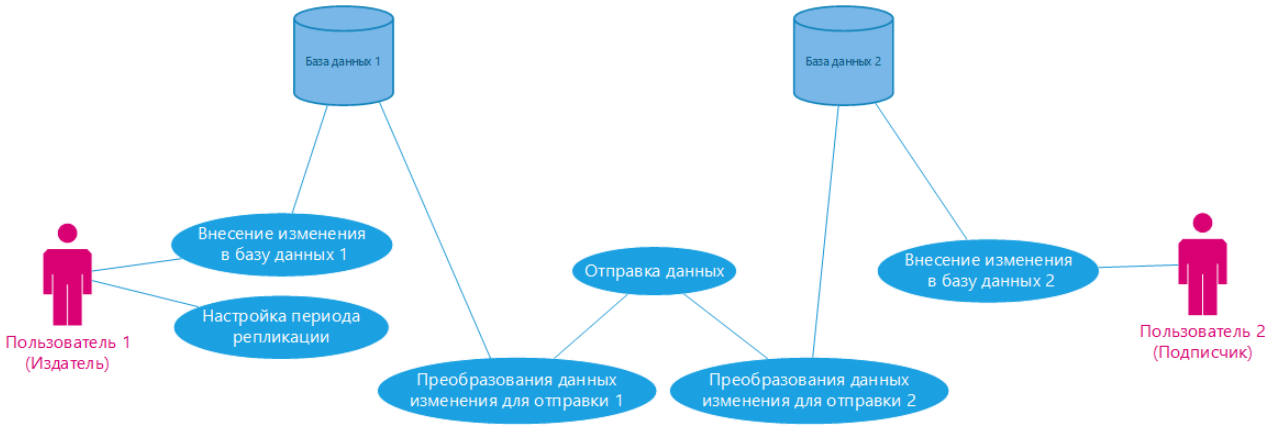

Рисунок 3.2 – Сценарий использования системы

Сценарий, при котором и издатель и подписчик одновременно совершают операции по изменению в базе данных, представлен на рисунке 3.3.

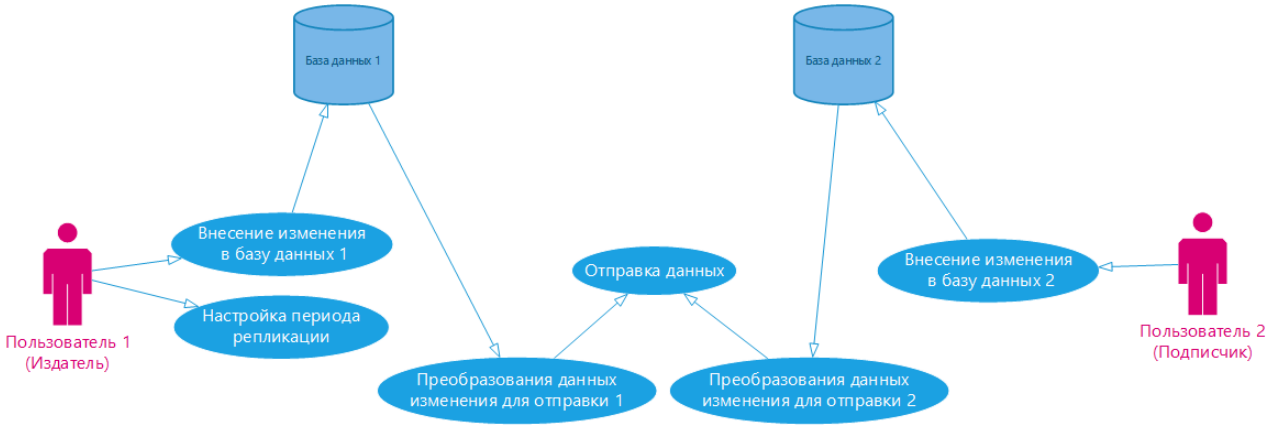

Рисунок 3.3 – Двусторонняя отправка данных

Сценарий, при котором издатель отправляет данные подписчику, представлен на рисунке 3.4.

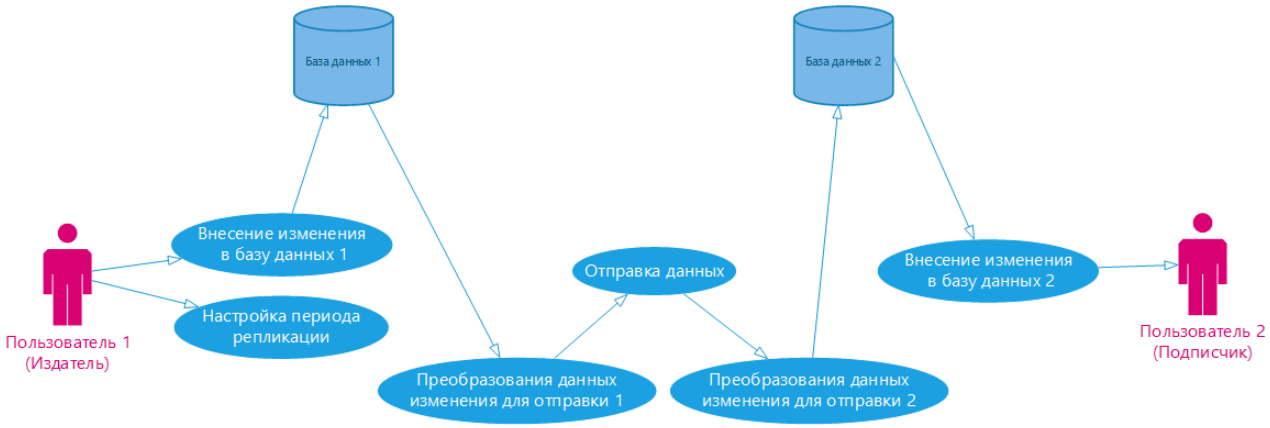

Рисунок 3.4 – Отправка данных от издателя к подписчику

Сценарий, при котором подписчик отправляет данные издателю, представлен на рисунке 3.5.

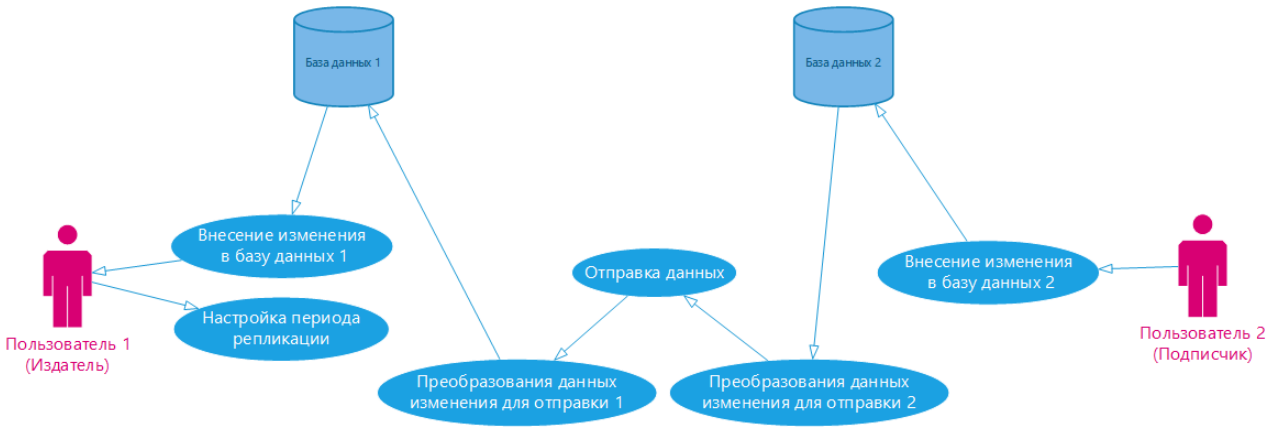

Рисунок 3.5 – Отправка данных от подписчика к издателю

3.2 Основные технические решения

В СУБД SQLite реализация журнала транзакций отличается от других систем. Журнал хранится не в виде файла вместе с базой данных, а создается в начале транзакции, после успешного выполнения которой удаляется. Механизм записи транзакций в общий журнал является удобным средством для

репликации баз данных, поэтому было принято решение создать его для работы модуля репликации.

При внедрении средства репликации в приложение будут созданы два файла: журнал транзакций, куда записываются все изменения в базе данных, и файл лога транзакций, в который записываются изменения за определенный период времени.

Структура двух файлов схожа, в них хранится информация об изменениях в базе данный в виде SQL запросов. В файле журнала хранятся все изменения начиная с создания базы. Используя этот файл, можно вернуть базу данных в исходное состояние при сбое. В файл лога транзакций записываются изменения за определенный период, который ограничен периодом репликации базы данных. После того завершения процесса репликации и отправки данных на другое устройство файл лога очищается.

В СУБД SQLite нет возможности получить структуру таблиц не на экран, а в файл или в виде строки запроса. Из-за этого в модуль репликации невозможно включить блок создания таблиц. Эта функция ляжет на плечи программиста мобильного приложения, который будет пользоваться модулем репликации. При создании базы данных ему будет необходимо самому решить, какие таблицы будут участвовать в процессе репликации и вместо стандартных методов взаимодействия с базой данных SQLite (update, insert, delete) использовать методы разработанного модуля репликации (newupdate, newinsert, newdelete).

3.3 Архитектура модуля репликации

Модуль репликации представлен в виде двух классов: первый класс отвечает за формирование строки запроса, отправленного пользователем, второй класс является вспомогательным и отвечает за работу с файлами лога и журнала транзакций.

Процесс репликации можно поделить на несколько этапов: сбор данных, преобразование их для отправки и сама отправка. Разберем каждый из этапов подробнее.

3.3.1 Этап сбора данных

Разработчик мобильного приложения сам создает таблицы в базе данных. Вместо имеющихся в SQLite методов реализации операций манипулирования данными он использует методы разработанного модуля. Новый метод преобразует подаваемые аргументы в строку SQL запроса, записывая ее в файл журнала и лога транзакций. За этот этап отвечает класс формирования строки запроса.

Для команд Update, Insert и Delete алгоритм работы схож:

1) по данным, указанным разработчиком, формируется строка SQL запроса;

2) запрос выполняется в базе данных;

3) строка SQL запроса сохраняется в файле лога и журнала транзакций.

3.3.2 Этап преобразования данных для отправки

Для увеличения объема полезных данных при отправке изменений по SMS необходимо избавиться от повторяющихся символов таких как: UPDATE, INSERT, DELETE, SET, WHERE. Методы класса работы с файлами лога парсят строку запроса и заменяют слова символами так, чтобы при обратном прочтении было однозначно понятно, какой символ за что отвечает. Например, слово « UPDATE » заменяется буквой « U ». Эта операция сокращает количество SMS, которые необходимо будет отправить для передачи больших изменений в базе данных.

3.3.3 Этап отправки данных на другое устройство

Из-за того, что процесс отправки и прием SMS нельзя вынести в отдельный модуль, реализация отправки и приема SMS между устройствами и настройка периода репликации лежит на разработчике мобильного приложения. Модуль репликации формирует текст сообщения, которое необходимо отправить – строку сообщения из файла лога транзакций.

4 РЕАЛИЗАЦИЯ

4.1 Реализация модуля репликации

Реализация происходит в интегрированной среде разработки для работы с платформой Android – Android Studio.[2]

Язык разработки программного обеспечения для платформы Android – Java.[5]

Модуль репликации состоит из двух классов: основного класса SQLiteReplicationHelper (Приложение А) и вспомогательного класса SingleLine (Пиложение Б).

4.1.1 Класс SQLiteReplicationHelper

Основной задачей данного класса является формирование строки запроса для изменения в базе данных и запись ее в файл журнала транзакций и лога для репликации.

Класс SQLiteReplicationHelper состоит из пяти методов, описание которых представлено в таблице 4.1.

Таблица 4.1 – Описание методов класса SQLiteReplicationHelper

Обозначение	Функционал	Возвращаемое значение
newInsert (String table, String nullColumnHack, ContentValues initialValues, Context context)	Метод newInsert служит для формирования строки SQL запроса для вставки новых записей в базу данных, а также для добавления этой строки в файл логирования	String
newDelete(String table, String whereClause, String[] whereArgs, Context context)	Метод newDelete служит для формирования строки SQL запроса для удаления записей из базы данных, а также для добавления этой строки в файл логирования	String

Продолжение таблицы 4.1

Обозначение	Функционал	Возвращаемое значение
newUpdate(String table, ContentValues values, String whereClause, String[] whereArgs, Context context)	Метод newUpdate служит для формирования строки SQL запроса для обновления записей в базе данных,а также для добавления этой строки в файл логирования	String
queryBuilder(String sql_query, Object[] values)	Метод queryBuilder связывает значения выбора с аргументами	String
getTypeOfObject(Object obj)	Метод getTypeOfObject возвращает тип данных объекта	Int

Рассмотрим каждый из методов подробнее:

1. Метод newInsert

Этот метод используется вместо стандартного метода добавления строки в таблицу. Он возвращает строку SQL запроса. Чтобы вызвать этот метод, необходимо в качестве его аргументов подать: имя таблицы, в которую вносится запись, далее необходимо указать имя столбца, если его значение необходимо установить null, затем необходимо указать объект класса ContentValues, который хранит в первой части название столбца, а во второй его значение, последним аргументом идет объект класса Context, который предоставляет глобальную информацию о среде приложения. Он нужен для записи данных в файлы. Метод записывает в файл журнала и лога транзакций строку SQL запроса и возвращает ее для того, чтобы внести новую строку в базу данных.

2. Метод newDelete

Этот метод используется вместо стандартного метода удаления строки из таблицы. Он возвращает строку SQL запроса. Чтобы вызвать этот метод, необходимо в качестве его аргументов подать: имя таблицы, предикат условия удаления, аргументы для условия, последним аргументом идет объект класса Context, который предоставляет глобальную информацию о среде приложения, он нужен для записи данных в файлы. Метод записывает в файл журнала и лога транзакций строку SQL запроса и возвращает ее для того, чтобы удалить строку из базы данных.

3. Метод newUpdate

Этот метод используется вместо стандартного метода обновления строки из таблицы Он возвращает строку SQL запроса. Чтобы вызвать этот метод, необходимо в качестве его аргументов подать: имя таблицы, объект класса ContentValues, который хранит в первой части название столбца, а во второй его значение, предикат условия обновления, аргументы для условия, последним аргументом идет объект класса Context, который предоставляет глобальную информацию о среде приложения. Он нужен для записи данных в файлы. Метод записывает в файл журнала и лога транзакций строку SQL запроса и возвращает ее для того, чтобы обновить строку в базе данных.

Метод newInsert составляет строку типа:

«**INSERT INTO user (name, pills) VALUES (?, ?);**»,

где вместо аргументов стоят знаки «?», это сделано потому, что необходимо определить, какого типа данных поступили аргументы в запрос. Определением типа данных и подстановкой аргументов в строку SQL запроса занимается метод queryBuilder, куда и идет дальше сформированная строка. Для методов newDelete и newUpdate строка с соответствующими ключевыми словами формируется аналогично.

4. Метод queryBuilder

Этот метод необходим для того, чтобы подставить в строку SQL запроса аргументы согласно их типу данных. Он является вспомогательным методом, который вызывается в каждом из методов, описанных выше. Разработчик не использует его напрямую. В качестве аргументов метод принимает строку SQL запроса, в которой вместо аргументов условия стоят знаки «?», вторым аргументом метод принимает массив из объектов класса Object. В процессе работы метода вместо каждого знака «?» подставляется значение аргумента с учетом его типа данных, который определяется в методе getTypeOfObject.

5. Метод getTypeOfObject

Этот метод необходим для определения типа данных аргумента, который обступает на вход. В качестве входных данных принимает объект класса Object и с помощью оператора instanceof проверяет принадлежность объекта к классу типа данных.

Для разработчика мобильного приложения отличия между обычными операциям по изменению данных и операциями с модулем репликации минимальны. В листинге 4.1 представлен фрагмент кода стандартного изменения строки в таблице.

Листинг 4.1 – Стандартное изменения строки в таблице

```
database.update(DBHelper.TABLE_NAME, contentValues, DBHelper.KEY_ID + "= ?", new String[] {id});
```

В листинге 4.2 представлен фрагмент кода изменения строки с использованием модуля репликации.

Листинг 4.2 – Изменение строки при помощи модуля репликации

```
database.execSQL(replicationHelper.newUpdate(DBHelper.TABLE_NAME, contentValues, DBHelper.KEY_ID + "= ?", new String[]{id}, this));
```

Как видно из листингов 4.1 и 4.2, при использовании модуля репликации программисту необходимо воспользоваться стандартным методом execSQL, который выполняет любой SQL запрос, внутрь которого поместить метод модуля репликации. Последний отличается от стандартного метода только добавлением в конец аргумента «контекст приложения», который требуется для записи данных в файл. При этих несложных манипуляциях таблица станет доступна для репликации.

4.1.2 Класс SingleLine

Основной задачей данного класса является работа с файлами журнала и лога транзакций базы данных, которые называются TransactionJournal и TransactionLog соответственно.

Класс SingleLine состоит из шести методов, описание которых представлено в таблице 4.2.

Таблица 4.2 – Описание методов класса SingleLine

Обозначение	Функционал	Возвращаемое значение
saveToFile(String data)	Сохраняет строку SQL запроса в файл журнала и лога транзакций	Void
readFromLogForSend()	Возвращает из файла лога транзакций строку со всеми данными и преобразует их для дальнейшей отправки	String
convertSMS(String msgBody)	Принимает от обработчика входящих SMS сообщений строку и преобразует ее для обновления базы данных	String
readAllFromJournal()	Возвращает из файла журнала транзакций строку со всеми запросами	String
clearLog()	Очищает файл лога транзакций	Void
clearJournal()	Очищает файл журнала транзакций	Void

Рассмотрим каждый метод класса подробнее:

1. Метод saveToFile

Этот метод предназначен для создания файлов журнала и лога транзакций, а также сохранения строки SQL запроса в них. На вход он принимает строку с запросом, далее для экономии места и количества требуемых SMS сообщений при отправке данных, преобразует данные в сокращенный вид. Все ключевые слова, а именно: INSERT, UPDATE, DELETE, SET, WHERE, VALUES, заменяются на буквы: I, U, D, S, W, V, соответственно.

2. Метод readFromLogForSend

Этот метод предназначен для считывания всех данных из лога транзакций для отправки по SMS. Метод возвращает строку со всеми данными, находящимися в файле, и очищает файл.

3. Метод convertSMS

Этот метод предназначен для преобразования входящего SMS сообщения в полный формат. Он находит все сокращенные слова и заменяет их целыми. На вход метод получает строку из обработчика входящих SMS сообщений, преобразует ее и отправляет обратно.

4. Метод readAllFromJournal

Этот метод предназначен для считывания всех данных из журнала транзакций и преобразования их в полный формат. Метод возвращает строку со всеми данными в полном виде.

5. Методы clearLog и clearJournal

Эти два метода очищают файлы журнала и лога транзакций.

4.2. Реализация демонстрационного мобильного приложения

Для демонстрации функционала модуля репликации создано мобильное приложение (Приложения В – Е). Оно является универсальным, выполняя функции как издателя, так и подписчика в процессе репликации. В нем предусмотрены возможности ввода информации в базу данных, изменения и удаления, а также настройка периода репликации с использованием календаря. Для примера создана небольшая база данных, состоящая из двух таблиц. Первая хранит полезные данные: имя пользователя и некое действие, которое отслеживается, например, прием таблеток. Таблица создана следующим скриптом: «`CREATE TABLE users (_id INTEGER PRIMARY KEY AUTOINCREMENT, name TEXT, pills INTEGER);`». Вторая таблица является служебной, она предназначена для хранения номера телефона второго устройства, а также информации для реализации выбора периода репликации. Можно выбрать варианты: дата прошлого дня, дата дня репликации и значение периода репликации. Таблица создана следующим скриптом: «`CREATE TABLE dates (date_before TEXT, day_d TEXT, period INTEGER, phone_number TEXT);`».

На основном экране, который представлен на рисунке 4.1, расположено все управление базой данных и механизмом репликации. Пользователь может добавлять, удалять, изменять данные в таблице, механизм репликации выглядит следующим образом: пользователь вводит номер телефона второго устройства в специальную графу и сохраняет его в базе данных, нажав кнопку Подтвердить номер, далее необходимо выбрать период репликации (1 день, 1 неделя, 1 месяц) и дату, когда будет осуществляться репликация для периода 1 неделя и 1 месяц. При каждом запуске приложения проверяется, необходимо ли произвести процесс репликации.

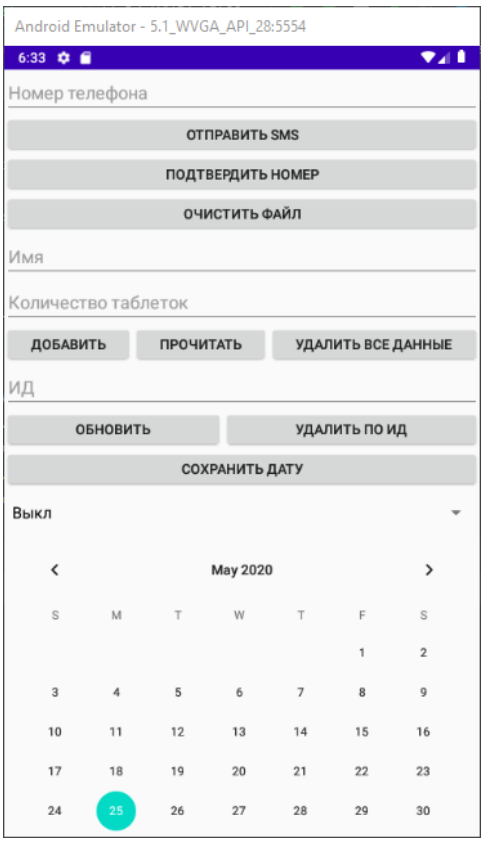

Рисунок 4.1 – Основной экран демонстрационного приложения

На втором экране, который представлен на рисунке 4.2, выводится информация из базы данных, представленная в виде списка.

Рисунок 4.2 – Экран вывода информации из базы данных

Максимальный размер сообщения в стандарте GSM — 140 байт, что составляет 160 символов на латинице или 70 на кириллице.[9] При превышении данного объема, мобильное приложение автоматически разделяет данные для отправки на части и также без участия пользователя соединяет пришедшее сообщение и обновляет базу данных. Файл журнала и лог транзакций содержат SQL запросы в сокращенном виде для увеличения полезных данных при отправке SMS (INSERT заменено на I, UPDATE заменено на U и т. д.). Содержимое файла журнала представлено в листинге 4.1.

Листинг 4.3 – Содержимое файлов журнала и лога транзакций

```
U users S name='Max' ,pills='12' W _id= '4' ;I INTO users(name,pills) V ( 'Kate' , '10' );I INTO users(name,pills) V ( 'Kate' , '10' );
```

В таком же виде, как на листинге 4.1, информация передается с устройства на устройство посредством SMS.

5 ТЕСТИРОВАНИЕ

5.1 Методология тестирования

Так как система представляет собой полностью программный комплекс для мобильных устройств, необходимо провести ее интеграционное тестирование на реальных и эмулированных мобильных телефонах с разными версиями операционных систем. Проводилось системное тестирование, а именно альфа-тестирование реализованного функционала. Альфа-тестирование – имитация реальной работы с системой разработчиком, либо реальная работа с системой потенциальными пользователями. Этот тип тестирования может проводиться как на ранней стадии разработки продукта, так и для законченного продукта в качестве внутреннего приемочного тестирования. Иногда альфа тестирование выполняется под отладчиком или с использованием окружения, которое помогает быстро выявлять найденные ошибки.

Обнаруженные ошибки могут быть переданы тестировщикам для дополнительного исследования в окружении, подобном тому, в котором будет использоваться программа.

5.2 Проведение процедуры тестирования

На момент написания средства репликации актуальными версиями операционной системы Android являются: версия 9 и версия 10. Для проведения интеграционного тестирования после окончательной сборки приложения были взяты или эмулированы на эмуляторе интеграционной среды разработки Android Studio следующие устройства: Redmi Note 8 Pro (версия ОС Android 10), SAMSUNG Galaxy A6+ (версия ОС Android 9), Pixel 2 (версия ОС Android 9),

Pixel 3 XL (версия ОС Android 10). На устройствах были проведены тестирования расположения элементов интерфейса на различных по размеру экранах, были протестированы все основные функции приложения. В результате тестирования ошибок не выявлено.

Выполнено альфа тестирование с разными условиями работы программного комплекса. Проводилась репликация базы данных с разным объемом содержимого файла лога транзакций, 8 строк SQL запросов, при которых сообщение разбивалось на 3 части, и 50 строк SQL запросов, при которых сообщение разбивалось уже на 14 частей.

чтобы посмотреть, насколько корректно система пересылает данные с одного устройства и принимает их на другом. Также была проверена работа функции настройки периода репликации, подтвердилась корректность выбора периода и своевременная отправка данных с устройства.

6 РУКОВОДСТВО ПО ИСПОЛЬЗОВАНИЮ

Средство репликации для СУБД SQLite представлено в виде модуля, который можно подключить как к существующему проекту мобильного приложения, так и использовать его в самом начале разработки программного комплекса. Выглядит модуль как папка с названием sqlitesms. Подробно рассмотрим этапы внедрения модуля в проект и продемонстрируем на примерах, как реализовать полный функционал системы репликации базы данных SQLite.

6.1 Скачивание и внедрение

Для внедрения модуля репликации СУБД SQLite необходимо скачать архив и распаковать его в любое место. Далее необходимо зайти в среду разработки Android Studio, во вкладке File-New выбрать пункт «Import Module…», откроется окно выбора папки, скачанной ранее, как показано на рисунке 5.1.

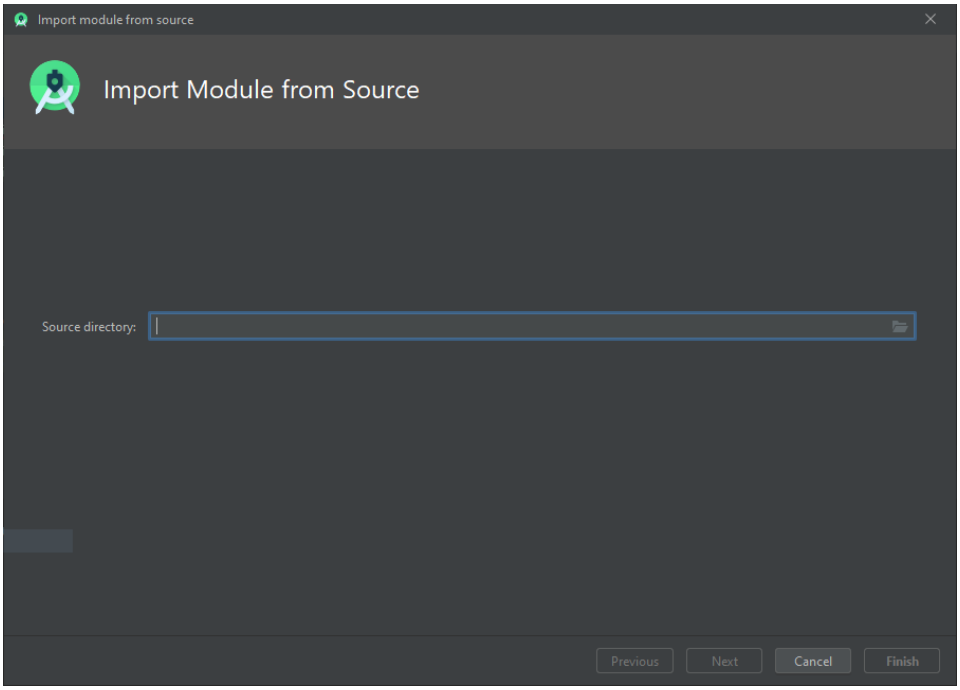

Рисунок 5.1 – Окно импорта модуля

Выбираем директорию расположения скачанного модуля и нажимаем Finish. Среда разработки сама добавит этот модуль в проект мобильного приложения.

Следующим шагом необходимо провести зависимости между модулем и мобильным приложением. Для этого во вкладке File выбираем пункт «Project Structure…», откроется окно структура проекта. В окне структуры проекта необходимо выбрать пункт Dependencies, как показано на рисунке 5.2.

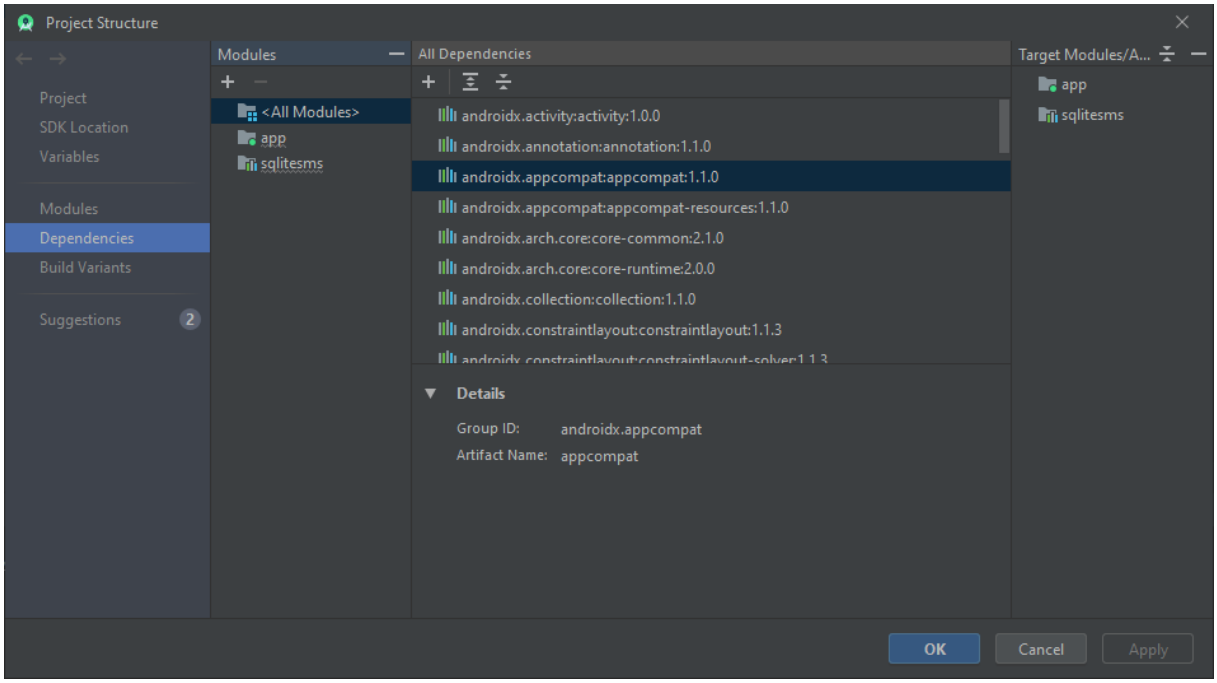

Рисунок 5.2 – Окно структуры проекта

Далее необходимо выбрать в области Modules основное приложение, нажать на знак «+» в области All Dependencies и выбрать Module Dependency, как показано на рисунке 5.3.

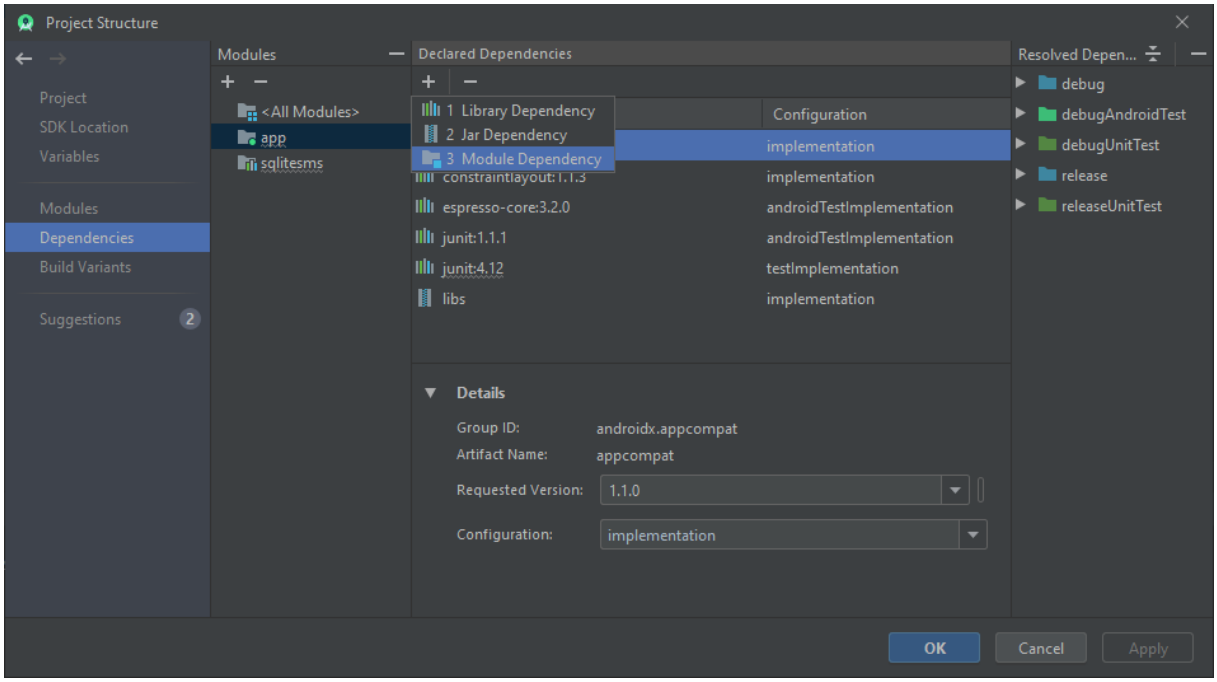

Рисунок 5.3 – Окно структуры проекта

В открывшимся окне выбираем модуль репликации и нажимаем кнопку ОК, как показано на рисунке 5.4.

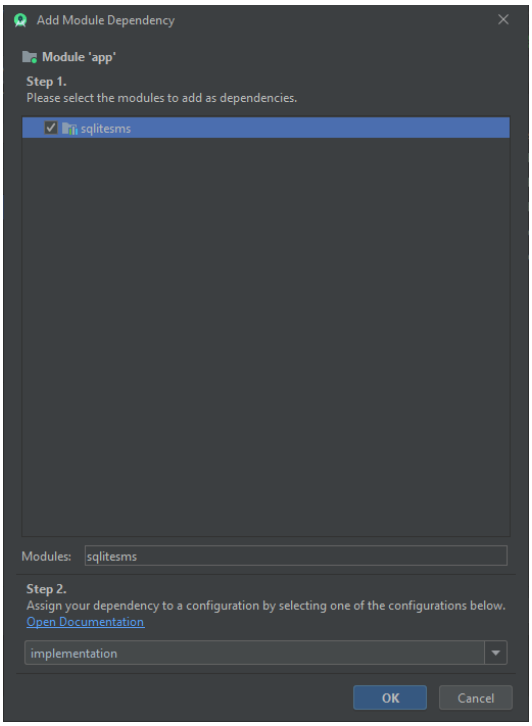

Рисунок 5.4 – Окно добавления зависимости модуля

После этого сохраняем настройки, нажав кнопку Apply в окне структура проекта. Процесс внедрения модуля репликации СУБД SQlite в проект мобильного приложения завершен, можно использовать его для своей базы данных.

6.2 Обзор функционала модуля

Подробно рассмотрим классы и методы модуля репликации для СУБД SQLite. Модуль репликации sqlitesms обладает базовыми функциями для создания журнала и лога транзакций для созданной базы данных. Он состоит из двух классов: основного класса SQLiteReplicationHelper и вспомогательного класса SingleLine. Основной класс модуля представляет собой формирователь строки запроса. При работе с модулем репликации с ним будет происходить основное взаимодействие разработчика приложения.

Для того, чтобы разработчик мобильного приложения использовал знакомые механизмы манипулирования данными в таблицах, методы модуля репликации используют в качестве входных параметров схожие аргументы, как у стандартных методов. Отличие заключается в том, что новые методы возвращают строку SQL запроса, которую, в свою очередь, необходимо выполнить, используя стандартный метод execSQL. Также новые методы принимают в качестве аргумента контекст приложения. В таблице 5.1 представлены стандартные методы манипулирования данными и методы модуля репликации.

Таблица 5.1 – Сравнение стандартных методов манипулирования данными и методов модуля репликации

Функция	Стандартный метод и метод модуля репликации
Метод внесения строки в таблицу (INSERT)	database.insert(DBHelper.*TABLE_NAME*, null, contentValues);
	database.execSQL(replicationHelper.newInsert(DBHelper.*TABLE_NAME*, null, contentValues, this));
Метод удаления строки из таблицы (DELETE)	database.delete(DBHelper.*TABLE_NAME*, DBHelper.*KEY_ID* + "= ", new String[]{id});
	database.execSQL(replicationHelper.newDelete(DBHelper.*TABLE_NAME*, DBHelper.*KEY_ID* + "= ", new String[]{id}, this));
Метод изменения строки в таблице (UPDATE)	database.update(DBHelper.*TABLE_NAME*, contentValues, DBHelper.*KEY_ID* + "= ?", new String[]{id});
	database.execSQL(replicationHelper.newUpdate(DBHelper.*TABLE_NAME*, contentValues, DBHelper.*KEY_ID* + "= ?", new String[]{id}, this));

Рассмотрим каждый из них подробнее.

1. Метод внесения строки в таблицу базы данных (INSERT)

Стандартный метод принимает в качестве аргументов: имя таблицы, куда мы хотим внести строку, далее необходимо указать имя столбца, если его значение необходимо установить null, затем необходимо указать объект класса ContentValues, который хранит в первой части название столбца, а во второй его значение.

2. Метод удаления строки из таблицы (DELETE)

Стандартный метод принимает в качестве аргументов: имя таблицы, из которой мы хотим удалить строку, предикат условия удаления, аргументы для условия.

3. Метод изменения строки в таблице (UPDATE)

Стандартный метод принимает в качестве аргументов: имя таблицы, в которой мы хотим обновить строку, объект класса ContentValues, который хранит в первой части название столбца, а во второй его значение, предикат условия обновления, аргументы для условия.

Методы модуля репликации принимают в конце еще один аргумент – объект класса Context, который предоставляет глобальную информацию о среде приложения. Он нужен для записи данных в файлы.

Второстепенный класс SingleLine отвечает за работу с файлами журнала и лога транзакций. которые называются TransactionJournal и TransactionLog соответственно.

Именно в нем находятся методы добавления информации в файлы, ее извлечения и очистки файлов.

Класс SingleLine состоит из шести методов, описание которых представлено в таблице 5.2.

Таблица 5.2 – Описание методов класса SingleLine

Обозначение	Функционал	Возвращаемое значение
saveToFile(String data)	Сохраняет строку SQL запроса в файл журнала и лога транзакций	Void
readFromLogForSend()	Возвращает из файла лога транзакций строку со всеми данными и преобразует их для дальнейшей отправки	String

Продолжение таблицы 5.2

Обозначение	Функционал	Возвращаемое значение
convertSMS(String msgBody)	Принимает от обработчика входящих SMS сообщений строку и преобразует ее для обновления базы данных	String
readAllFromJournal()	Возвращает из файла журнала транзакций строку со всеми запросами	String
clearLog()	Очищает файл лога транзакций	Void
clearJournal()	Очищает файл журнала транзакций	Void

Рассмотрим каждый метод класса подробнее:

1. Метод saveToFile

Этот метод предназначен для создания файлов журнала и лога транзакций, а также сохранения строки SQL запроса в них. На вход он принимает строку с запросом, далее для экономии места и количества требуемых SMS сообщений при отправке данных, преобразует данные в сокращенный вид. Все ключевые слова, а именно: INSERT, UPDATE, DELETE, SET, WHERE, VALUES, заменяются на буквы: I, U, D, S, W, V, соответственно.

2. Метод readFromLogForSend

Этот метод предназначен для считывания всех данных из лога транзакций для отправки по SMS. Метод возвращает строку со всеми данными, находящимися в файле, и очищает файл.

3. Метод convertSMS

Этот метод предназначен для преобразования входящего SMS сообщения в полный формат. Он находит все сокращенные слова и заменяет их целыми. На вход метод получает строку из обработчика входящих SMS сообщений, преобразует ее и отправляет обратно.

4. Метод readAllFromJournal

Этот метод предназначен для считывания всех данных из журнала транзакций и преобразования их в полный формат. Метод возвращает строку со всеми данными в полном виде.

5. Методы clearLog и clearJournal

Эти два метода очищают файлы журнала и лога транзакций.

6.3 Расширение функционала

Из-за того, что не все функции можно вынести в модуль, функционал модуля репликации ограничен созданием файлов журнала и лога транзакций. Остальные функции, такие как: отправка измененных данных с помощью SMS и выбор периода репликации лежит на плечах разработчика.

Приведем примеры реализации функции отправки и приема SMS.

В демонстрационном приложении была создана служебная таблица, она предназначена для хранения номера телефона второго устройства, а также информации для реализации выбора периода репликации. Можно выбрать варианты: дата прошлого дня, дата дня репликации и значение периода репликации. Таблица создана следующим скриптом: «`CREATE TABLE dates (date_before TEXT, day_d TEXT, period INTEGER, phone_number TEXT);`».

6.3.1 Отправка и получение SMS

Для отправки SMS необходимо учесть, что при наличии большого количества данных в файле лога транзакций, может не хватить одного SMS сообщения для передачи измененных данных, поэтому необходимо грамотно реализовать методы отправки и получения SMS.

Метод отправки SMS сообщения представлен на листинге 5.1.

Листинг 5.1 – Метод отправки SMS сообщения

```
   public void sendSms() {

    SingleLine singleLine = new SingleLine(MainActivity.this);
    REQUEST = singleLine.readFromLogForSend();
    singleLine.clearLog();
    try {
        Cursor phone = database.query(DBHelper.TABLE_DATES, null, null,
null, null, null, null);
        if (phone.moveToFirst()) {
            int phoneIndex =
phone.getColumnIndex(DBHelper.KEY_PHONE_NUMBER);
            phoneN = phone.getString(phoneIndex);
        }
        phone.close();
        Log.d("phone_Num", phoneN);

        if (ActivityCompat.checkSelfPermission(MainActivity.this,
Manifest.permission.SEND_SMS) == PackageManager.PERMISSION_GRANTED) {
            SmsManager smsMan = SmsManager.getDefault();
            if (REQUEST.length() >= 150) {
                ArrayList<String> parts = smsMan.divideMessage(REQUEST);
                smsMan.sendMultipartTextMessage(phoneN, null, parts,
null, null);
            } else {
                Log.d("REQUEST", REQUEST);
                smsMan.sendTextMessage(phoneN, null, REQUEST, null,
null);
            }
            Toast.makeText(MainActivity.this, "SMS send for " + phoneN,
Toast.LENGTH_LONG).show();
        } else {
            Toast.makeText(MainActivity.this, "PERMISSION_NOT_GRANTED",
```

```
Toast.LENGTH_SHORT).show();
            ActivityCompat.requestPermissions(MainActivity.this, new
String[]{
                    (Manifest.permission.SEND_SMS)},
REQUEST_CODE_PERMISSION_SEND_SMS);

        }
        Log.d("request", REQUEST);
        REQUEST = "";
    } catch (Exception e) {
        Toast.makeText(MainActivity.this, "Your SMS sent has failed!",
Toast.LENGTH_LONG).show();
        e.printStackTrace();
    }
}
```

Из листинга 5.1 видно, что текст сообщения записывается в созданную ранее переменную REQUEST типа String с помощью метода readFromLogForSend класса SingleLine. Номер телефона мы берем из служебной таблицы, куда он вносится пользователем приложения. Для отправки сообщения, необходимо предоставить приложению разрешение, наличие которого проверяется условием «`if (ActivityCompat.checkSelfPermission(MainActivity.this, Manifest.permission.SEND_SMS) == PackageManager.PERMISSION_GRANTED)`». Если разрешение не было получено, оно запрашивается у пользователя в блоке «else». Также необходима проверка REQUEST на максимальный объем одного SMS сообщения. Для этого вводится условие «`if (REQUEST.length() >= 150)`». Если REQUEST умещается в объем одного сообщения, используется метод `smsMan.sendTextMessage`, иначе строка делится на части с помощью метода `ArrayList<String> parts = smsMan.divideMessage(REQUEST);` для отправки используется другой метод – `smsMan.sendMultipartTextMessage`.

Процесс отслеживания полученных SMS сообщений нуждается в создании отдельного класса, который наследует от класса BroadcastReceiver. Класс представлен на листинге 5.2.

Листинг 5.2 – Класс SmsReceiver

```
public class SmsReceiver extends BroadcastReceiver {

    @Override
    public void onReceive(Context context, Intent intent) {

        if
(intent.getAction().equals("android.provider.Telephony.SMS_RECEIVED")) {
            Bundle bundle = intent.getExtras();
            SmsMessage[] msgs = null;

            if (bundle != null) {
                try {
                    Object[] pdus = (Object[]) bundle.get("pdus");
                    msgs = new SmsMessage[pdus.length];
                    String msgBody = "";
                    for (int i = 0; i < msgs.length; i++) {

                        if (Build.VERSION.SDK_INT >=
Build.VERSION_CODES.M){
                            String format = bundle.getString("format");
                            msgs[i] =  SmsMessage.createFromPdu((byte[])
pdus[i], format);
                        }else
                        {
                            msgs[i] =  SmsMessage.createFromPdu((byte[])
pdus[i]);
                        }
                        msgBody += msgs[i].getMessageBody();
                    }
```

```
                    MainActivity inst = MainActivity.Instance();
                    inst.updateDbFromSMS(msgBody);
                } catch (Exception e) {
                }
            }
        }
    }
}
```

Данный класс является стандартным решением для приема SMS сообщений, в переменную msgBody вносится текст всех частей сообщения, но для того, чтобы выполнить код, переданный в SMS сообщении, необходимо разбить его на отдельные запросы. Для этой задачи в классе основного окна был создан метод updateDbFromSMS, который представлен на листинге 5.3, он принимает полное сообщение и разбивает его на запросы и каждый запрос в дальнейшем выполняется.

Листинг 5.3 – Метод updateDbFromSMS

```
public void updateDbFromSMS(String Massage) {

    String query = "";
    SingleLine singleLine = new SingleLine(this);
    query = singleLine.convertSMS(Massage);
    Toast.makeText(this, query, Toast.LENGTH_SHORT).show();
    Log.d("receive svs:", query);
    String[] request = query.split(";");
    for (String requests : request) {
        database.execSQL(requests);
    }
}
```

Из листинга 5.3 видно, что метод разбивает сообщение по знаку «;», который есть в конце каждого SQL запроса.

6.3.2 Выбор периода репликации

Для реализации данного функционала было принято решение использовать календарь. В мобильных приложениях есть стандартный компонент calendarView, который мы используем для получения даты следующей репликации. Для выбора периода репликации используем компонент spinner, в котором можно выбрать период (1 день, 1 неделя, 1 месяц).

Для установки периода репликации необходимо выбрать день в календаре, а также из выпадающего списка выбрать период репликации. Эти данные будут записаны в служебную таблицу `dates`.

Каждый раз при запуске приложения система необходимо проверять, какой стоит период репликации, и в зависимости от периода использовать специальные методы: one_day_period_replication, one_week_period_replication, one_month_period_replication. Данные методы приведены в листингах 5.4–5.6.

Листинг 5.4 – Метод one_day_period_replication

```
public void one_day_period_replication() {
 ContentValues date_update = new ContentValues();
 Date date_before = new Date(0);
 Cursor date = database.query(DBHelper.TABLE_DATES, null, null, null,
null, null, null);
 if (date.moveToFirst()) {
     int dateIndex = date.getColumnIndex(DBHelper.KEY_DATE_BEFORE);
     String date_before_string = date.getString(dateIndex);
     if (!date_before_string.isEmpty()) {
         try {
             SimpleDateFormat parser = new SimpleDateFormat("EEE MMM d
HH:mm:ss zzz yyyy");
             date_before = parser.parse(date_before_string);
         } catch (ParseException e) {
             e.printStackTrace();
```

```
            }
            Date date_before1 = new Date(date_before.getTime() +
21600000L);
            if (date_now.after(date_before1)) {
                Log.d("SMS", "ПОРА ОТПРАВЛЯТЬ СМС");
                sendSms();
                date_update.put(DBHelper.KEY_DATE_BEFORE,
date_now.toString());
                database.update(DBHelper.TABLE_DATES, date_update, null,
null);
            }
        } else {
            Log.d("SMS", "РАНО ОТПРАВЛЯТЬ СМС");
            date_update.put(DBHelper.KEY_DATE_BEFORE,
date_now.toString());
            database.update(DBHelper.TABLE_DATES, date_update, null,
null);
        }

    }
    date.close();
}
```

Листинг 5.5 – Метод one_week_period_replication

```
public void one_week_period_replication() {
    ContentValues date_update = new ContentValues();
    Date date_d = new Date(0);
    Cursor date = database.query(DBHelper.TABLE_DATES, null, null, null,
null, null, null);
    if (date.moveToFirst()) {
        int dateIndex = date.getColumnIndex(DBHelper.KEY_DAY_D);
        String date_d_string = date.getString(dateIndex);
        SimpleDateFormat parser = new SimpleDateFormat("EEE MMM d
```

```
HH:mm:ss zzz yyyy");
        if (!date_d_string.equals("null")) {
            Log.d("Date_D:", date.getString(dateIndex));
            try {
                date_d = parser.parse(date_d_string);
            } catch (ParseException e) {
                e.printStackTrace();
            }
            if (date_now.after(date_d)) {
                Log.d("SMS", "ПОРА ОТПРАВЛЯТЬ СМС");
                sendSms();
                Date new_date = new Date(date_now.getTime() +
604800000L);
                date_update.put(DBHelper.KEY_DAY_D, new_date.toString());
                database.update(DBHelper.TABLE_DATES, date_update, null,
null);
            } else Log.d("SMS", "РАНО ОТПРАВЛЯТЬ СМС");
        } else {
            sendSms();
            Date new_date = new Date(date_now.getTime() + 604800000L);
            date_update.put(DBHelper.KEY_DAY_D, new_date.toString());
            database.update(DBHelper.TABLE_DATES, date_update, null,
null);
        }

    }
    date.close();
}
```

Листинг 5.6 – Метод one_month_period_replication

```
public void one_month_period_replication() {
    ContentValues date_update = new ContentValues();
    Date date_d = new Date(0);
```

```
    Cursor date = database.query(DBHelper.TABLE_DATES, null, null, null,
null, null, null);
    if (date.moveToFirst()) {
        int dateIndex = date.getColumnIndex(DBHelper.KEY_DAY_D);
        String date_d_string = date.getString(dateIndex);
        SimpleDateFormat parser = new SimpleDateFormat("EEE MMM d
HH:mm:ss zzz yyyy");
        if (!date_d_string.equals("null")) {
            try {
                date_d = parser.parse(date_d_string);
            } catch (ParseException e) {
                e.printStackTrace();
            }
            if (date_now.after(date_d)) {
                Log.d("SMS", "ПОРА ОТПРАВЛЯТЬ СМС");
                sendSms();
                Date new_date = new Date(date_now.getTime() +
2592000000L);
                date_update.put(DBHelper.KEY_DAY_D, new_date.toString());
                database.update(DBHelper.TABLE_DATES, date_update, null,
null);
            } else Log.d("SMS", "РАНО ОТПРАВЛЯТЬ СМС");
        } else {
            sendSms();
            Date new_date = new Date(date_now.getTime() + 2592000000L);
            date_update.put(DBHelper.KEY_DAY_D, new_date.toString());
            database.update(DBHelper.TABLE_DATES, date_update, null,
null);
        }

    }
    date.close();
}
```

В листинге 5.4 метод сравнивает сегодняшнюю дату с датой, которая записана в служебной таблице. Если она больше, то происходит репликация и обновляется дата в таблице. Если дата меньше, например, если мы второй раз за день запустили приложение, то ничего не произойдет.

В листинге 5.5 метод сравнивает сегодняшнюю дату с датой выбранной отправки, которая записана в служебной таблице. Если она больше, то происходит репликация, и дата следующей отправки в таблице становится на неделю больше.

В листинге 5.6 функционал аналогичен листингу 5.5, только после отправки дата становится больше на месяц.

Данные методы являются лишь примером реализации функционала. Разработчик приложения в праве сам выбрать, как ему удобнее использовать возможности для дальнейшей работы своего программного продукта.

6. ЗАКЛЮЧЕНИЕ

В рамках выпускной квалификационной работы был спроектирован и реализован модуль репликации для системы управления базами данных SQLite.

Для достижения поставленной цели были решены следующие задачи:

1) выполнена постановка и анализ задачи;

2) выполнен анализ-обзор существующих решений;

3) определены требования к системе;

4) разработана архитектура модуля репликации;

5) реализован модуль репликации и демонстрационное мобильное приложение с учетом необходимых функциональных и нефункциональных требований;

6) проведено тестирование мобильного приложения с использованием интеграционного тестирования и системного тестирования.

7) составлена подробная документация на разработанные средства, оформленная в виде руководства программисту, даны рекомендации для полноценной реализации всех возможностей программного обеспечения в мобильном приложении.

БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1. BLUETOOTH SPECIFICATION Version 2.0 + EDR. – URL: http://www.bluetooth.com (Дата обращения: 26.03.2020).

2. Developers / Страница скачивания Android Studio. – URL: https://developer.android.com/studio (Дата обращения: 30.02.2020).

3. Библиотека Litesync – URL: http://litesync.io/index.html (Дата обращения: 20.01.2020).

4. Семенов, Ю.А. NFC и беспроводные интерфейсы приложений / Ю.А. Семенов (ИТЭФ-МФТИ) – URL: http://book.itep.ru/4/41/nfc.htm (Дата обращения: 25.02.2020).

5. Голощапов, А.Л. Google Android: создание приложений для смартфонов и планшетных ПК / А.Л. Голощапов. – 2-е изд. – Санкт-Петербург: БХВ-Петербург, 2014. – 923 с.

6. Известия ТулГУ. Технические науки. – Тула: Изд-во ТулГУ, – 2013. – Вып. 9. – ч. 2. – 421 с.

7. Боев, В.Д. Компьютерное моделирование систем: учебное пособие для среднего профессионального образования / В.Д. Боев. – Москва: Издательство Юрайт, 2019. – 253 с.

8. Смирнова, Е.В. Технологии современных беспроводных сетей Wi-Fi / Е.В. Смирнова, А.В. Пролетарский, Е.А. Ромашкина и др. – Москва: Изд-во МГТУ им. Н. Э. Баумана, 2017. – 446 с.

9. Ле Бодик, Гвинель. Технологии и службы мобильной передачи данных SMS, EMS и MMS / Г. Ле Бодик; пер. с англ. Ш. Салиева; под ред. В. Орлова. – Москва: Техносфера, 2008. – 543 с.