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
_____ Д.В. Топольский
«___»_____ 2024 г.

Разработка чат-бота для составления недельного меню на семью с интеграцией системы доставки

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА
К ВЫПУСКНОЙ КВАЛИФИКАЦИОННОЙ РАБОТЕ
ЮУРГУ-090301.2024.405 ПЗ ВКР

Руководитель работы,
к.п.н., доцент каф. ЭВМ
_____ Ю.Г. Плаксина
«___»_____2024 г.

Автор работы,
студент группы КЭ-405
_____А.В. Вавилов
«___»_____2024 г.

Нормоконтролёр,
ст. преподаватель каф. ЭВМ
_____ С.В. Сяськов
«___»_____2024 г.

Челябинск-2024

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
_____ Д.В. Топольский
«___»_____2023 г.

ЗАДАНИЕ

на выпускную квалификационную работу бакалавра

студенту группы КЭ-405
Вавилову Александру Викторовичу
обучающемуся по направлению
09.03.01 «Информатика и вычислительная техника»

Тема работы: «Разработка чат-бота для составления недельного меню на семью с интеграцией системы доставки»

Срок сдачи студентом законченной работы: 01 июня 2024 г.

Исходные данные к работе:

Функциональные требования

1. Пользователи могли бы отправлять рецепты в течении недели.
2. Система должна генерировать меню в конце недели.
3. Пользователь должен иметь возможность добавить максимальную сумму денег для расчёта стоимости продуктов.

Перечень подлежащих разработке вопросов:

Выпускная квалификационная работа (ВКР) должна содержать разработку следующих вопросов:

1. Аналитический обзор научно-технической, нормативной и методической литературы по тематике работы.
2. Анализ и выбор средств реализации проекта согласно поставленным требованиям.
3. Проектирование бота.
4. Реализация и проведение тестирования работоспособности бота.

Дата выдачи задания: ___ декабря 2023 г.

Руководитель работы _____/ Ю.Г. Плаксина /

Студент _____/ А.В. Вавилов /

КАЛЕНДАРНЫЙ ПЛАН

Этап	Срок сдачи	Подпись руководителя
Аналитический обзор научно-технической, нормативной и методической литературы	03.03.2024	
Анализ и выбор средств реализации проекта согласно поставленным требованиям	21.03.2024	
Проектирование бота	04.04.2024	
Проведение тестирования работоспособности бота	25.04.2024	
Компоновка текста работы и сдача на нормоконтроль	27.05.2024	
Подготовка презентации и доклада	30.05.2024	

Руководитель работы _____/ Ю.Г. Плаксина /

Студент _____/ А.В. Вавилов /

Аннотация

А.В. Вавилов. Разработка чат-бота для составления недельного меню на семью с интеграцией системы доставки. – Челябинск: ФГАОУ ВО «ЮУрГУ (НИУ)», ВШ ЭКН; 2024, 66 с., 19 ил., библиогр. список – 31 наим.

Выпускная квалификационная работа посвящена разработке Telegram бота для составления меню на семью в течения недели с интеграцией сервиса доставки.

В первой главе проведен анализ предметной области, рассмотрены существующие аналоги разрабатываемого приложения.

Во второй главе обоснованы функциональные и нефункциональные требования к приложению, а также представлены критерии выбора основного программного обеспечения, в частности, React. Осуществлен выбор фреймворков, с учетом масштабируемости и возможности добавления новых функциональных компонентов.

Третья глава посвящена разработке составных частей приложения, а также структуры и логики приложения. Реализована основа для создания приложения.

В четвертой главе показаны этапы создания приложения. Рассмотрены и проведены методы тестирования приложения.

ОГЛАВЛЕНИЕ

ВВЕДЕНИЕ

В современном мире, где люди постоянно заняты своими делами, всё больше ценится удобство и экономия времени. Быстрый темп жизни приводит к распространению неправильных пищевых привычек, что в свою очередь способствует увеличению случаев ожирения и заболеваний желудочно-кишечного тракта. Одной из причин таких проблем является несбалансированное питание. В этой связи разработка чат-бота, который поможет составить недельное меню для семьи и организовать заказ продуктов с доставкой, может стать востребованным инструментом для улучшения качества жизни.

Чаты и мессенджеры уже стали неотъемлемой частью нашей повседневной жизни. Мы используем их для общения, ведения переговоров и развлечений. Современные технологии, включая чат-боты, активно упрощают и улучшают различные аспекты нашей жизни. Чат-боты способны предоставить удобные и оперативные решения для выполнения разнообразных задач, что делает их идеальными кандидатами для автоматизации рутинных процессов, таких как планирование питания и организация покупок.

Целью данного проекта является разработка чат-бота для составления недельного меню на семью с интеграцией системы доставки. Для достижения этой цели необходимо создать компоненты приложения, способные эффективно обрабатывать входящую информацию и предоставлять адекватные ответы пользователям.

Выбор в пользу чат-бота обусловлен несколькими факторами. Во-первых, чат-боты обеспечивают круглосуточный доступ к информации и услугам, что особенно важно для людей с плотным графиком. Во-вторых, использование чат-бота позволяет минимизировать время, затрачиваемое на планирование меню и покупку продуктов, за счёт автоматизации этих процессов. Наконец, чат-боты

могут предложить персонализированные рекомендации на основе предпочтений пользователей и их диетических требований, что способствует более здоровому и сбалансированному питанию.

Особое внимание следует уделить тому факту, что для использования чат-ботов не требуется установка дополнительного программного обеспечения. Это значительно упрощает доступ к сервису для пользователей, поскольку они могут взаимодействовать с ботом через привычные им мессенджеры. Более того, чат-боты обладают кроссплатформенностью, что обеспечивает их совместимость с различными устройствами и операционными системами. Это позволяет пользователям получать доступ к функционалу бота в любое время и в любом месте, используя удобный для них способ.

Разработка чат-бота для составления недельного меню на семью с интеграцией системы доставки может стать ценным инструментом для тех, кто стремится питаться здоровой и вкусной пищей, но при этом не имеет возможности тратить много времени на планирование и покупку продуктов. Такой подход способствует улучшению пищевых привычек и общего качества жизни.

1. АНАЛИТИЧЕСКИЙ ОБЗОР НАУЧНО-ТЕХНИЧЕСКОЙ, НОРМАТИВНОЙ И МЕТОДИЧЕСКОЙ ЛИТЕРАТУРЫ

Современный человек очень занят. У него нет свободного времени, как у его предков. Темп жизни ускоряется, что оказывает значительное влияние на образ жизни, включая питание и здоровье [1]. Из-за нехватки времени мы питаемся на ходу, перекусами, что вызывает ожирение и проблемы со здоровьем [2]. Эта проблема появляется не только у людей, живущих одни, но и в семьях. Это сильно влияет на качество жизни, ведь через рацион питания мы получаем много полезных веществ для здоровья организма. Недостаток или переизбыток тех или иных микроэлементов приводит к проблемам со здоровьем, и в наше время очень распространены болезни, вызванные неправильным рационом. Для решения вопроса "Что съесть?" создаются специальные помощники — интеллектуальные системы контроля за рационом питания и качеством жизни.

Для улучшения жизни людей и помощи в контроле соблюдения норм питания было решено разработать Telegram-бот для составления меню на семью с возможностью интеграции с системой доставки продуктов. Это позволит сократить время на походы по магазинам. Сегодня на рынке существует множество приложений, у каждого из которых есть свои плюсы и минусы. Рассмотрим аналоги данных приложений.

Дietограф: планировщик полезной диеты (AppStore) [3]

Какие задачи решает:

- Расчет сложного уравнения с учетом калорий, белков, жиров и углеводов автоматически;
- Расширение списка продуктов питания, не только творог, яйца и куриная грудка;

- Использование ваших продуктов, а не бесконечного списка из интернета;
- Точный расчет диеты до грамма;
- Оперативный учет всех ваших маленьких грехов;
- Контроль съеденного в любой момент дня;
- Возможность вписать ваши любимые продукты в диету, не снижая её эффективность;
- Приведение вашего питания в прозрачную и точную систему;
- Выработка полезных пищевых привычек и образа жизни;
- Помощь спортсменам при обычной программе занятий;
- Помощь спортсменам при наборе массы/сушке.

Для кого предназначено:

- Для профессиональных спортсменов;
- Для тех, кто придерживается здорового образа жизни;
- Для персональных тренеров, расчет диеты клиентам за считанные минуты;
- Для желающих питаться полезно и разнообразно;
- Для тех, кто придерживается ограничений в питании;
- Для желающих похудеть или набрать вес.

Среда разработки и язык программирования:

- Приложение разработано для IOS, использует Swift.

Ограничения:

- Приложение платное;
- Есть ограничения по платформе — только IOS;
- Сложный интерфейс для некоторых пользователей.

Календарь Рецептов (AppStore) [4]

Возможности:

- Автоматический список продуктов;
- Инструкция к каждому рецепту;
- Подбор рецептов по предпочтениям.

Для кого предназначено:

- Для семей;
- Людей, которые хотят разнообразить своё питание;
- И просто кто любит вкусно покушать.

Среда разработки и язык: Приложение разработано для IOS, на языке Swift.

Ограничения:

- Только на IOS;
- Один пользователь;
- Нет интеграции с доставкой.

Данное приложение имеет свои преимущества по отношению с первым. Оно выдаёт автоматически список продуктов. Это очень полезная функция в подобных приложениях, но всё же оно только для одного человека и только на IOS, что ограничивает возможности или даже ущемляет права членов семьи. Ведь если только один человек из семьи будет составлять меню, то идея составлять меню быстро придёт к провалу.

Weekly Meal Planner + Calendar(AppStore/PlayMarket) [5]

Возможности:

- Автоматический список продуктов;
- Использование продуктов, которые есть дома;

- Инструкция к каждому рецепту;
- Подбор рецептов по предпочтениям;
- Можно поделиться меню с партнёром.

Для кого предназначено:

- Для семейной пары;
- Людей, которые хотят разнообразить своё питание;
- И просто кто любит вкусно покушать.

Среда разработки и язык: Приложение разработано для IOS и Android, на языке Swift (iOS), Kotlin/Java (Android).

Ограничения:

- Приложение платное;
- Рассчитано только на двух пользователей;
- Нет интеграции с доставкой.

Данный вариант предоставляет хорошие возможности для составления и использования меню для пары или для одного человека. Но в семье будет неудобно использовать, ведь данное приложение рассчитано только на двух людей, но ведь дети тоже хотят внести свою лепту в семейное меню.

Приложение	Плюсы	Минусы
Диетограф	- Расчет калорий, БЖУ. - Расширенный список продуктов. - Контроль съеденного. - Подходит для спортсменов.	- Платное. - Только iOS. - Сложный интерфейс.
Календарь Рецептов	- Автоматический список продуктов. - Инструкции к рецептам. - Подбор рецептов по предпочтениям.	- Только iOS. - Один пользователь. - Нет интеграции с доставкой.
Weekly Meal Planner	- Автоматический список продуктов. - Использование продуктов из дома. - Инструкции к рецептам. - Подбор рецептов. - Можно поделиться с партнером.	- Платное. - Только два пользователя. - Нет интеграции с доставкой.

Таблица 1 — Обзор аналогов приложений

Все вышеперечисленные приложения имеют свои плюсы и минусы, которые представлены в (Таблица 1), но в общем счёте пользователям не хватает кроссплатформенности, возможности совместного составления меню и так как сейчас обширно развились сервисы доставки, а у людей стало меньше времени на походы в магазины, то не хватает в приложении для составления меню питания интеграции с сервисами доставки.

В книге “Designing Bots: Creating Conversational Experiences” [6] написано о подходах проектирования чат-ботов и создания положительного пользовательского опыта.

Рассмотрим основные аспекты книги:

1. Фокус на пользователе и UX: Книга подчеркивает важность создания удобного и интуитивно понятного пользовательского интерфейса. Для успешного бота важно, чтобы пользователи могли легко составлять меню, выбирать продукты и делать заказы без необходимости в специальных технических навыках.

2. Использование чат-ботов для улучшения качества жизни: Чат-боты, как интеллектуальные помощники, могут значительно улучшить образ жизни пользователей, помогая контролировать рацион питания и обеспечивая доступ к полезной информации о продуктах и рецептах.

3. Интеграция с внешними сервисами: Один из ключевых моментов, который подчеркивает книга, это важность интеграции с внешними системами, такими как сервисы доставки продуктов. Это не только удобство для пользователей, но и потенциальная модель монетизации через партнёрство с доставками или рекламу продуктов.

4. Моделирование пользовательских сценариев: Одной из методик, предложенных в книге, является моделирование типичных сценариев использования бота. Это позволяет учесть разнообразие потребностей пользователей и обеспечить соответствующую функциональность и интерфейс.

Рассмотренные подходы помогут в дальнейшем в разработке и проектирование.

Вывод по главе 1

В результате аналитического обзора научно-технической, нормативной и методической литературы можно сделать несколько ключевых выводов. Современный образ жизни, характеризующийся высоким темпом и нехваткой времени, существенно влияет на питание и здоровье людей. В ответ на эти

вызовы разрабатываются инновационные подходы, такие как приложения для составления меню и контроля за питанием.

Из обзора видно, что существующие приложения (например, Диетограф, Календарь Рецептов, Weekly Meal Planner) предлагают различные функциональные возможности, но имеют определенные ограничения, такие как ограниченность платформой, отсутствие интеграции с доставкой продуктов и ограниченное количество пользователей.

Важным аспектом для дальнейшего развития подобных приложений является фокус на создание удобного пользовательского интерфейса, что подчеркивается в литературе по дизайну чат-ботов. Интеграция с внешними сервисами, включая доставку продуктов, необходима для повышения удобства использования и расширения функциональности приложений.

Таким образом, для успешного развития Telegram-бота для составления меню на семью с интеграцией доставки продуктов следует уделить внимание пользовательскому опыту, функциональным возможностям и потенциалу интеграции с внешними сервисами, что способствует улучшению качества жизни пользователей и эффективности использования приложения.

2. АНАЛИЗ И ВЫБОР СРЕДСТВ РЕАЛИЗАЦИИ ПРОЕКТА СОГЛАСНО ПОСТАВЛЕННЫМ ТРЕБОВАНИЯМ

В данной главе рассмотрим и определимся с требованиями к системе и установим сценарии работы Telegram-бота.

2.1. Функциональные требования

Определимся с функциональными требованиями для Telegram-бота.

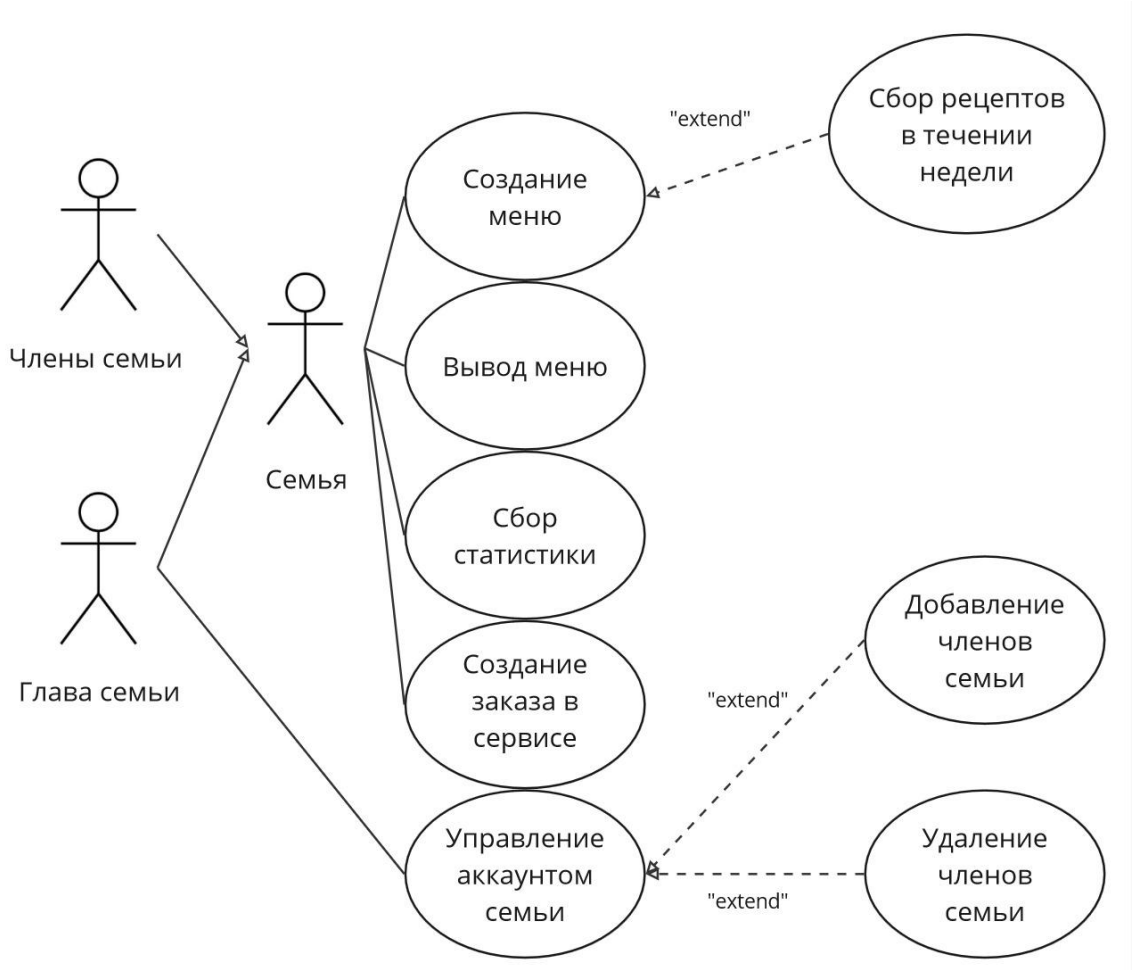

Рисунок 1 — Диаграмма вариантов использования Telegram-бота

На рисунке 1 изображена диаграмма вариантов использования системы. Можно выделить следующих ключевых актеров, взаимодействующих с системой:

1. *Глава семьи*: ответственный член семьи, создающий аккаунт в данной системе и заполняющий всю требуемую информацию.
2. *Члены семьи:* все оставшиеся члены семьи, которые заинтересованы в данной системе.

Рассмотрим сценарии использования разрабатываемой системы.

1. Первый сценарий:

Пользователю необходимо *отправлять рецепты желаемых блюд*, чтобы система Family-Food в течении недели могла составить необходимое меню для семьи.

После процедуры создания меню пользователь системы может начать составление заказа в сервисе доставки продуктов.

2. Второй сценарий:

Пользователь *Глава семьи* (администратор семьи) имеет возможность редактировать аккаунт семьи вносить корректную и важную информацию для работы с системой.

Пользователь *Глава семьи* (администратор семьи) может добавлять и удалять членов семьи.

2.2. Нефункциональные требования

1. Производительность:
   - API должен обеспечивать быстрый доступ к базе данных для обеспечения оперативного выполнения запросов клиентов.
   - Чат-бот должен обладать высокой отзывчивостью и быстро обрабатывать запросы пользователей.
   - Web-приложение должно иметь минимальное время загрузки страниц и быстро реагировать на действия пользователя.

2. Масштабируемость:
   - API и база данных должны быть спроектированы с учетом возможности горизонтального масштабирования для поддержки роста количества пользователей и объема данных.
3. Надежность:
   - Чат-бот и web-приложение должны корректно обрабатывать ошибки и предоставлять пользователю информацию о проблемах при возникновении сбоев.

2.3. Инструменты реализации

Для реализации проекта рассмотрим инструменты и технологии разработки веб приложений и Telegram-ботов.

Выбор языка программирования:

Это всегда очень важный этап в начале разработки так как от этого выбора зависит скорость создания продукта, так и его работоспособность. Так как каждый язык программирования имеет области применения и свои ограничения. На данный момент существует пять подходящих языков программирования: Python, JavaScript, PHP, Go, Ruby. Перейдём к рассмотрению их плюсов и минусов.

1. Python [7]

Это высокоуровневый язык программирования с динамической типизацией, известный своей простотой и читабельностью. Широко используется в науке о данных, веб-разработке и автоматизации.

Плюсы:

- Широкая поддержка библиотек для создания ботов, например, python-telegram-bot;
- Простота синтаксиса, что ускоряет разработку;
- Широкая база обучающих материалов.

Минусы:

- Не подходит для высокопроизводительных задач;
- Может быть медленнее в обработке запросов по сравнению с другими языками.

2. JavaScript(Node.js) [8]

Это язык программирования, работающий в браузере, а Node.js позволяет использовать его на сервере. Известен своей асинхронной моделью выполнения и широким использованием в веб-разработке [9].

Плюсы:

- Асинхронная модель выполнения, что обеспечивает высокую производительность и масштабируемость;
- Большая экосистема модулей, включая такие библиотеки, как node-telegram-bot-api;
- Возможность использования единого языка для фронтенда и бэкенда, что упрощает разработку и поддержку;
- Легкая интеграция с Web-App, поскольку Telegram Web-Apps чаще всего используют фронтенд на JavaScript.

Минусы:

- Асинхронное программирование может быть сложнее для новичков;
- Бывают случаи, когда типизация слабее по сравнению с компилируемыми языками (например, Java или C#).

3. PHP [10]

Это скриптовый язык общего назначения, широко используемый для веб-разработки. Преимущественно используется на серверной стороне для создания динамических веб-страниц.

Плюсы:

- Широкое распространение и доступность хостингов;
- Простота интеграции с веб-сервером.

Минусы:

- Не так хорош для асинхронных задач так как был изначально разработан для синхронного выполнения веб-запросов;
- Менее популярный выбор для современных разработчиков ботов.

4. Go [11]

Это компилируемый язык программирования, разработанный Google. Известен своей производительностью, простотой и поддержкой параллельного программирования.

Плюсы:

- Высокая производительность;
- Хорошо подходит для многозадачности и обработки большого количества запросов.

Минусы:

- Меньше библиотек и примеров для создания Telegram-ботов;
- Сложнее в освоении по сравнению с Python или JavaScript.

5. Ruby [12]

Это высокоуровневый язык программирования с динамической типизацией, ориентированный на простоту и продуктивность. Популярен благодаря своему фреймворку для веб-разработки, Ruby on Rails.

Плюсы:

- Высокая скорость разработки;
- Хорошо читаемый и поддерживаемый код.

Минусы:

- Меньше производительности по сравнению с Go и Node.js;
- Меньше ресурсов и примеров для Telegram-ботов.

•

Язык программирования	Плюсы	Минусы
Python	Простота синтаксиса, широкий выбор библиотек, активное сообщество	Меньшая производительность, не подходит для высокопроизводительных задач
JavaScript (Node.js)	Асинхронная модель, высокая производительность, единый язык для фронтенда и бэкенда, богатая экосистема библиотек	Сложность асинхронного программирования, слабая типизация
PHP	Широкое распространение, доступность хостингов, простота интеграции с веб-сервером	Не подходит для асинхронных задач, менее популярный выбор для ботов
Go	Высокая производительность, поддержка параллелизма	Меньше библиотек для Telegram-ботов, сложнее в освоении
Ruby	Высокая скорость разработки, хорошо читаемый код	Меньшая производительность, меньше ресурсов для Telegram-ботов

Таблица 2 — Сравнение языков программирования

Выводы:

Из таблицы 2 сделаем выводы о лучшем языке программирования для данной разработки.

1. Единый язык для фронтенда и бэкенда:

- Использование JavaScript для обеих сторон (клиентской и серверной) позволяет упростить разработку, отладку и поддержку приложения. Это особенно важно при создании сложных систем с интеграцией Web-App.

2. Асинхронность и производительность:

- Node.js использует неблокирующий ввод/вывод и асинхронную модель выполнения, что делает его очень эффективным для обработки большого количества параллельных запросов, что важно для Telegram-ботов.

3. Экосистема и библиотеки:

- JavaScript имеет одну из самых крупных экосистем библиотек и фреймворков. Библиотеки, такие как node-telegram-bot-api, упрощают создание ботов, а популярные фронтенд-фреймворки, такие как React или Vue.js, помогают в разработке Web-App.

4. Широкая поддержка и сообщество:

- Огромное сообщество разработчиков, множество примеров и руководств, а также активное развитие языка и его экосистемы.

5. Интеграция с другими сервисами:

- JavaScript отлично интегрируется с различными веб-сервисами и API, что облегчает добавление дополнительных функций в бот.

JavaScript (Node.js) из рассмотренных языков самый подходящий для создания Telegram-бота с интеграцией Web-App благодаря своей производительности, возможности использования одного языка для фронтенда и бэкенда, а также богатой экосистеме библиотек и инструментов. Это упрощает процесс разработки, ускоряет время вывода продукта на рынок и снижает затраты на поддержку и масштабирование системы.

Подходы разработки Telegram-бота:

1. Создание через конструктор:

Создание через конструктор позволяет быстро собирать небольшие Telegram-боты с простым функционалом. С помощью них можно собрать бот по доставке товаров или опросник-отзывов. В данном методе логика работы состоит из блоков, простых действий, которые располагаются в конструкторе и выстраивают логику работы бота. [13]

Этот метод достаточно прост, но функционал его сильно ограничен.

2. Создание с помощью написания кода:

На сегодняшний день Telegram-бот можно написать практически на любом популярном языке программирования. Telegram предоставляет открытый API для взаимодействия с ботом. [14] Для упрощения и ускорения разработки используют готовые библиотеки. Так для языка Java Script существует достаточно популярная библиотека Telegraf. [15]

Данный метод позволяет выполнить любой сложности Telegram-бот с любым функционалом.

И в нашем случае подходит второй вариант по причине гибкости и необходимости дополнительного функционала.

Разработка Web-App приложения:

- В 2022 году Telegram в обновлении Bot API 6.1 [16] добавили возможность внедрять web-приложения внутрь ботов, что позволило усовершенствовать работу ботов и функционал, а также пользовательский опыт.

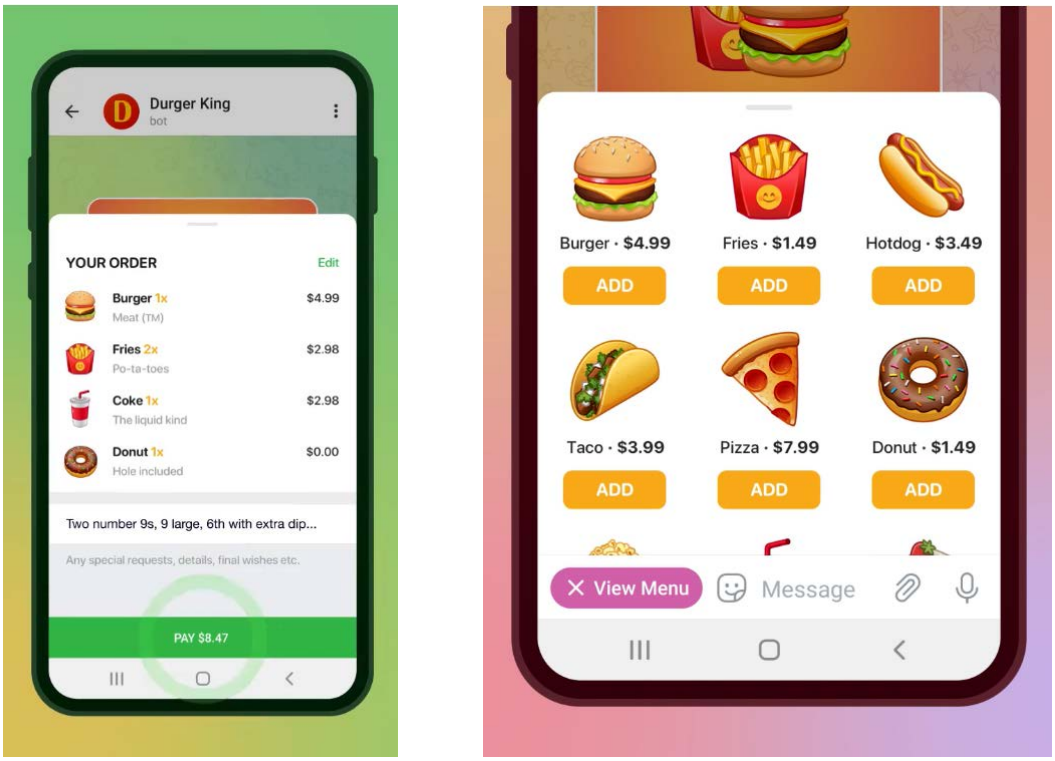

Рисунок 2 — Примеры Web-App приложения в Telegram-бот

Для данного решения необходимо разработать web-приложение и развернуть в сети интернет.

При создании Web-App приложений есть несколько подходов разработки:

- С использованием фреймворка:

На данный момент — это наиболее популярный и современный способ разработки приложений. Основные представители: React, Angular, Next.js. С помощью данного подхода быстрее протекает разработка так как уже есть заготовленные объекты и современные подходы. Также для быстрой разработки используют UI-библиотеки — это библиотеки готовых компонентов, которые позволяют ещё быстрее производить разработку сохраняя общую стилистику.

- Без использования фреймворка:

У данного подхода есть плюсы и минусы. Из плюсов можно отметить абсолютную свободу. Разработчик не ограничен рамками фреймворка, получая

полный контроль над каждой строкой кода. Это открывает возможности для тонкой оптимизации и реализации уникальных решений. Компактность так как отсутствие лишних библиотек и модулей приводит к созданию более легких и производительных приложений, особенно критичных в плане быстродействия.

Из минусов можно отметить низкую скорость разработки так как для использования современных элементов дизайна потребуется потратить время и спроектировать их.

Сравнение подходов:

Критерий	**Разработка с фреймворком**	**Разработка без фреймворка**
Скорость разработки	Быстрее: готовые компоненты, паттерны, инструменты ускоряют создание типовых элементов и функций.	Медленнее: все компоненты и функции требуют разработки "с нуля", что увеличивает время.
Повторяемость кода	Меньше: использование библиотек и паттернов снижает вероятность дублирования кода.	Больше: высока вероятность повторения однотипных фрагментов кода, что усложняет его поддержку.
Масштабируемость	Проще: фреймворки, как правило, хорошо спроектированы для масштабирования сложных приложений.	Сложнее: требует дополнительных усилий и нестандартных решений для обеспечения масштабируемости.
Контроль над кодом	Меньше: разработчик ограничен возможностями и архитектурой фреймворка.	Больше: полный контроль над каждой строкой кода, свобода реализации любых решений.
Размер приложения	Больше: код фреймворка увеличивает размер приложения.	Меньше: отсутствие кода фреймворка приводит к меньшему размеру приложения.

Таблица 3 — Сравнение подходов

Продолжение таблицы 3

Сложность разработки	Проще: использование готовых решений упрощает разработку типовых функций.	Сложнее: требует большего опыта, навыков и самостоятельной реализации многих функций.

Фрэймворк для Web-App:

Для создания Web-App приложения был выбран подход разработки с использованием фреймоврка. Далее необходимо рассмотреть фреймворки для разработки и выбрать самый оптимальный.

1. React [17]

Это JavaScript-библиотека для создания пользовательских интерфейсов, разработанная Facebook. Она фокусируется на компонентном подходе и управлении состоянием.

Плюсы:

- Простота, большая экосистема, гибкость, хорошая производительность.

Минусы:

- Требуется настроить множество инструментов для полноценной работы, как маршрутизация и управление состоянием.

2. Angular [18]

Это полноценный фреймворк для создания веб-приложений, разработанный Google. Он использует TypeScript и включает множество встроенных инструментов.

Плюсы:

- Все в одном фреймворк, строгая структура, поддержка TypeScript, богатая функциональность.

Минусы:

- Большая сложность настройки и использования, относительно большая размерность приложений.

3. Next.js [19]

Это фреймворк для создания React-приложений с серверным рендерингом, разработанный Vercel. Он обеспечивает простую маршрутизацию, статическую генерацию и другие возможности из коробки.

Плюсы:

- Серверный рендеринг, поддержка статической генерации, встроенная маршрутизация, отличные возможности для SEO, простота настройки.

Минусы:

- Некоторые специфические функции могут потребовать глубокого понимания архитектуры Next.js.

Критерий	React	Angular	Next.js
Производительность	Высокая, но зависит от конфигурации	Хорошая, но может быть медленнее	Высокая благодаря SSR и статической генерации
Легкость использования	Легко начать, требует настройки	Крутая кривая обучения	Легко начать, минимальная настройка
Экосистема	Огромная, много инструментов	Большая, все в одном	Огромная благодаря поддержке React и собственным инструментам
Гибкость	Очень гибкий, можно выбрать любые инструменты	Меньше гибкости из-за строгой структуры	Высокая, встроенные лучшие практики

Таблица 4 — Сравнение фреймворков

Вывод:

При выборе фреймворков было проведено сравнение 3 основных фреймворков подходящих под критерии React, Angular, Next.js. Рассмотрим таблицу 4, на основе неё придем к выводу, что больше всего подходит Next.js.

Причины выбора:

1. Серверный рендеринг (SSR):

Next.js поддерживает серверный рендеринг из коробки, что улучшает производительность. Это особенно важно для приложений, которые должны обеспечивать быстрый ответ на запросы.

2. Статическая генерация:

Next.js позволяет генерировать статические страницы во время сборки, что делает приложение еще более быстрым и надежным. Это полезно для страниц с неизменяемым контентом.

3. Простота настройки и использования:

В отличие от Angular, Next.js имеет более пологую кривую обучения и требует минимальной настройки для начала работы. Он предоставляет встроенные инструменты для маршрутизации и управления состоянием, что упрощает разработку.

4. Совместимость с React:

Next.js построен на базе React, что позволяет использовать всю мощь и гибкость React-экосистемы. Вы можете использовать все популярные библиотеки и инструменты, предназначенные для React.

Next.js является подходящим выбором для создания современных веб-приложений благодаря своим возможностям серверного рендеринга, статической генерации и простоте использования. Он предоставляет встроенные инструменты для маршрутизации и управления состоянием, улучшая производительность. Используя Next.js, можно быстро начать разработку и сосредоточиться на создании функциональности, не беспокоясь о настройке и оптимизации инфраструктуры.

Библиотека для работы с Telegram API:

1. node-telegram-bot-api [20]

Это одна из самых популярных библиотек для создания Telegram-ботов на Node.js. Она предоставляет все основные функции для работы с Telegram API.

Плюсы:

- Полная поддержка Telegram API;
- Простота в использовании и настройке;
- Поддержка пользовательские обратные вызовы по HTTP и долгого опроса (long polling).

Минусы:

- Меньше возможностей для расширения функциональности и настройки middleware по сравнению с другими библиотеками.

2. Telegraf.js [21]

Это мощная библиотека для создания Telegram-ботов на Node.js с поддержкой middleware и многих встроенных функций.

Плюсы:

- Простота в использовании и настройке;
- Поддержка middleware, что позволяет легко расширять функциональность;
- Встроенная поддержка популярных функций, таких как сцены и опросы;
- Поддержка вебхуков и долгого опроса.

Минусы:

- Меньше документации и примеров по сравнению с node-telegram-bot-api.

3. telegram-bot [22]

Это простая библиотека для создания Telegram-ботов на Node.js с основными функциями.

Плюсы:

- Простота и легкость использования;
- Поддержка основных функций Telegram API.

Минусы:

- Меньше функциональности и гибкости по сравнению с node-telegram-bot-api и Telegraf.js;
- Ограниченная документация и примеры.

Критерий	node-telegram-bot-api	Telegraf.js	telegram-bot
Простота использования	Высокая	Высокая	Высокая
Поддержка Telegram API	Полная	Полная	Основные функции
Middleware	Ограниченная	Полная	Отсутствует
Расширяемость	Средняя	Высокая	Низкая
Документация	Широкая и подробная	Хорошая, но меньше примеров	Ограниченная
Поддержка вебхуков и долгого опроса	Да	Да	Да
Популярность	Высокая	Средняя, но растет	Низкая

Таблица 5 — Сравнение библиотек для работы с Telegram API

Вывод:

Из таблицы 5 видно все плюсы и минусы данных библиотек. Telegraf.js представляет большую перспективу для разработки. Так как он имеет больший потенциал для развития приложения.

Фреймворк для API:

API (Application Programming Interface) — это интерфейс программирования приложений, который позволяет различным программным приложениям взаимодействовать друг с другом. Он предоставляет набор правил и стандартов, через которые программы могут запрашивать и обмениваться данными.

1. Express.js [23]

Это один из самых популярных фреймворков для создания веб-приложений и API на Node.js. Express.js предоставляет минимальный набор инструментов и высокую гибкость.

Плюсы:

- Очень популярный и широко используемый;
- Простота и гибкость;
- Обширная документация и большое сообщество.

Минусы:

- Может быть медленнее по сравнению с другими фреймворками;
- Нет встроенной поддержки для многих функций (например, валидация запросов).

2. Koa.js [24]

Это современный фреймворк для Node.js, разработанный командой создателей Express. Koa.js предлагает более гибкую и минималистичную архитектуру.

Плюсы:

- Асинхронная архитектура;
- Простота и минимализм;
- Высокая гибкость и возможность настройки middleware.

Минусы:

- Требует больше настроек по сравнению с Express;
- Меньше примеров и документации.

3. Fastify [25]

Это современный и быстрый фреймворк для Node.js, разработанный с упором на производительность и низкое потребление ресурсов. Fastify поддерживает множество плагинов и обеспечивает высокую производительность.

Плюсы:

- Очень высокая производительность;
- Низкое потребление ресурсов;
- Встроенная поддержка валидации запросов и схем;
- Широкая экосистема плагинов.

Минусы:

- Менее распространён и имеет меньше примеров, чем Express.

Критерий	Express.js	Koa.js	Fastify
Простота использования	Высокая	Средняя	Высокая
Производительность	Средняя	Высокая	Очень высокая
Гибкость	Высокая	Очень высокая	Высокая
Поддержка плагинов	Много плагинов	Требует сторонних библиотек	Широкая экосистема плагинов
Документация	Обширная и подробная	Хорошая, но менее обширная	Хорошая, но менее обширная
Сообщество	Очень большое и активное	Среднее	Растущее, но меньшее
Валидация запросов	Отсутствует из коробки	Отсутствует из коробки	Встроенная поддержка
Архитектура	Традиционная	Современная и асинхронная	Современная и асинхронная

Таблица 6 — Сравнение фреймворков для API

Вывод:

На основе таблицы 6 сделаем вывод, что больше всего подходит Fastify.

Причины выбора:

1. Высокая производительность:

Fastify разработан с упором на производительность и является одним из самых быстрых фреймворков для Node.js. Это особенно важно для проектов с высокой нагрузкой и требующих быстрой обработки запросов.

2. Низкое потребление ресурсов:

Fastify эффективно использует системные ресурсы, что позволяет обслуживать больше запросов с меньшими затратами.

3. Встроенная поддержка валидации запросов:

Fastify имеет встроенную поддержку схем для валидации запросов и ответов, что повышает безопасность и надежность API.

4. Широкая экосистема плагинов:

Fastify предоставляет широкий набор плагинов, что упрощает расширение функциональности и интеграцию с другими системами.

5. Современная архитектура:

Fastify использует современные подходы и асинхронную архитектуру, что упрощает разработку и поддержку приложений.

Выбор системы управления базами данных:

СУБД (система управления базами данных) — это специализированное программное обеспечение, предназначенное для создания и управления базами данных. Она обеспечивает доступ к данным, их хранение, манипулирование и обработку в соответствии с определёнными правилами и структурами.

1. MySQL [26]

Это одна из самых популярных реляционных СУБД с открытым исходным кодом. Широко используется в веб-разработке.

Плюсы:

- Широко известна и используется;
- Обширная документация и поддержка;
- Высокая производительность для чтения;
- Поддержка транзакций и сложных запросов.

Минусы:

- Ограниченная гибкость структуры данных;
- Меньше подходит для работы с полуструктурированными данными.

2. MongoDB [27]

Это NoSQL база данных, которая хранит данные в формате JSON-подобных документов. Поддерживает горизонтальное масштабирование и высокую производительность.

Плюсы:

- Высокая производительность при работе с большими объемами данных;
- Гибкость структуры данных;
- Хорошая поддержка горизонтального масштабирования;
- Простота интеграции с приложениями на JavaScript благодаря формату данных BSON/JSON.

Минусы:

- Отсутствие поддержки сложных транзакций (хотя транзакции на уровне коллекций поддерживаются);
- Меньше гарантий целостности данных по сравнению с реляционными СУБД.

3. PostgreSQL [28]

Это мощная реляционная СУБД с открытым исходным кодом, известная своей надежностью и расширяемостью.

Плюсы:

- Поддержка сложных запросов и транзакций;
- Высокая надежность и согласованность данных;
- Расширяемость (можно добавлять пользовательские функции и типы данных);
- Поддержка JSON-документов.

Минусы:

- Сложнее в настройке и управлении по сравнению с другими СУБД;
- Может быть медленнее при больших объемах данных по сравнению с MongoDB.

4. SQLite [29]

Это легковесная реляционная СУБД с открытым исходным кодом, которая встроена в приложение.

Плюсы:

- Простота использования и настройки;
- Нет необходимости в отдельном сервере;
- Отличная производительность для небольших приложений.

Минусы:

- Не подходит для масштабируемых приложений с высокими нагрузками;
- Ограниченные возможности параллелизма и транзакций.

Критерий	MySQL	PostgreSQL	MongoDB	SQLite
Тип данных	Реляционная	Реляционная	Документо-ориентированная	Реляционная
Производительность	Высокая для чтения	Высокая, но может быть медленнее	Высокая при больших объемах данных	Высокая для небольших приложений
Гибкость структуры данных	Ограниченная	Высокая (включая JSON)	Очень высокая	Ограниченная
Поддержка транзакций	Полная поддержка	Полная поддержка	Поддержка транзакций на уровне коллекций	Ограниченная
Масштабируемость	Вертикальная	Вертикальная и горизонтальная	Горизонтальная	Ограниченная
Простота использования	Средняя	Сложнее, чем MySQL	Высокая	Очень высокая
Сообщество и поддержка	Большое и активное	Большое и активное	Большое и активное	Среднее
Поддержка JSON	Ограниченная	Полная	Полная	Ограниченная

Таблица 7 — Сравнение СУБД

Вывод:

На основе таблицы 7 сделаем вывод, что больше всего подходит MongoDB.

Критерии выбора:

1. Гибкость структуры данных:

MongoDB позволяет хранить данные в формате JSON-подобных документов, что обеспечивает высокую гибкость в структуре данных и позволяет легко изменять схему без необходимости миграции данных.

2. Высокая производительность:

MongoDB обеспечивает высокую производительность при работе с большими объемами данных и поддерживает индексацию для ускорения операций поиска.

3. Горизонтальное масштабирование:

MongoDB поддерживает горизонтальное масштабирование, что позволяет легко добавлять новые серверы и распределять данные для повышения производительности и отказоустойчивости.

4. Простота интеграции с JavaScript:

Благодаря использованию JSON-подобного формата данных (BSON), MongoDB легко интегрируется с приложениями на JavaScript, что упрощает обмен данными между сервером и клиентом.

5. Экосистема и инструменты:

MongoDB предоставляет множество инструментов для управления и мониторинга базы данных, таких как MongoDB Atlas для облачного хостинга и MongoDB Compass для визуализации данных.

Вывод по главе 2

Во второй главе работы были определены ключевые функциональные и нефункциональные требования к разрабатываемому приложению. Акцент был

сделан на обеспечение гибкости и возможности легкой интеграции новых функциональных компонентов в будущем. Для этого была разработана и детально описана архитектура приложения, учитывающая текущие требования и потенциальные изменения в будущем.

Особое внимание было уделено выбору основных технологий для реализации приложения. В качестве основных кандидатов рассматривались фреймворки React, Next.js и Angular. Каждый из них был проанализирован с точки зрения их преимуществ и возможных ограничений, чтобы выбрать наиболее подходящий инструмент для конкретного проекта.

Таким образом, в данной главе были заложены основы для успешной разработки и дальнейшей интеграции приложения, обеспечивая его устойчивость к изменениям и способность эффективно реагировать на новые требования пользователей и рынка.

3. ПРОЕКТИРОВАНИЕ БОТА

В предыдущей главе определили основные требования и подход к разработке. Определились с используемыми технологиями разработки, а также рассмотрели фреймворки и библиотеки для разработки приложения.

3.1.Используемые технологии

Для данного проекта были выбраны такие технологии:

- Язык программирования: Java Script;
- Подход для разработки Web-App: С использованием фрэймворка;
- Фрэймворк для Web-App: Next.js;
- Библиотека для работы с Telegram API: Telegraf.js;
- Фреймворк для API: Fastify;
- Система управления базами данных: MongoDB;

3.2.Проектирование компонентов telegram-бота

Для разработки Telegram-бота с интеграцией сервиса доставки продуктов понадобится разработать несколько компонентов.

Компоненты:

1. API для работы с базой данных.
2. Чат-бот.
3. Web-App приложение.

У данных систем есть общие правила и зависимости в бизнес процессах. Для этого была разработана DFD-модель, которая представлена на Рисунках 3, 4. На данных рисунках отображены сущности, взаимодействующие с Telegram-ботом. А также потоки данных исходящие от этих сущностей. Так на Рисунке 4 представлена декомпозиция 2 уровня, где описаны более подробно процессы внутри Telegram-бота.

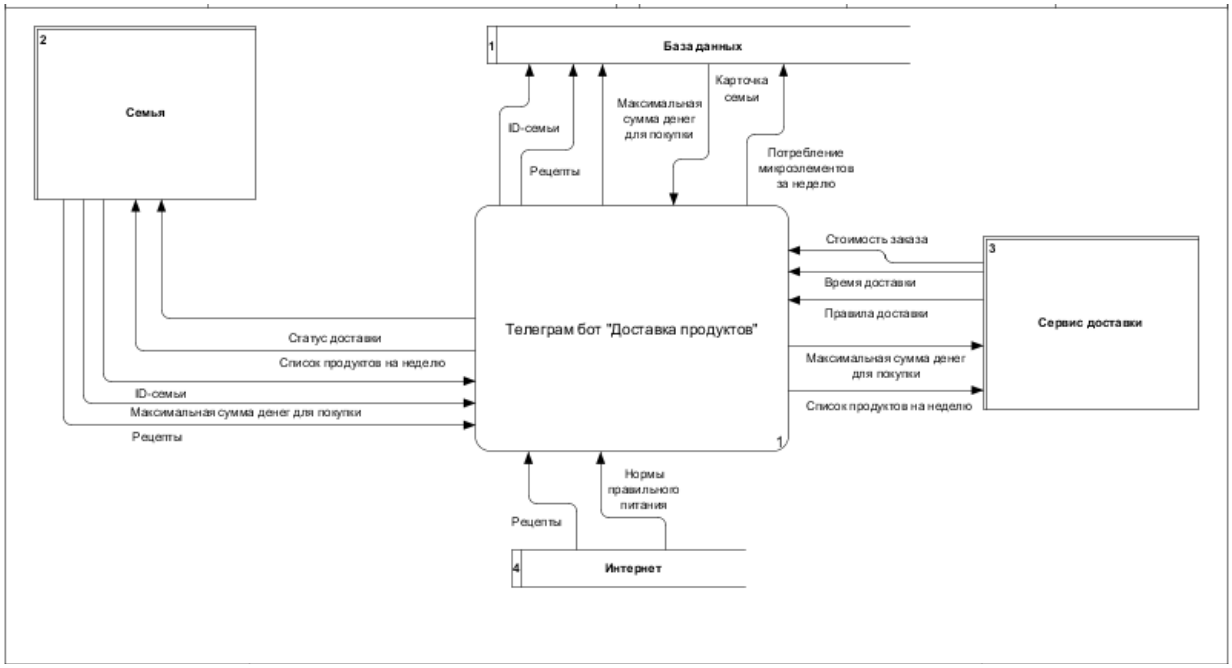

Рисунок 3 — DFD-модель Telegram-бота

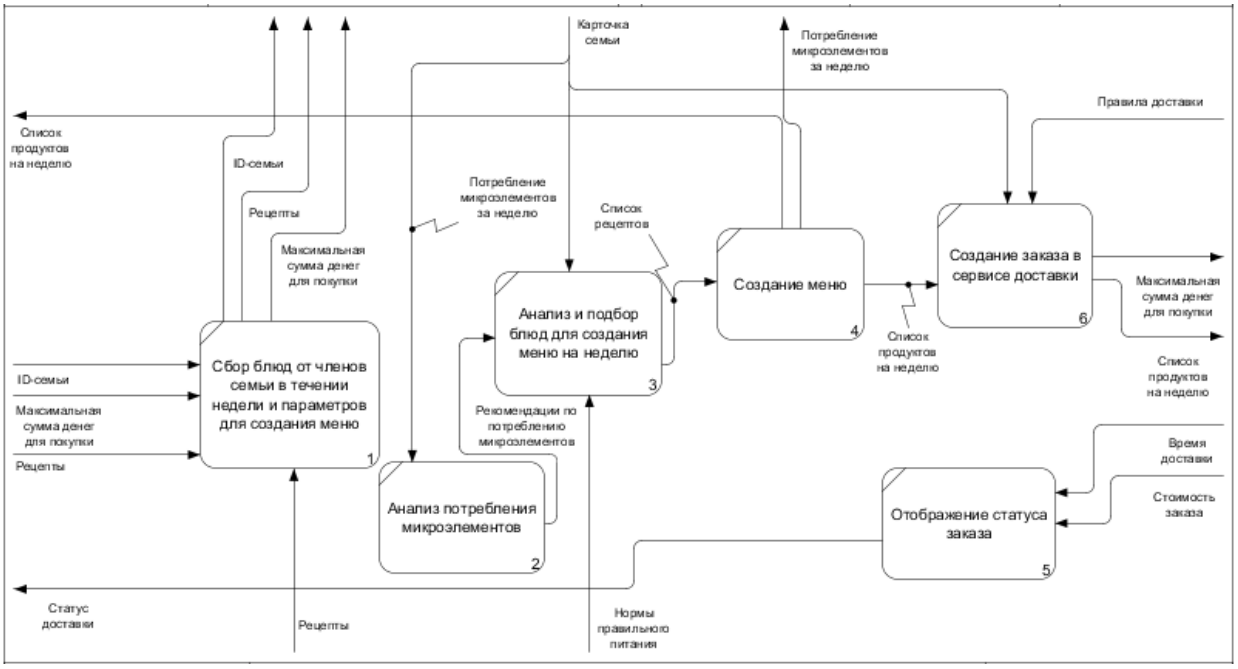

Рисунок 4 — DFD-модель Telegram-бота

3.3.Проектирование системы

После того как определились с основными сущностями, взаимодействующими с Telegram-ботом. А также определившись с данными которыми будет оперировать Telegram-бот, следует перейти к разработке

необходимых частей для работы приложения. Для этого построим UML-схему (Рисунок 5).

3.3.1. Определение необходимых сервисов для работы telegram-бота

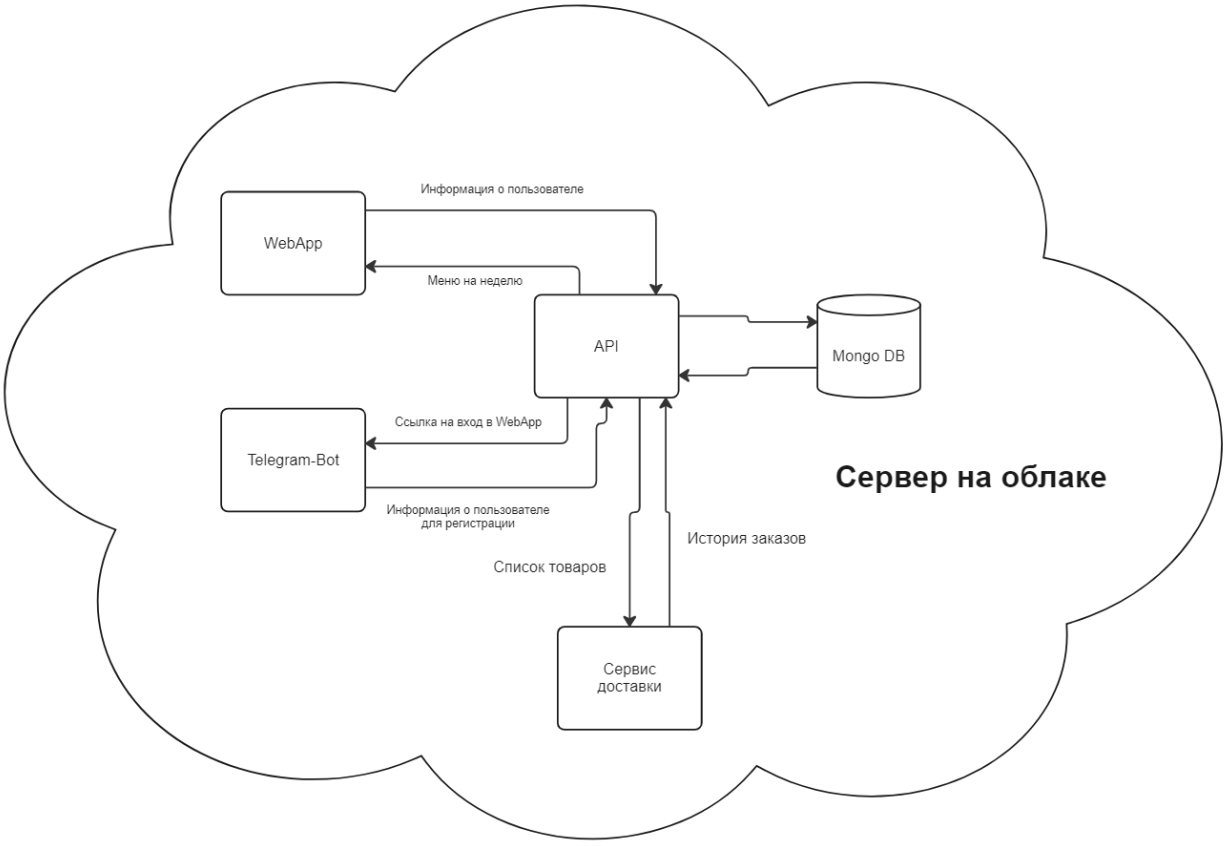

Рисунок 5 — Схема архитектуры необходимых сервисов для работы приложения

Из схемы видно, что для реализации данной системы необходим следующий набор подсистем:

1. API для работы с базой данных:

   Обоснование: Этот уровень необходим для правильной работы Telegram-бота и защищённости данных. Так как он отвечает за получение, обработку, хранение и отправку данных для работы Telegram-бота и Web-App приложения.

2. Сервис работы с Telegram API и ботом:

   Обоснование: Этот уровень необходим для работы Telegram-бота, он обрабатывает запросы от пользователя и общается с API для работы с базой данных.

3. Веб сайт с Web-App приложением:

   Обоснование: Данный уровень был рассмотрен в предыдущей главе и обоснован. Он позволяет улучшить взаимодействие и понятность использования приложения клиентом.

4. MongoDB база данных, обеспечивающая хранение данных о пользователях, их семьях и меню на неделю:

   Обоснование: Выбор СУБД MongoDB было обосновано в предыдущей главе. Это самый подходящий выбор для работы данного приложения. Он удовлетворяет всем условиям хранения данных.

Рассмотрим в разработку каждой системы по-отдельности.

3.3.2. Проектирование API для работы с базой данных.

Ранее было определён фрэймворк для разработки API это Fastify. По причине того, что он очень быстро обрабатывает запросы и в тестах показывает очень хорошую производительность 30000 запросов в секунду [17].

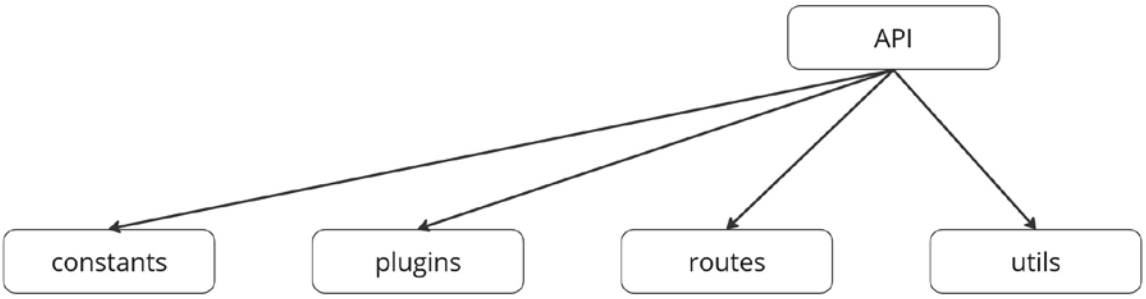

Рисунок 6 — Структурная схема проекта API

Данная структура организации проекта на рисунке 6 позволяет оптимизировать работу, упростить поиск проблемных мест, а также упростить работу с той или иной частью API.

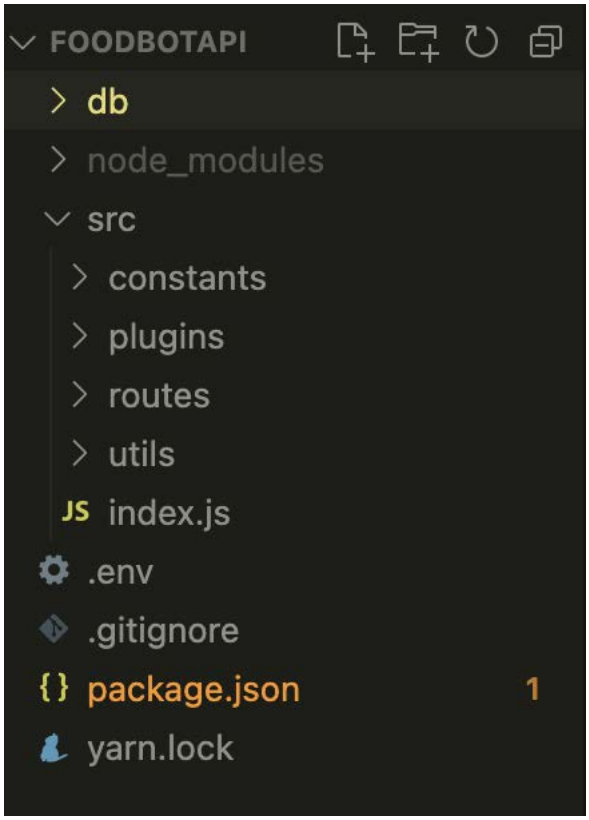

Рисунок 7 — Скриншот организации проекта в среде разработки VS Code

Значение папок (Рисунок 7):

- constants — здесь размещаются все необходимые константы для проекта;
- plugins — здесь размещаются все необходимые обработчики по типу CORS, обработчик .env файла, swagger;
- routes — здесь размещаются все необходимые пути и запросов, а также обработчики запросов;
- utils — здесь размещаются все необходимые функции для обработки и работы с информацией;
- index.js — это основной файл API;

Листинг 1 — Листинг файла index.js

```
import Fastify from "fastify";
import dbConnector from "./utils/db.js";
import routes from "./routes/index.js";
import plugins from "./plugins/index.js";

import { HOST, PORT } from "./constants/env.js";

const fastify = Fastify({
  logger: true,
});
// Добавление модуля подключения к БД
fastify.register(dbConnector);
fastify.register(plugins); // Добавление модуля с плагинами
fastify.register(routes);  //  Добавление обработчиков ответов на запросы

const start = async () => {
  try {
    await fastify.listen({ host: HOST, port: PORT });

    console.log(`🚀 Server 1.0.0 ready at: ${HOST}:${PORT}`);
  } catch (err) {
    fastify.log.error(err);
    process.exit(1);
  }
};

start();
```

3.3.3. Проектирование сервиса работы с telegram api и ботом

Для взаимодействия с Telegram-ботом потребуется сервис, который будет обрабатывать запросы и отвечать пользователям, а также взаимодействовать с API который был описан выше. Для этого был определён фреймворк для работы с Telegram API это Telegraf. Чтобы продолжить разработку необходимо спроектировать и понять какое взаимодействие с ботом будет происходить для этого построим диаграмму (Рисунок 8).

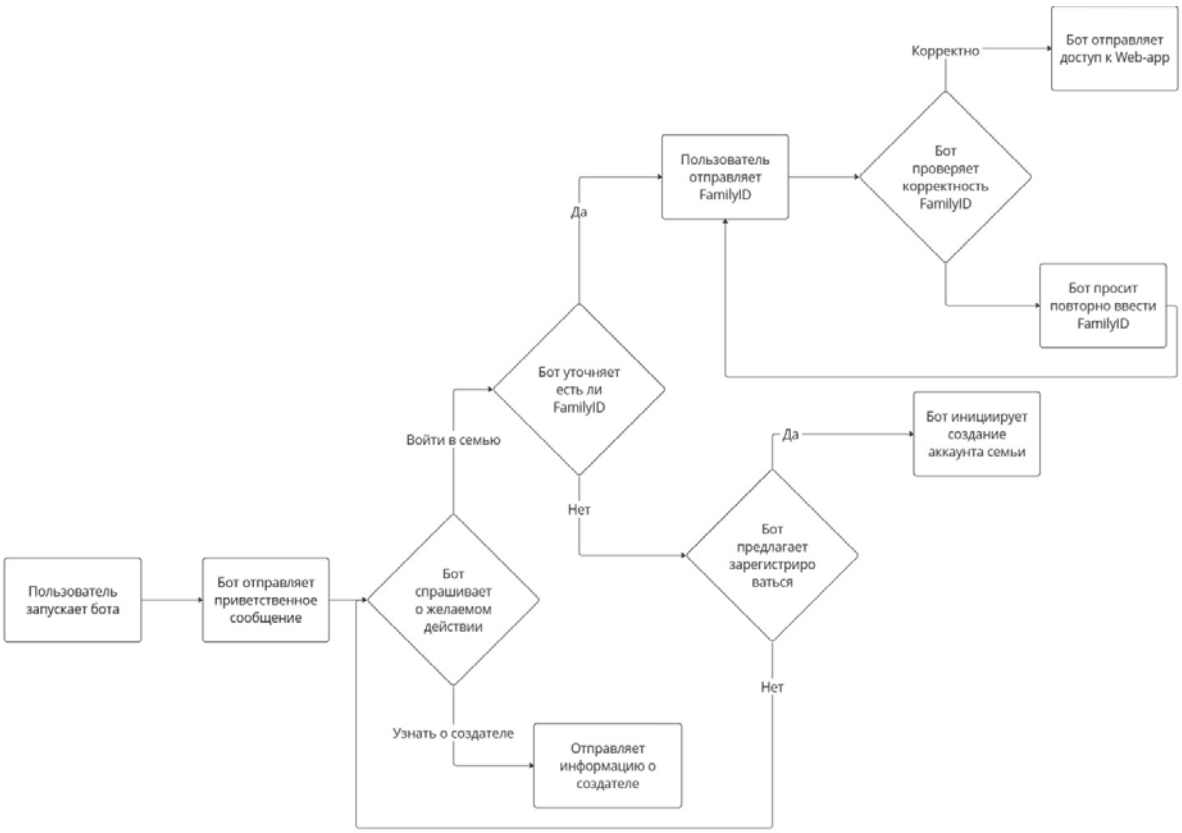

Рисунок 8 — Диаграмма потока данных работы Telegram-бота

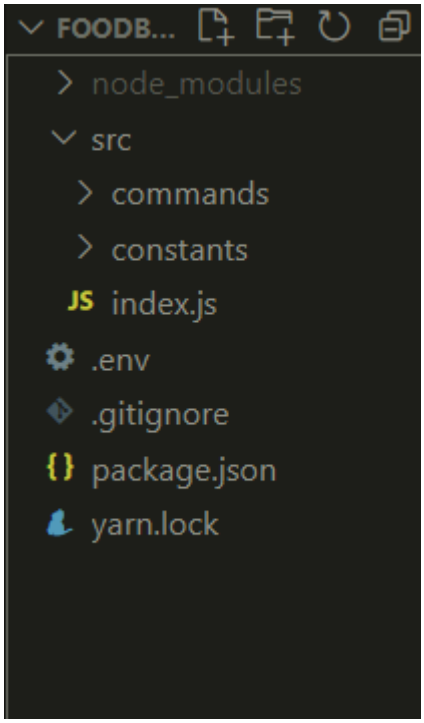

Рисунок 9 — Скриншот организации проекта в среде разработки VS Code

На рисунке 9 отражена организация проекта, данная структура позволяет легко масштабировать проект и добавлять новые функции для бота.

Описание организации:

- commands — в данной папке размещаются все обработчики команд для бота;

Листинг 2 — Представление команды бота /start

```
import { helloMessage } from "../../constants/text.js";
import { Markup } from "telegraf";
// Команда Start в боте запускает работу Telegram-бота
export const start = (bot) => {
  bot.start(async (ctx) => {
     // Ответ бота на запрос содержащий приветственное сообщение //
     и клавиатуру с ответами
    await ctx.reply(helloMessage, keyboard);
  });
};

// Объект описывающий Клавиатуру ответов
const keyboard = Markup.inlineKeyboard([
  Markup.button.callback("Вход в семью", "sign-in"),
  Markup.button.callback("Кто твой автор?", "creator"),
]);
```

- constants — в данной папке размещаются необходимые константы для работы бота в том числе и текста ответов;

Листинг 3 — Пример констант

```
export const helloMessage = `
Приветствую! Вы запустили FamilyFood Bot!

Я помогу вам сделать вашу семью более здоровой
и с составленным меню)
`;

export const aboutCreatorText = `
Мой создатель:
Выпускник кафедры ЭВМ ЮУрГУ, frontend-разработчик на React, React-
Native и просто классный парень)
`;

export const helpIdText = `
ID-семьи - это уникальный номер вашей семьи, который позволяет вам
войти в аккаунт семьи или добавить новых членов семьи.
`;
```

В листинге 3 представлен текст, используемый в ответах бота. В дальнейшем можно расширить и усовершенствовать бот, добавив библиотеку i18n, интернационализации.

3.3.4. Проектирование веб сайта с web-app приложением

Выше были определены технологии для разработки Web-App приложения. Это фреймворк Next.js и библиотека компонентов Next.js UI-Kit. Далее для продолжения разработки необходимо спроектировать интерфейс приложения. Для этого воспользуемся Figma. И спроектируем макеты экранов для приложения.

Нюансы:

- Главный экран должен включать кнопки для перехода по странницам;
- С каждого экрана должен быть доступ к любому экрану посредством выпадающего меню.

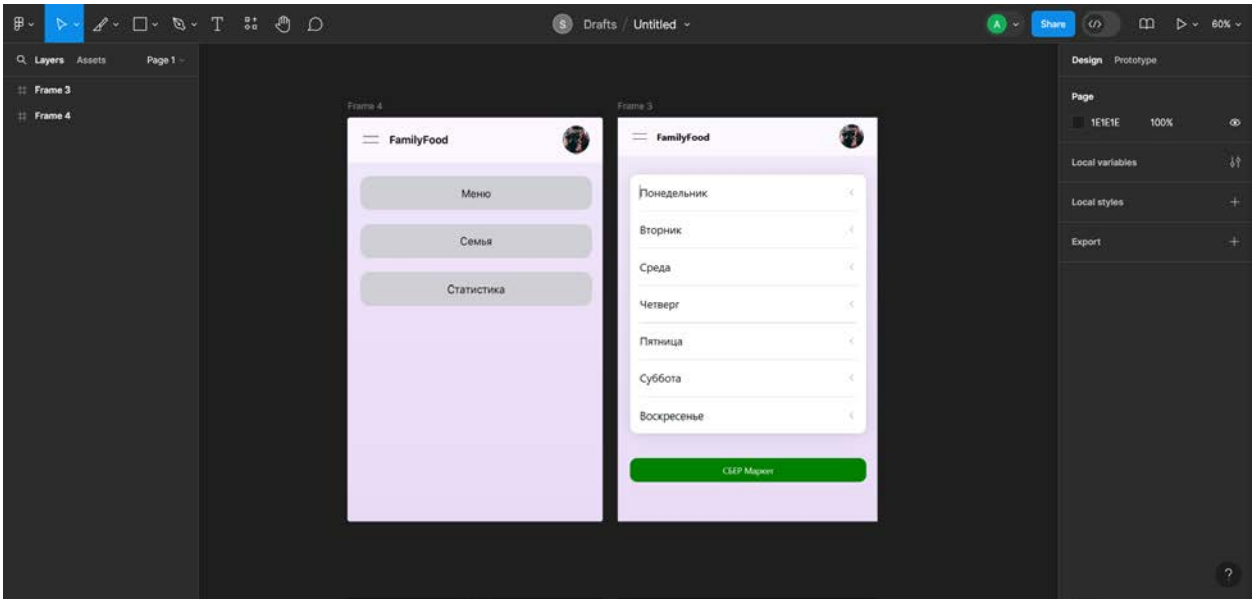

Рисунок 10 — Скриншот среды разработки интерфейса Figma

На рисунке 10 представлены часть разработанных экранов приложения в Figma.

Рисунок 11 — Представление элемента навигации, позволяющий дать доступ к любому экрану посредством выпадающего меню.

Рисунок 11 отражает представление элемента интерфейса для навигации по экранам.

3.3.5. Проектирование моделей базы данных на MongoDB

Чтобы хранить данные необходимо спроектировать и описать схемы данных. Для этого создадим схему, в которой будет указаны основные таблицы данных.

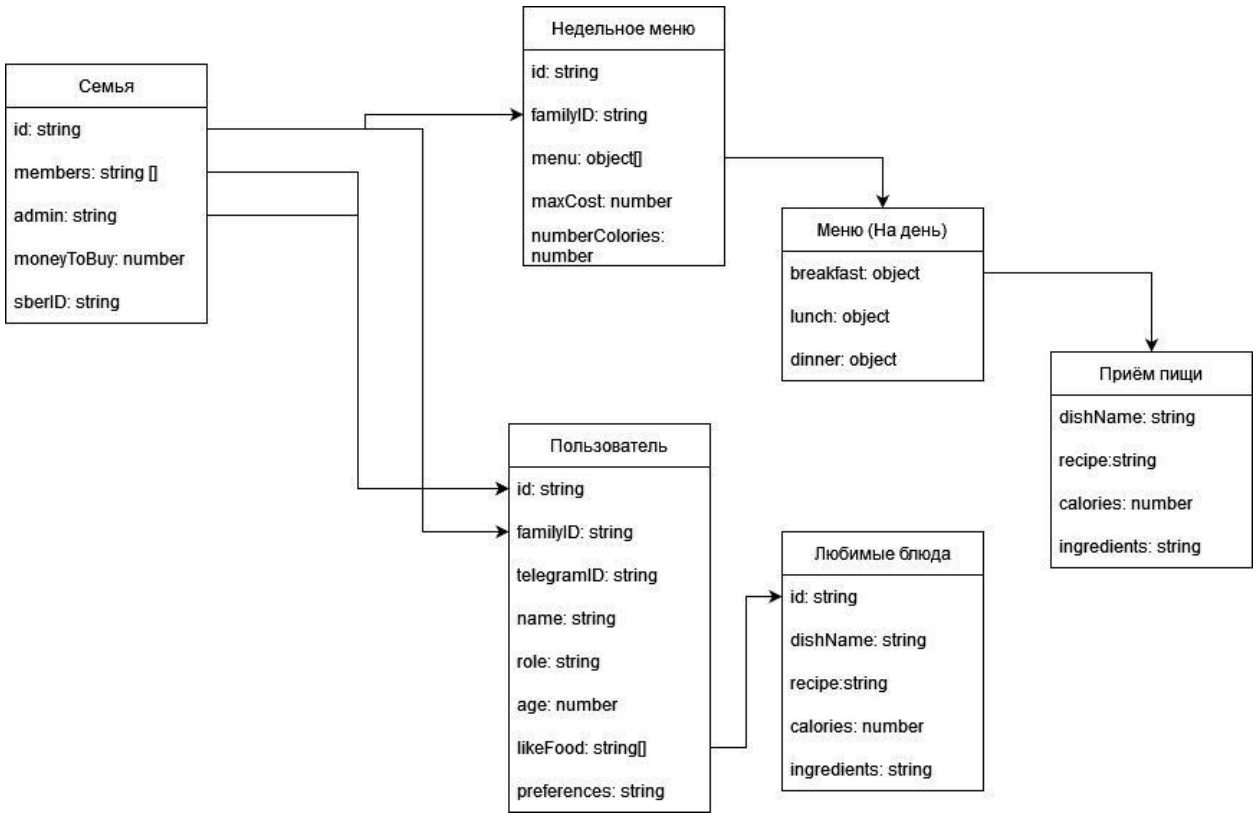

Рисунок 12 — Схема данных в MongoDB

Из рисунка 12 видно, что данные хранятся в нескольких таблицах.

Описание таблиц:

- Семья: в данной таблице хранятся аккаунты семей пользователей, в ней хранятся члены семьи, кто главный, сумма денег на неделю для покупки продуктов;
- Пользователь: в данной таблице хранятся все пользователи приложения, их familyID для связи и поиска семьи в таблице, telegramID для верификации, и основные данные о пользователе.
- Недельное меню: в данной таблице располагаются все меню для семей на неделю;
- Любимые блюда: в данной таблице хранятся данные о блюдах и рецептах, которые добавил пользователь.

Вывод по главе 3

Третья глава дипломной работы посвящена проектированию Telegram-бота. Основное внимание уделено проектированию API для работы с базой данных, его организации и строения. Спроектированы таблицы базы данных для MongoDB, описаны типы данных и взаимосвязи между таблицами.

Описаны схемы организации данных. Спроектирована DFD-модель, отражающая сущности, взаимодействующие с приложением, а также потоки данных. Всего было разработано три схемы и три диаграммы.

Описана схема необходимых для работы сервисов Telegram-бота, которая позволила понять из каких составных частей состоит приложение.

Важной частью работы стало проектирование элементов интерфейса, которые напрямую влияют на удобство и функциональность использования приложения.

4. РЕАЛИЗАЦИЯ И ПРОВЕДЕНИЕ ТЕСТИРОВАНИЯ РАБОТОСПОСОБНОСТИ БОТА

В данном разделе рассмотрим реализацию приложения и взаимодействие с ним.

4.1.Реализация Telegram бота

Сейчас для открытия бота можно использовать ссылку https://t.me/family_food_bot. Она ведёт на чат с ботом. На рисунке 13 представлен экран, который встречает пользователя.

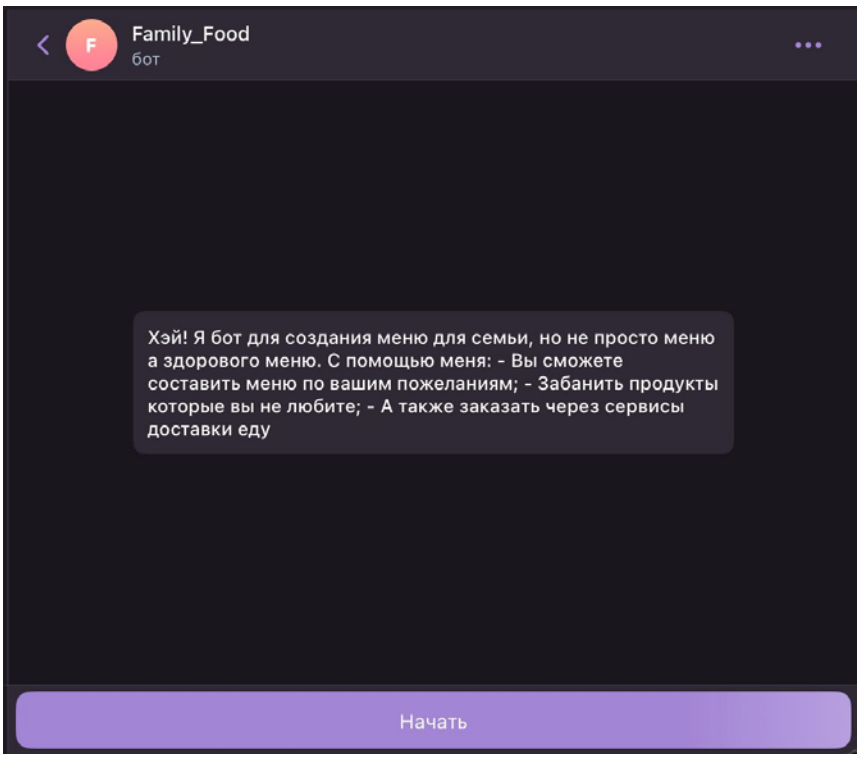

Рисунок 13 — Экран при открытии бота.

При нажатии на кнопку “Начать” происходит начало работы бота.

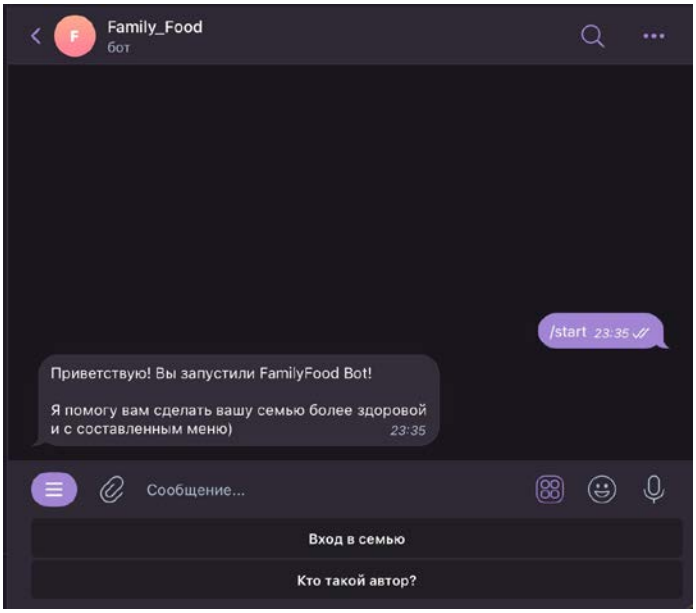

Рисунок 14— Начало работы бота

Мы видим, что пользователю предоставляются варианты выбора для удобства использования приложения (Рисунок 14).

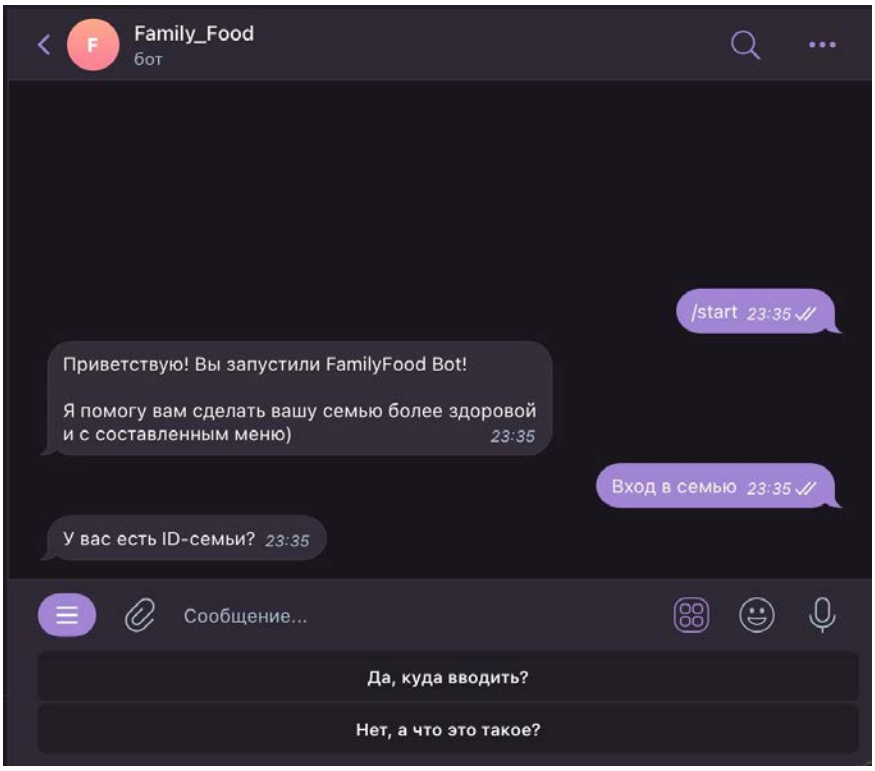

Рисунок 15— Вход через Family-ID

Далее пользователь, который прошёл процедуру регистрации вводит ID-семьи для входа или присоединения в аккаунт (Рисунок 15).

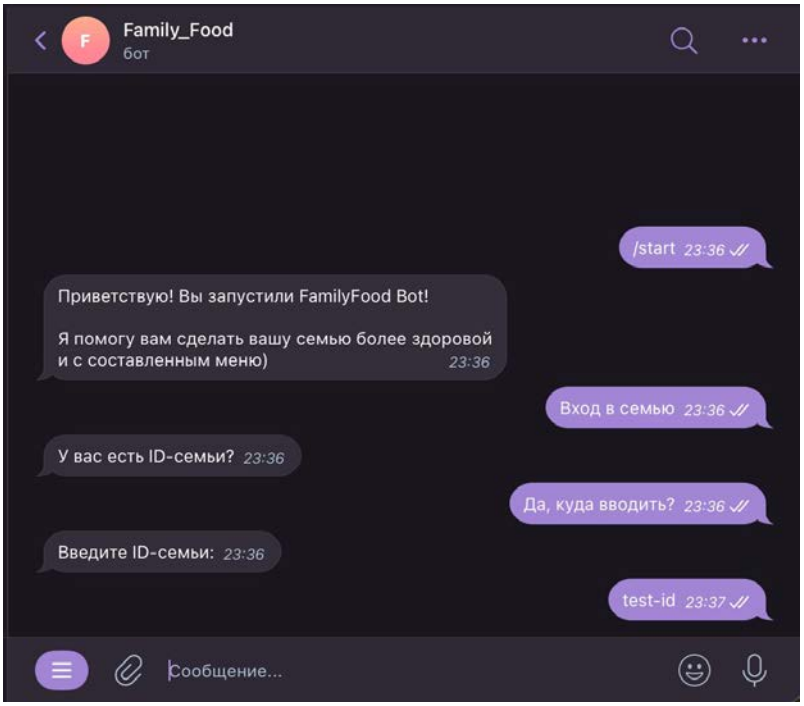

Рисунок 16 — Пользователь вошёл в аккаунт

Далее бот переводит пользователя в Web-App приложение (Рисунок 16).

4.2.Реализация web-app приложение

Для повышения удобства использования было решено воспользоваться возможностью, добавленной в одном из последних обновлений Telegram для подключения Web-App приложения. Пользователи привыкли использовать приложение на своих мобильных устройствах, а чат ограничивает функционал приложения.

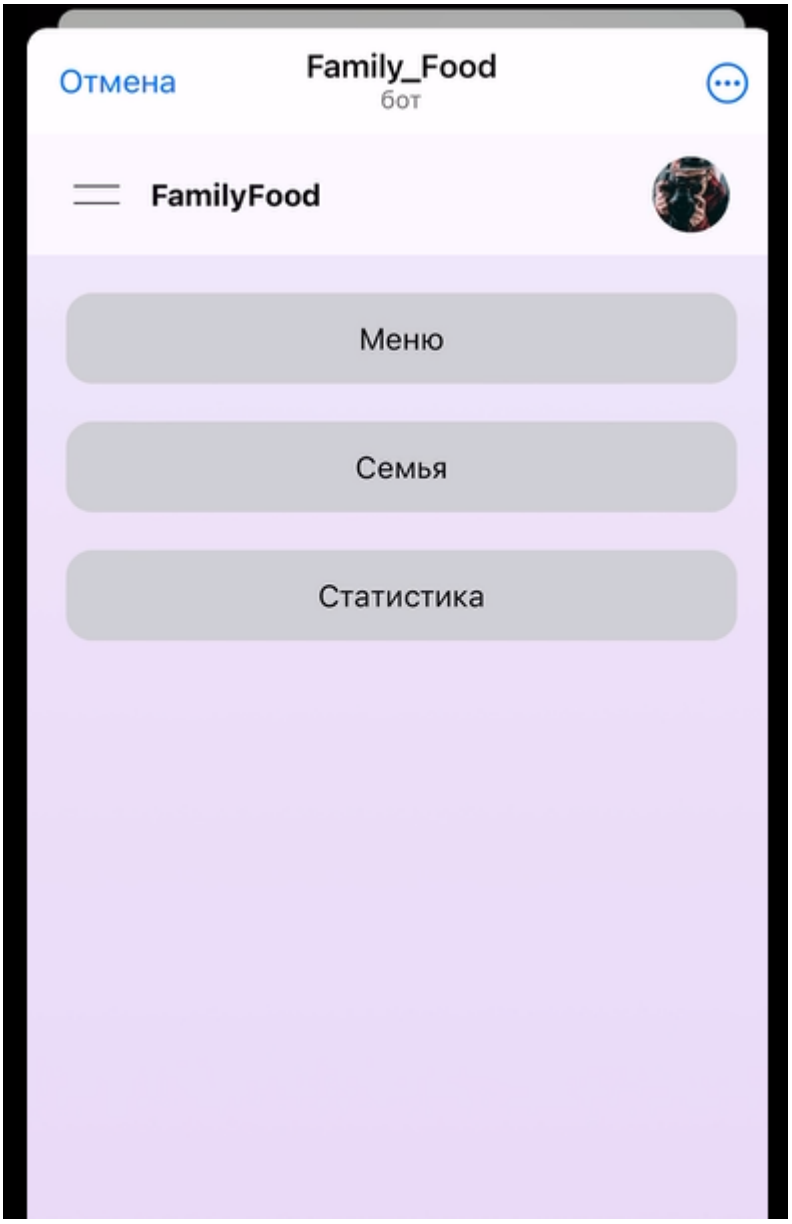

Рисунок 17 — Главное меню приложения

При открытии Web-App приложения пользователя встречает меню с выбором вкладок. Первая — это меню на неделю в ней отображается меню по дням недели с блюдами и рецептами на неделю (Рисунок 17).

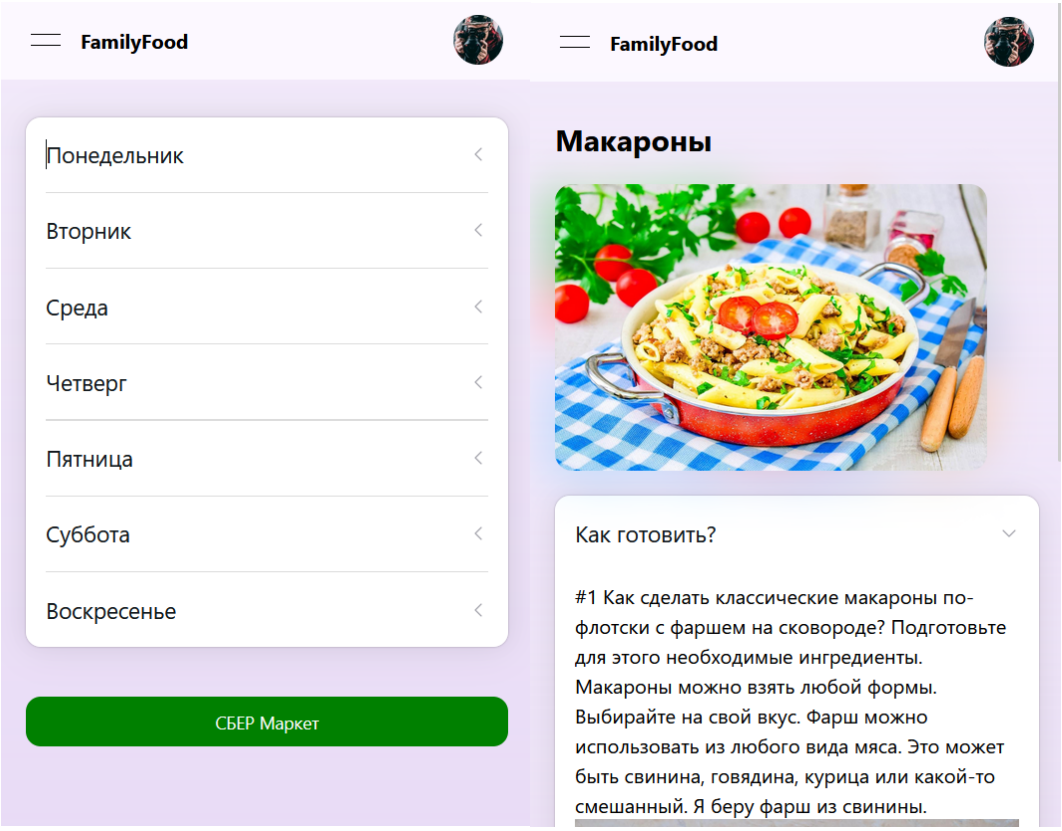

Рисунок 18 — Недельное меню и карточка блюда

Далее можно увидеть карточку блюда в которой представлено название, картинка блюда и рецепт для готовки на рисунке 18.

Рисунок 19 — Листинг карточки блюда

На рисунке 19 представлен листинг карточки блюда. Также тут имеется рецепт для приготовления блюда.

4.3Тестирование

В рамках тестирования приложения, проведем функциональное тестирование. Существует большое количество различных видов тестирования продуктов, но для тестирования данного приложения выберем функциональное тестирование, так как оно проверяет работу функция приложения. Функциональное тестирование помогает убедиться, что каждая функция приложения выполняет заданные требования и корректно реагирует на разнообразные входные данные.

Преимущества функционального тестирования включают его простоту и эффективность в выявлении ошибок в поведении приложения, что делает его идеальным выбором для проверки Telegram-бота. Это позволяет обеспечить, что приложение и его системы будут работать так, как ожидается от них, предоставляя пользователю корректную и последовательный отклик [30].

Для тестирования развернём проект на персональном компьютере, конфигурация которого включала в себя процессор Intel Core i5 с тактовой частотой 2.40 ГГц, 8 гигабайт оперативной памяти DDR4 и жесткий диск емкостью 500 гигабайт. Для работы Web-App настроем ngrok. Ngrok — это сервис который открывает доступ к внутренним ресурсам машины, на которой он запущен, из внешней сети, путем создания публичного адреса, все запросы на который будут переброшены на локальный адрес и заданный порт [31]. После завершения процесса развертывания и настройки можно переходить к тестированию.

4.3.1 Функциональное тестирование

Функция	**Результат**	**Тест пройден?**
Регистрация пользователя	Пользователь успешно регистрируется и получает доступ к Web-App	Да
Добавление блюда	Пользователь успешно добавил блюдо	Да
Получение недельного меню	Пользователь успешно получил недельное меню с рецептами	Да
Интеграция с сервисом доставки	Пользователь успешно перешёл на сервис доставки и прошёл аутентификация	Да

Таблица 8 — Результаты функционального тестирования

По результатам функционального тестирования (Таблица 8) можно сделать вывод, что тестирование пройдено. Приложение готово к работе.

Вывод по главе 4

Четвертая глава посвящена разработке и тестированию Telegram-бота с интеграцией сервиса доставки. В рамках этой главы было разработано приложение.

Telegram-бот и Web-App приложение позволило создавать меню и делать жизнь более здоровой, а так же разнообразной. Для приложения были проведены функциональные тесты, результаты которых подтверждают, что заданные цели достигнуты, а функции Telegram-бота с интеграцией сервиса доставки работает корректно. Это подтверждается успешной реализацией и проверкой приложения, таких как авторизация в сервисе и получения недельного меню.

ЗАКЛЮЧЕНИЕ

Дипломная работа была направлена на разработку и проектирование системы, включающей в себя Telegram-бота с интеграцией сервиса доставки продуктов. В процессе выполнения работы были достигнуты следующие ключевые результаты:

- Определение требований: Были сформулированы функциональные и нефункциональные требования к системе, определены ключевые сценарии использования и установлены показатели производительности, масштабируемости и надежности. Это обеспечило четкое понимание того, каким должен быть конечный продукт и какие задачи он должен решать.
- Выбор технологий: Обоснован выбор технологий для разработки, включая язык программирования JavaScript и фреймворки (React, Next.js, Angular). Было принято решение использовать библиотеку Telegraf для написания кода Telegram-бота, что позволило обеспечить гибкость и возможность расширения функционала. Также рассмотрены подходы к разработке web-приложений с использованием фреймворков, что позволило ускорить процесс разработки и повысить качество продукта.
- Проектирование системы: Разработаны и описаны компоненты системы, включая API для работы с базой данных, чат-бот и web-приложение. Спроектированы DFD-модель и UML-схема, визуализирующие структуру и взаимодействие основных частей системы. Определены таблицы базы данных для MongoDB, описаны типы данных и взаимосвязи между таблицами, что обеспечивает целостность и согласованность данных в системе.

- Интерфейс и пользовательский опыт: Спроектированы основные элементы интерфейса, учитывающие удобство и функциональность использования приложения, что напрямую влияет на пользовательский опыт.

В результате проделанной работы была создано представление системы Telegram-бота с интеграции системы доставки. Основные решения по выбору технологий и подходов к разработке обеспечивают высокое качество и гибкость системы, что позволит ей эффективно выполнять поставленные задачи и удовлетворять потребности пользователей. Вся структура и архитектура системы спроектированы таким образом, чтобы обеспечить её масштабируемость, надежность и возможность легкой интеграции новых функциональных компонентов в будущем.

С помощью интеграции с сервисами доставки можно построить модель монетизации приложения и впоследствии получении прибыли от приложения. Ведь сервисы доставки работают на условиях привлечения клиентов и если мы перенаправили клиента на сервис доставки, и он совершил покупку, то нам будет начислена сумма или же процент от покупки клиента. Также можно рассмотреть модель монетизации с помощью размещения рекламы продуктов.

Таким образом, выполненная работа создала надежную основу для разработки и внедрения Telegram-бота с интеграции системы доставки, которая поможет улучшить организацию питания семей, предоставляя удобные и современные инструменты для планирования меню и заказа продуктов.

БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1. Гапанович, И. Г. (2018). Современное восприятие времени и ускорение темпа жизни. Социология: теория, методы, маркетинг, 4, 60-71.
2. Лопатина, И. В., & Суслов, А. Н. (2018). Влияние изменений в сфере труда на восприятие времени и структуру досуга. Вестник Российского университета дружбы народов. Серия: Социология, 18(1), 94-108.
3. "5 лучших приложений для планирования меню на неделю" [Электронный ресурс]. — URL: https://sgastronomy.ru/5-luchshih-prilozheniy-planirovaniya-menu-na-nedelu/ (дата обращения: 20.12.2023).
4. Приложение "Календарь рецептов" в App Store. [Электронный ресурс]. — URL: https://apps.apple.com/ru/app/%D0%BA%D0%B0%D0%BB%D0%B5%D0%BD%D0%B4%D0%B0%D1%80%D1%8C-%D1%80%D0%B5%D1%86%D0%B5%D0%BF%D1%82%D0%BE%D0%B2/id1329462150?platform=iphone (дата обращения: 20.12.2023).
5. Приложение "Weekly Meal Planner Calendar" в App Store [Электронный ресурс]. — URL: https://apps.apple.com/us/app/weekly-meal-planner-calendar/id1619509620?platform=iphone (дата обращения: 20.12.2023).
6. Shevat, A. (2017). Designing Bots: Creating Conversational Experiences. Китай: O'Reilly Media.
7. Python [Электронный ресурс]. — URL: https://docs.python.org/3/ (дата обращения: 20.12.2023).
8. Node.js [Электронный ресурс]. — URL: https://nodejs.org/docs/latest/api/ (дата обращения: 20.12.2023).
9. Асинхронность Node.js [Электронный ресурс]. — URL: https://nodejs.org/en/learn/asynchronous-work/javascript-asynchronous-programming-and-callbacks (дата обращения: 20.12.2023).

10. PHP [Электронный ресурс]. — URL: https://www.php.net/manual/ru/ (дата обращения: 20.12.2023).
11. Go [Электронный ресурс]. — URL: https://go.dev/ (дата обращения: 20.12.2023).
12. Ruby [Электронный ресурс]. — URL: https://ruby-doc.org (дата обращения: 20.12.2023).
13. Интеграция с Telegram [Электронный ресурс]. — URL: https://developers.sber.ru/help/salutebot/telegram-integration (дата обращения: 20.12.2023).
14. Telegram API [Электронный ресурс]. — URL: https://core.telegram.org/api (дата обращения: 20.12.2023).
15. Telegraf.js [Электронный ресурс]. — URL: https://telegraf.js.org/ (дата обращения: 20.12.2023).
16. Telegram Web Apps [Электронный ресурс]. — URL: https://core.telegram.org/bots/webapps#recent-changes (дата обращения: 20.12.2023).
17. React [Электронный ресурс]. — URL: https://ru.react.dev/learn (дата обращения: 20.01.2024).
18. Angular [Электронный ресурс]. — URL: https://angular.dev/overview (дата обращения: 20.01.2024).
19. Next.js [Электронный ресурс]. — URL: https://nextjs.org/docs (дата обращения: 20.01.2024).
20. node-telegram-bot-api [Электронный ресурс]. — URL: https://www.npmjs.com/package/node-telegram-bot-api (дата обращения: 20.01.2024).
21. Telegraf.js [Электронный ресурс]. — URL: https://telegraf.js.org/ (дата обращения: 20.01.2024).

22. telegram-bot [Электронный ресурс]. — URL: https://www.npmjs.com/package/telegram-bot (дата обращения: 20.01.2024).
23. Express.js [Электронный ресурс]. — URL: https://expressjs.com/ru/ (дата обращения: 20.12.2023).
24. Koa.js [Электронный ресурс]. — URL: https://koajs.com/ (дата обращения: 20.01.2024).
25. Fastify [Электронный ресурс]. — URL: https://fastify.dev/ (дата обращения: 20.01.2024).
26. MySQL [Электронный ресурс]. — URL: https://dev.mysql.com/doc/ (дата обращения: 20.01.2024).
27. MongoDB [Электронный ресурс]. — URL: https://www.mongodb.com/docs/v2.2/ (дата обращения: 20.01.2024).
28. PostgreSQL [Электронный ресурс]. — URL: https://www.postgresql.org/ (дата обращения: 20.01.2024).
29. SQLite [Электронный ресурс]. — URL: https://sqlite.org/src4/doc/trunk/www/index.wiki (дата обращения: 20.01.2024).
30. Бейзер Б. Тестирование черного ящика: Технологии функционального тестирования программного обеспечения и систем. // Питер. 2004. – 320 с.
31. Ngrok [Электронный ресурс]. –URL: https://ngrok.com/ (дата обращения: 20.12.2023).