Министерство науки и высшего образования РФ
Федеральное государственное автономное
образовательное учреждение высшего образования
«СИБИРСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Институт космических и информационных технологий

Кафедра вычислительной техники

УТВЕРЖДАЮ
Заведующий кафедрой
_____О.В. Непомнящий
«___»_____2024 г.

БАКАЛАВРСКАЯ РАБОТА

090301 Информатика и вычислительная техника

Мобильное приложение – трекер дня. Клиентская часть.

Руководитель _____ _____ доцент, канд. техн. наук С.Н. Титовский
подпись *дата* *должность, ученая степень*

Выпускник _____ _____ И.С. Бахтеева
подпись *дата*

Нормоконтролёр _____ _____ доцент, канд. техн. наук С.Н. Титовский
подпись *дата* *должность, ученая степень*

Красноярск 2024

РЕФЕРАТ

Выпускная квалификационная работа по теме «Мобильное приложение – трекер дня. Клиентская часть.» содержит 38 страниц текстового документа, 21 рисунок, 2 таблицы и 9 использованных источников.

МОБИЛЬНОЕ ПРИЛОЖЕНИЕ, ТРЕКЕР ДНЯ, КЛИЕНТСКА ЧАСТЬ, ИНТЕРФЕЙС, АНДРОЙД

Цель работы — Разработать клиентскую часть мобильного приложения.

Задачи работы:

-анализ предметной области;

-проектирование пользовательского интерфейса;

-проектирование базы данных;

-реализация проекта.

В первой главе проводится анализ предметной области, включающий обзор существующих решений, таких как популярные мобильные приложения "To-do List", "TickTime" и "Tusk", и их сравнительный анализ, а также определение функциональных требований к мобильному приложению.

Вторая глава посвящена проектированию приложения: здесь рассматриваются диаграммы прецедентов, структура базы данных, структура интерфейса и графический интерфейс, и делаются выводы по главе.

В третьей главе описывается процесс реализации, включая выбор инструментов и описание интерфейсов приложения, таких как страницы с интересами пользователя, выбора шаблона, создания своего блока, шаблоны сна, питания, спорта, страницы новостей, профиля и расписания, и делаются выводы по главе.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

В мире, где люди больше стремятся к организации времени и контролю за повседневной активностью, мобильные приложения для трекинга дня становятся популярными. Эти приложения позволяют не только фиксировать события и задачи, но и отслеживать прогресс в различных областях, таких как спорт, питание и сон. Данное приложение поможет пользователям систематизировать деятельность и достичь поставленных целей.

Трекер дня — это инструмент, который помогает пользователям отслеживать и управлять повседневными задачами и активностями.

Объектом исследования является процесс разработки мобильного приложения. Предметом исследования выступает клиентская часть мобильного приложения для трекинга дня.

Мобильное приложение отслеживает интересы каждого пользователя каждый день. Каждый пользователь может настроить приложение под свои интересы. Также пользователь получает полный анализ и рекомендации по интересам.

Целью данной работы является разработка клиентской части мобильного приложения для трекинга дня, которое предоставляет пользователям возможность создания и настройки блоков интересов. Основные задачи включают анализ предметной области, проектирование приложения и его программную реализацию.

Целевая аудитория:

Подростки (до 18 лет): к сожалению, у подростков слишком много комплексов. Например, лишний вес. От лишнего веса можно избавиться тренировками, которые они могут отслеживать в нашем приложении.

Молодёжь (до 35 лет): в это время люди ищут себя и пытаются попробовать от жизни всё, но дел слишком много. Эти вещи они могут заносить в наше приложение, чтобы не забыть, что они хотят попробовать и понять, что из этого списка им сейчас важнее.

Взрослые (после 35 лет): в данном возрасте люди могут просто отслеживать свой досуг, чтобы потом лучше понять в какой области они могут начать развиваться больше.

В первой главе проводится анализ предметной области, включающий обзор существующих решений, таких как популярные мобильные приложения "To-do List", "TickTime" и "Tusk", и их сравнительный анализ, а также определение функциональных требований к мобильному приложению.

Вторая глава посвящена проектированию приложения: здесь рассматриваются диаграммы прецедентов, структура базы данных, структура интерфейса и графический интерфейс, и делаются выводы по главе.

В третьей главе описывается процесс реализации, включая выбор инструментов и описание интерфейсов приложения, таких как страницы с интересами пользователя, выбора шаблона, создания своего блока, шаблоны сна, питания, спорта, страницы новостей, профиля и расписания, и делаются выводы по главе.

1 Анализ предметной области

1.1 Обзор существующих решений

На рынке существуют всевозможные мобильные приложения для трекинга дня, такие как "To-do List", "TickTime", и "Tusk". Каждое из них обладает оригинальными функциями и особенностями. Анализ этих приложений позволил выявить основные требования и функции, которые будут реализованы в разрабатываемом приложении.

Обзор аналогичных мобильных приложений

Мобильное приложение «To-do List».

"To-do List" представляет собой мобильное приложение, предоставляющее удобный инструмент для управления задачами и планирования времени. Пользователи могут создавать заметки, которые автоматически интегрируются в календарь, обеспечивая систематизацию и хронологию задач.

Основные функции "To-do List" включают:

– напоминания задач;

– виджеты для главного экрана;

– список задач;

– просмотр календаря;

– статистика выполненных задач.

"To-do List" является полезным инструментом для пользователей, стремящихся организовать свою повседневную жизнь, отслеживать задачи и быть в курсе предстоящих событий, объединяя функциональность заметок и календаря.

Мобильное приложение «TickTime».

"TickTime" представляет собой мобильное приложение, ориентированное на точное измерение времени и предоставление удобных инструментов для эффективного управления рабочим временем. Основные функции приложения включают:

– секундомер;

– таймер;

– статистика времени;

– календарь.

"TickTime" является полезным инструментом для тех, кто ценит точность измерения времени и стремится эффективно его использовать. Сочетание секундомера, таймера помодоро, статистики времени и календаря делает приложение многофункциональным и удовлетворяющим потребности пользователей в эффективном управлении временем.

Мобильное приложение «Tusk».

"Tusk" представляет собой многофункциональное мобильное приложение, ориентированное на эффективное управление задачами и повышение продуктивности пользователей. Основные функции приложения включают:

– многообразие выбора задач;

– большой выбор сортировки задач;

– календарь;

– статистика;

– достижения.

"Tusk" предоставляет многообразные и интуитивно понятные инструменты для эффективного управления задачами. Сочетание широкого выбора задач, всевозможных опций сортировки, интегрированного календаря, статистики и структуры достижений делает приложение привлекательным для тех, кто ищет комплексное решение для организации рабочего процесса.

Сравнительный анализ

Характеристика приложений представлена в таблице 1.

Таблица 1 – Характеристика приложений

Характеристика	Трекер дня	To-do List	TickTime	Tusk
Напоминания задач	Есть	Есть	Нет	Есть
Виджеты для главного экрана	Есть	Есть	Нет	Нет
Список задач	Есть	Есть	Нет	Есть
Просмотр календаря	Есть	Есть	Есть	Есть
Изменение настроек	Есть	Есть	Нет	Есть

1.2 Функциональные требования к мобильному приложению

В этом разделе представлены функциональные требования к мобильному приложению, которые определяют ключевые функции и возможности, необходимые для удовлетворения потребностей пользователей в управлении задачами и повышении продуктивности. Ниже приведены основные функциональные требования:

Напоминание задач

Описание: Приложение должно предоставлять возможность создавать напоминания для предстоящих задач.

Функционал:

– настройка уведомлений о задачах на определенное время;

– возможность выбора звуковых сигналов и вибрации для уведомлений;

– повторяющиеся напоминания для регулярных задач.

Виджеты для главного экрана

Описание: Пользователи должны иметь возможность добавлять виджеты на главный экран устройства для быстрого доступа к текущим задачам и событиям.

Функционал:

– виджеты с отображением списка текущих задач;

– виджеты с визуализацией ближайших событий;

– настраиваемый размер и внешний вид виджетов.

Список задач

Описание: Приложение должно предоставлять возможность создания и управления списком задач.

Функционал:

– добавление, редактирование и удаление задач;

– возможность группировки задач по категориям;

– настройка приоритетов задач.

Просмотр календаря

Описание: Приложение должно интегрироваться с календарем для отображения задач по дням.

Функционал:

– встроенный календарь для просмотра задач;

– синхронизация с внешними календарями (Google Calendar, Apple Calendar);

– фильтрация задач по дате и времени.

Изменение настроек

Описание: Приложение должно предоставлять пользователю возможность настраивать различные параметры для персонализации и удобства использования.

Функционал:

– изменение личной информации пользователя;

– включение/отключение уведомлений, настройка частоты и типа уведомлений;

– выбор темы оформления, изменение шрифтов и цветовых схем.

1.3 Выводы по главе

Анализ предметной области и существующих мобильных приложений для управления задачами, таких как "To-do List", "TickTime" и "Tusk", выявил

ключевые функции, необходимые для нашего приложения: напоминания задач, виджеты, интеграция с календарем, статистика выполненных задач и времени, а также система достижений. Эти требования обеспечат конкурентоспособность и удовлетворение потребностей пользователей, создавая основу для эффективного проектирования и реализации продукта.

2 Проектирование приложения

2.1 Диаграмма прецедентов

Диаграмма прецедентов описывает взаимодействие пользователя с приложением и основные функциональные возможности. Диаграмма представлена на рисунке 1.

При использовании приложения пользователь может:

– посмотреть расписание, где указанно расписания пользователя на каждый день, которое он может изменять, добавлять и удалять;

– посмотреть свой профиль, в котором указана информация о самом пользователи;

– изменить настройки самого приложения;

– посмотреть боки со своими интересами, также изменить или удалить существующие блоки и создать новый блок, используя шаблоны или создав свой собственный шаблон.

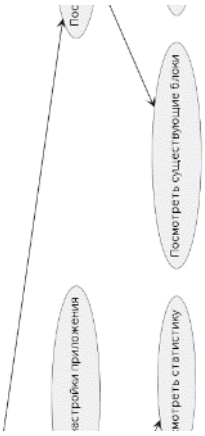

Рисунок 1 – Диаграмма

2.2 Взаимодействие с базой данных

Атрибуты сущностей

Сущность представляет собой тип объектов, которые хранятся в базе данных, и каждая таблица соответствует одной сущности. Обычно сущности

отражают объекты из реальной жизни. У сущности имеется набор атрибутов, которые описывают её характеристики. В таблице каждый столбец соответствует одному атрибуту, а каждая строка – отдельному экземпляру сущности.

Атрибуты сущностей представлены в таблице 2.

Таблица 2 – Атрибуты сущностей

Атрибут	Назначение	Тип	Ограничения
Users			
ID	Идентификатор	Bigint	Primary key
name	Имя	Text	Not null
email	Логин	Text	Not null
password	Пароль	Text	Not null
settings			
ID	Идентификатор	Bigint	Primary key
user_id	Идентификатор пользователя	Bigint	Foreign key
theme	Тема интерфейса	Text	Not null
notifications	Уведомления	Text	Not null
profile			
ID	Идентификатор	Bigint	Primary key
user_id	Идентификатор пользователя	Bigint	Foreign key
bio	Биография пользователя	Text	Not null
avatar_url	Аватар пользователя	Text	Not null
interest_blocks			
ID	Идентификатор	Bigint	Primary key
user_id	Идентификатор пользователя	Bigint	Foreign key
title	Название	Text	Not null
daily_schedule			
ID	Идентификатор	Bigint	Primary key
user_id	Идентификатор пользователя	Bigint	Foreign key
date	Дата	Timestamp without time zone	Not null
task	Описание	Text	Not null

Окончание таблицы 2

Атрибут	Назначение	Тип	Ограничения
notes			
ID	Идентификатор	Bigint	Primary key
user_id	Идентификатор пользователя	Bigint	Foreign key
interest_block_id	Идентификатор блока с интересами	Bigint	Not null

content	Заметки	Text	Not null

Схема данных представлена на рисунке 2.

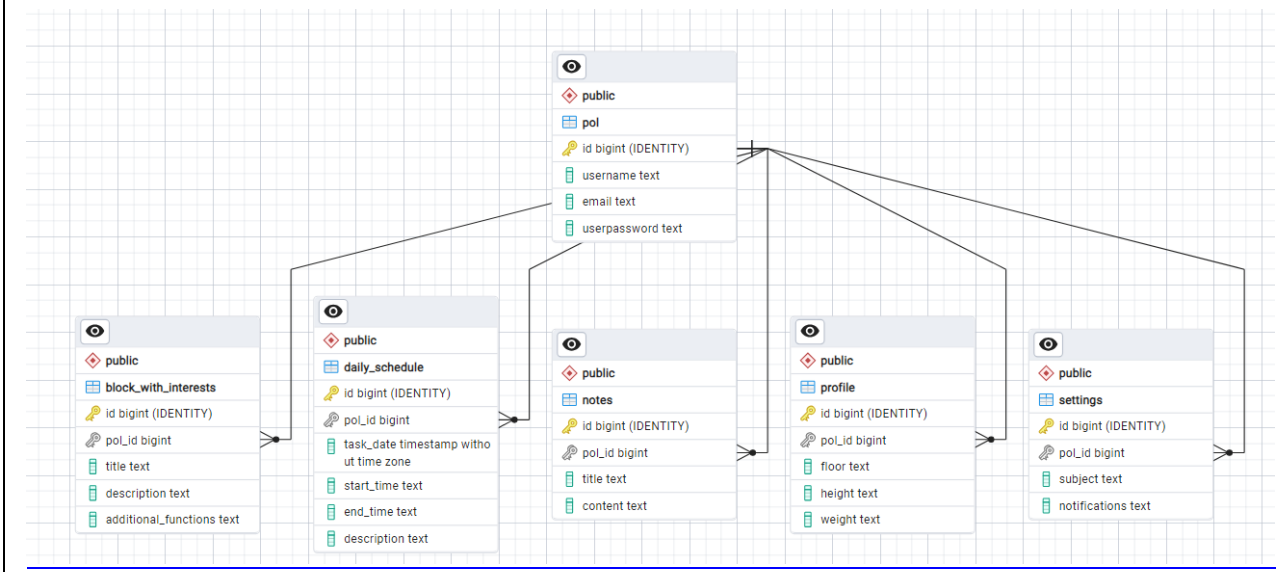

Рисунок 2 – Схема данных

Связующая таблица содержит столбцы-ссылки на первичные ключи и позволяет связать между собой несколько записей из одной таблицы с несколькими записями из другой таблицы.

Использование связующей таблицы позволяет эффективно управлять данными об интересах и обеспечивает гибкость при построении запросов и анализе данных. Это удобно для визуализации своих блоков, которые содержат определенные интересы пользователя.

2.3 Структура интерфейса

Структура приложения включает боковую панель с разделами: настройки, профиль, расписание, новости и выход. Главная страница содержит блоки с интересами, которые можно создавать, редактировать и удалять. Структура приложения представлена на рисунке 3.

Рисунок 3 – Структура разрабатываемого приложения

2.4 Графический интерфейс

Графический интерфейс приложения должен быть интуитивно понятным и удобным для пользователя. Для этого используются стандартные элементы управления и современные подходы к дизайну мобильных приложений. Графический интерфейс приложения представлен на рисунке 4.

Элементы графического интерфейса:

– 1-блоки с интересами пользователя;

– 2-профиль пользователя;

– 3-настройки;

– 4-расписание пользователя;

– 5-новости.

Рисунок 4 – График интерфейса мобильного приложения

2.5 Взаимодействие клиент–сервер

Практически все современное ПО построено на принципе взаимодействия Клиент-Сервер – это способ обмена информацией между двумя компьютерами, где одна сторона (клиент) запрашивает данные у серверной части. Последняя формирует ответ, направляя его в обмен [9].

Одно из главных – это возможность распределения функций между сторонами в связке. Клиент может выполнять легкие задачи, такие как отображение информации на экране, в то время как сервер может обрабатывать более сложные задачи, к примеру, вычисления и хранение данных. Другими же плюсами являются:

– централизованное управление данными и услугами, что упрощает их обслуживание, обновление и безопасность;

– клиенты могут пользоваться любыми системами и устройствами, что делает архитектуру универсальной и доступной для широкого круга пользователей;

– клиенты вправе обратиться к серверу через сеть, что обеспечивает географическую гибкость и позволяет пользователям работать удаленно;

– эффективное управление помогает оптимизировать нагрузку и гарантирует быстрый отклик приложений;

– сервер может предоставлять общий доступ к данным, что облегчает совместное использование информации между пользователями;

– архитектура активно контролирует безопасность и доступность информации, так как централизованный сервер может реализовывать строгие политики шифрования.

Главным же недатском данного решения считается возможность сбоев в работе сервера или сети.

Механизм взаимодействия сторон выглядит следующим образом:

а) клиент при соединении отсылает запрос: например, через сеть Интернет;

б) сервер его принимает и производит анализ, пытаясь понять, что нужно отправителю;

в) сервер формирует ответ или извлекает информацию из базы данных;

г) итоговые данные отправляются обратно клиенту;

д) клиент принимает информацию и выводит ее на экран пользователя, может продолжить взаимодействие.

Этот подход позволяет эффективно организовать обмен информацией и ресурсами в сети. Диаграмма последовательности взаимодействия клиента и сервера представлена на рисунке 5.

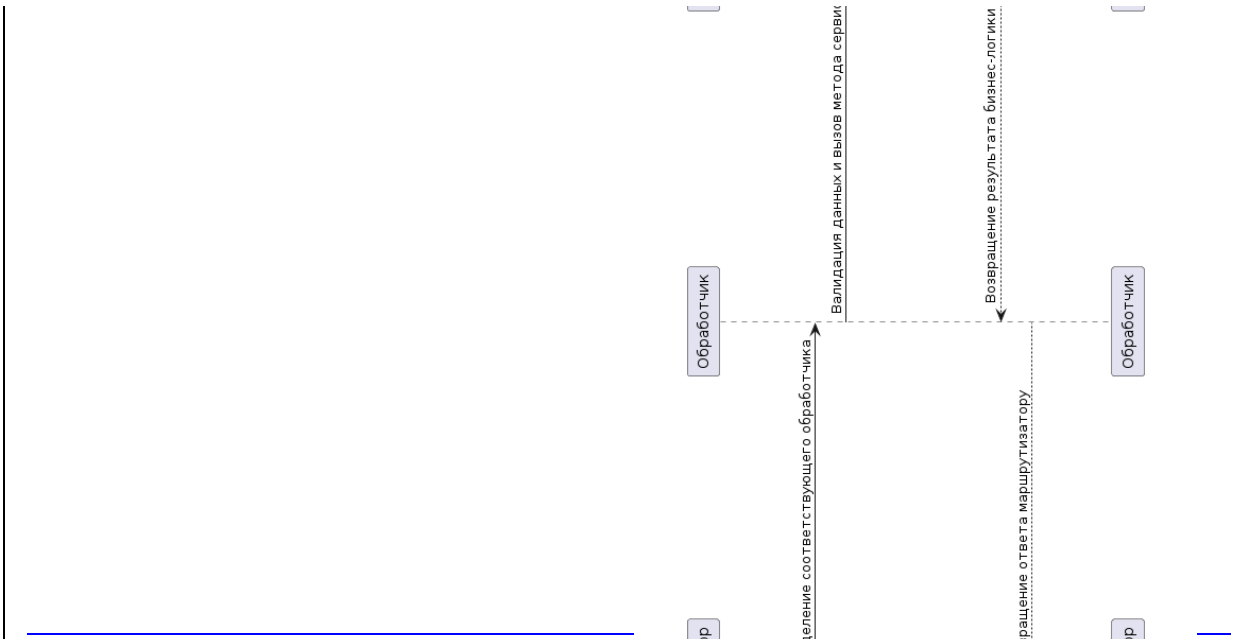

Рисунок 5 – Диаграмма

2.6 Выводы по главе

В данной главе рассмотрены основные аспекты проектирования приложения. Диаграмма прецедентов описала ключевые сценарии взаимодействия пользователя с приложением. Описаны атрибуты сущностей и взаимодействие с базой данных, включая использование связующих таблиц для управления информацией. Представлена структура интерфейса с боковой панелью и основными разделами, обеспечивающая интуитивную навигацию. Графический интерфейс разработан с учетом современных стандартов мобильного дизайна для удобства пользователей. Также рассмотрено взаимодействие клиент-сервер, что обеспечивает надежное распределение задач и эффективный обмен информацией.

3 Реализация

3.1 Выбор инструментов

Для разработки приложения были выбраны следующие инструменты и технологии:

– Visual Studio Code для написания кода;

– Android Studio для тестирования и отладки на платформе Android;

– языки программирования: HTML, CSS, JavaScript;

– компонент: Android WebView.

3.2 Ключевые фрагменты кода и их описание

К ключевым функциям и компонентам относятся:

– WebView;

– loadPage;

– setupEventListeners;

– loadBlockContent.

Инициализация компонента WebView представлена на рисунке 6 и рисунке 7.

```
<androidx.constraintlayout.widget.ConstraintLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity">

    <WebView
        android:id="@+id/webView"
        android:layout_width="match_parent"
        android:layout_height="match_parent"></WebView>

</androidx.constraintlayout.widget.ConstraintLayout>
```

Рисунок 6 – Определение структуры WebView в макете

Этот фрагмент XML-кода определяет макет активности, в котором используется ConstraintLayout для размещения WebView. WebView занимает все доступное пространство экрана, что позволяет отображать веб-контент на всю ширину и высоту устройства.

```
<application
    android:allowBackup="true"
    android:dataExtractionRules="@xml/data_extraction_rules"
    android:fullBackupContent="@xml/backup_rules"
    android:icon="@mipmap/ic_launcher"
    android:label="@string/app_name"
    android:roundIcon="@mipmap/ic_launcher_round"
    android:supportsRtl="true"
    android:theme="@style/Theme.MyApplication"
    tools:targetApi="31">
    <activity
        android:name=".MainActivity"
        android:exported="true">
        <intent-filter>
            <action android:name="android.intent.action.MAIN" />

            <category android:name="android.intent.category.LAUNCHER" />
        </intent-filter>

        <meta-data
            android:name="android.app.lib_name"
            android:value="" />
    </activity>
</application>
```

Рисунок 7 – Настройки для разрешений и метаданных

Этот фрагмент кода из AndroidManifest.xml включает разрешение на доступ в интернет, что необходимо для загрузки веб-страниц в WebView. Также определяет основную активность приложения и метаданные, такие как иконки и тема приложения.

Работа с контентом и пользовательским интерфейсом представлена на рисунках 8-12.

```
function loadPage(pageName) {
    fetch(pageName) // Загружаем содержимое страницы с помощью Fetch API
        .then(response => response.text())
        .then(data => {
            document.querySelector('.content').innerHTML = data;
            // Вставляем содержимое страницы в блок .content
            setupEventListeners();
            // Устанавливаем обработчики событий для кнопок в боковой панели
        })
        .catch(error => console.error('Ошибка загрузки страницы:', error));
}

function setupEventListeners() {
document.getElementById("block").addEventListener("click", function() {
    loadBlockContent();
});
document.getElementById("add").addEventListener("click", function() {
    loadAddContent();
});
document.getElementById("nast").addEventListener("click", function() {
    loadNastContent();
});
document.getElementById("kalend").addEventListener("click", function() {
    loadKalendContent();
});
document.getElementById("news").addEventListener("click", function() {
    loadNewsContent();
});
document.getElementById("exit").addEventListener("click", function() {
    loadExitContent();
});
```

Рисунок 8 – Работа боковой панели

Этот фрагмент кода реализует загрузку страниц и обработку событий боковой панели. Функция loadPage загружает содержимое указанной страницы с помощью Fetch API и вставляет его в элемент с классом ‘.content’. Функция setupEventListeners добавляет обработчики событий на элементы боковой панели, позволяя пользователям загружать соответствующие страницы при клике.

```
function loadBlockContent() {
    fetch('block.html')
        .then(response => response.text())
        .then(data => {
            document.querySelector('.content').innerHTML = data;
            // Вставляем содержимое block.html в блок .content
            setupEventListeners(); // Переустанавливаем обработчики событий
        })
        .catch(error => console.error('Ошибка загрузки файла block.html:', error));
}
// Вызываем функцию при загрузке страницы для отображения содержимого block.html
window.onload = loadBlockContent;
```

Рисунок 9 – Подключение страниц

Функция loadBlockContent загружает содержимое файла block.html и вставляет его в элемент с классом ‘.content’. Обработчики событий переустанавливаются после загрузки контента. Эта функция вызывается при загрузке страницы, чтобы отобразить содержимое block.html по умолчанию. Также подключаем другие страницы.

```
<div class="sleep-settings">
    <div class="setting-block">
        <h2>Вы спали:</h2>
        <div class="sub-block all-time">
            <label for="totalTime">Общее время:</label>
            <input type="time" id="totalTime" class="time-input">
        </div>
        <div class="sub-block time-range-container">
            <div class="time-range">
                <label for="startSleep">Начало сна:</label>
                <input type="time" id="startSleep" class="time-input">
            </div>
            <div class="time-range">
                <label for="endSleep">Конец сна:</label>
                <input type="time" id="endSleep" class="time-input">
            </div>
        </div>
    </div>
    <div class="setting-block">
        <h2>Вы просыпались ночью:</h2>
        <div class="sub-block time-range-container">
            <div class="time-range">
                <label for="nightWakeUp">Проснулись:</label>
                <input type="time" id="nightWakeUp" class="time-input">
            </div>
            <div class="time-range">
                <label for="nightSleepAgain">Уснули:</label>
                <input type="time" id="nightSleepAgain" class="time-input">
            </div>
        </div>
    </div>
```

Рисунок 10 – Часть шаблона сна

Этот HTML-шаблон представляет собой страницу "Настройки сна". Он включает элементы для ввода времени сна, времени пробуждения, качества сна и заметок о снах. Структура страницы помогает пользователям легко вводить и отслеживать данные о своём сне. Также на подобии данного шаблона были созданы ещё два шаблона.

```
<div class="content">
    <h2>Выберите готовый шаблон:</h2>
    <!-- Содержимое вашей страницы списка -->
    <div class="button-list">
        <button onclick="selectTemplate('sport')">Спорт</button>
        <!--<button onclick="selectTemplate('health')">Здоровье</button>-->
        <button onclick="selectTemplate('sleep')">Сон</button>
        <button onclick="selectTemplate('food')">Питание</button>
    </div>
</div>
```

Рисунок 11 – Список шаблонов

Этот фрагмент HTML-кода представляет страницу выбора шаблонов. Пользователь может выбрать готовый шаблон, такой как "Спорт", "Сон" или "Питание", для дальнейшей работы. Кнопки вызывают функцию selectTemplate, которая загружает соответствующий контент.

```
<div class="content">
    <h2>Опросник</h2>
    <form id="surveyForm">
        <div class="question">
            <label for="blockName">Как будет называться ваш блок?</label><br>
            <input type="text" id="blockName" name="blockName">
        </div>
        <div class="question">
            <label for="criteria">Какие критерии вы хотите отмечать?</label><br>
            <textarea id="criteria" name="criteria"></textarea>
        </div>
        <div class="question">
            <label for="showChart">Нужна ли диаграмма на месяц?</label><br>
            <input type="radio" id="showChartYes" name="showChart" value="yes">
            <label for="showChartYes">Да</label>
            <input type="radio" id="showChartNo" name="showChart" value="no">
            <label for="showChartNo">Нет</label>
        </div>
        <div class="question">
            <label for="marathon">Нужен ли марафон?</label><br>
            <input type="radio" id="marathonYes" name="marathon" value="yes">
            <label for="marathonYes">Да</label>
            <input type="radio" id="marathonNo" name="marathon" value="no">
            <label for="marathonNo">Нет</label>
        </div>
        <div class="question">
            <label for="rating">Нужна ли оценка от 1 до 5?</label><br>
            <input type="radio" id="ratingYes" name="rating" value="yes">
            <label for="ratingYes">Да</label>
            <input type="radio" id="ratingNo" name="rating" value="no">
            <label for="ratingNo">Нет</label>
        </div>
```

Рисунок 12 – Часть опросника для создания нового шаблона

Этот HTML-фрагмент представляет опросник для создания нового шаблона. Пользователь может указать название блока, критерии, необходимость диаграммы, марафона и оценки, а также выбрать иконки и фон. Форма позволяет настроить новый шаблон по индивидуальным требованиям пользователя.

3.3 Интерфейс

Android Studio предоставляет мощные инструменты для создания пользовательского интерфейса (UI) Android-приложений. Одним из ключевых компонентов для разработки интерфейса в Android Studio является Android Jetpack. Jetpack - это набор библиотек, инструментов и руководств, которые позволяют разработчикам легко создавать высококачественные приложения.

Интерфейс страницы с интересами пользователя представлен на рисунке 13.

Рисунок 13 – Страница с интересами

На данной странице у пользователя есть такие возможности как:

– просмотр своих блоков интересно(после их добавления);

– добавление нового блока с интересами через страницу выбора шаблона;

– добавление нового блока с интересами через страницу создания нового шаблона.

Интерфейс страницы выбора шаблона представлен на рисунке 14.

Рисунок 14 – Выбор шаблона

На этой странице находится список шаблонов. После перехода на данную страницу пользователь может:

– выбрать один из шаблонов и создать готовый блок с уже существующим интересом, например «Сон».

Интерфейс страницы создания своего блока представлен на рисунке 15.

Рисунок 15 – Создание блока

На этой странице находится опросник. После перехода на данную страницу пользователь может:

– ответить на все вопросы и создать свой собственный шаблон(блок) со своим интересам, которого нет в списке шаблонов.

Интерфейс шаблона «Сон» представлен на рисунке 16.

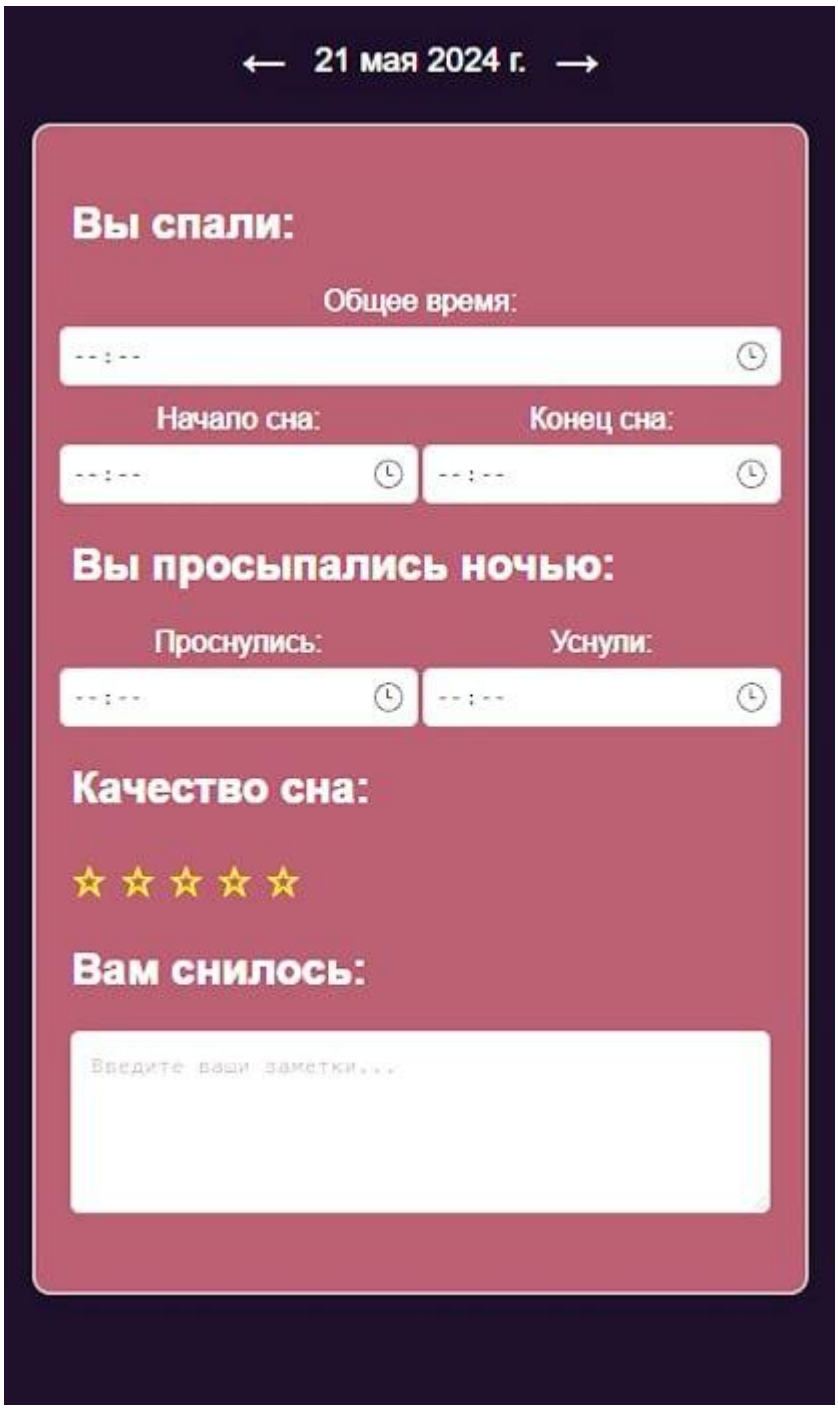

Рисунок 16 – Шаблон «Сон»

На данной странице у пользователя есть такие возможности как:

– отслеживать начало и конец своего сна, а также общее время сна;

– отслеживать своё пробуждение ночью, если оно было;

– отмечать качество своего сна;

– записывать свои сны, если они были.

Интерфейс шаблона «Питание» представлен на рисунке 17.

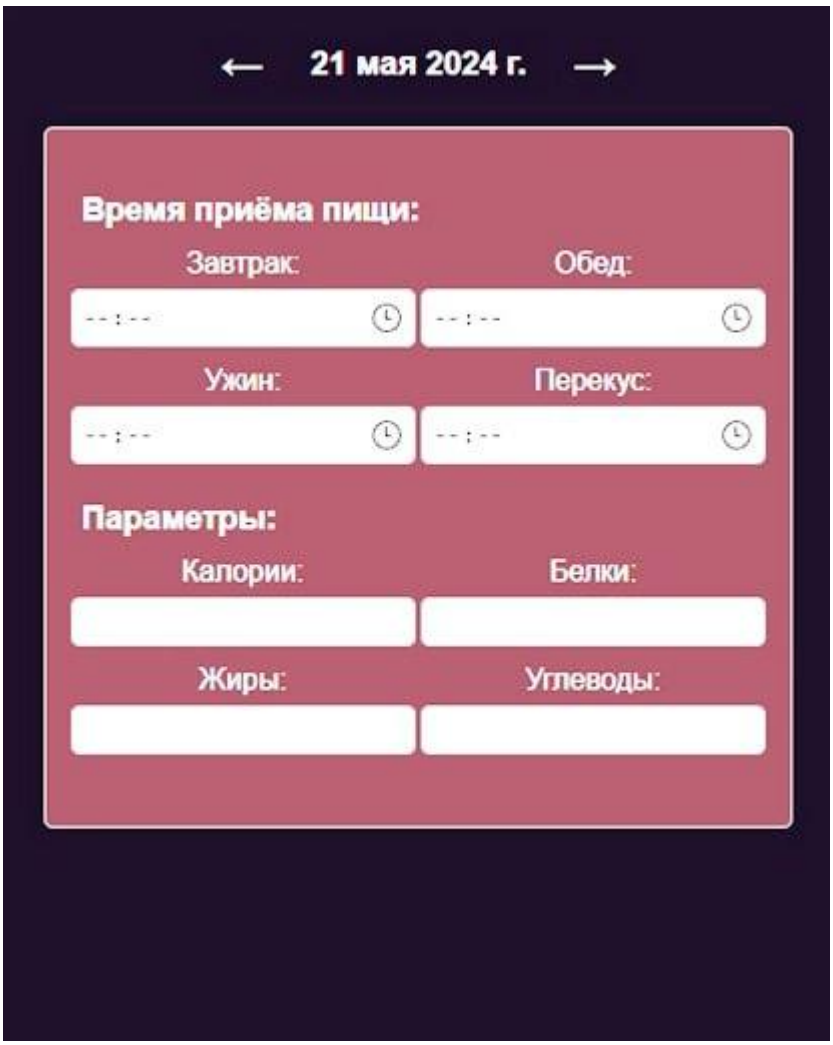

Рисунок 17 – Шаблон «Питание»

На данной странице у пользователя есть такие возможности как:

– отслеживать время приёма пищи;

– отслеживать параметры своей пищи за весь день.

Интерфейс шаблона «Спорт» представлен на рисунке 18.

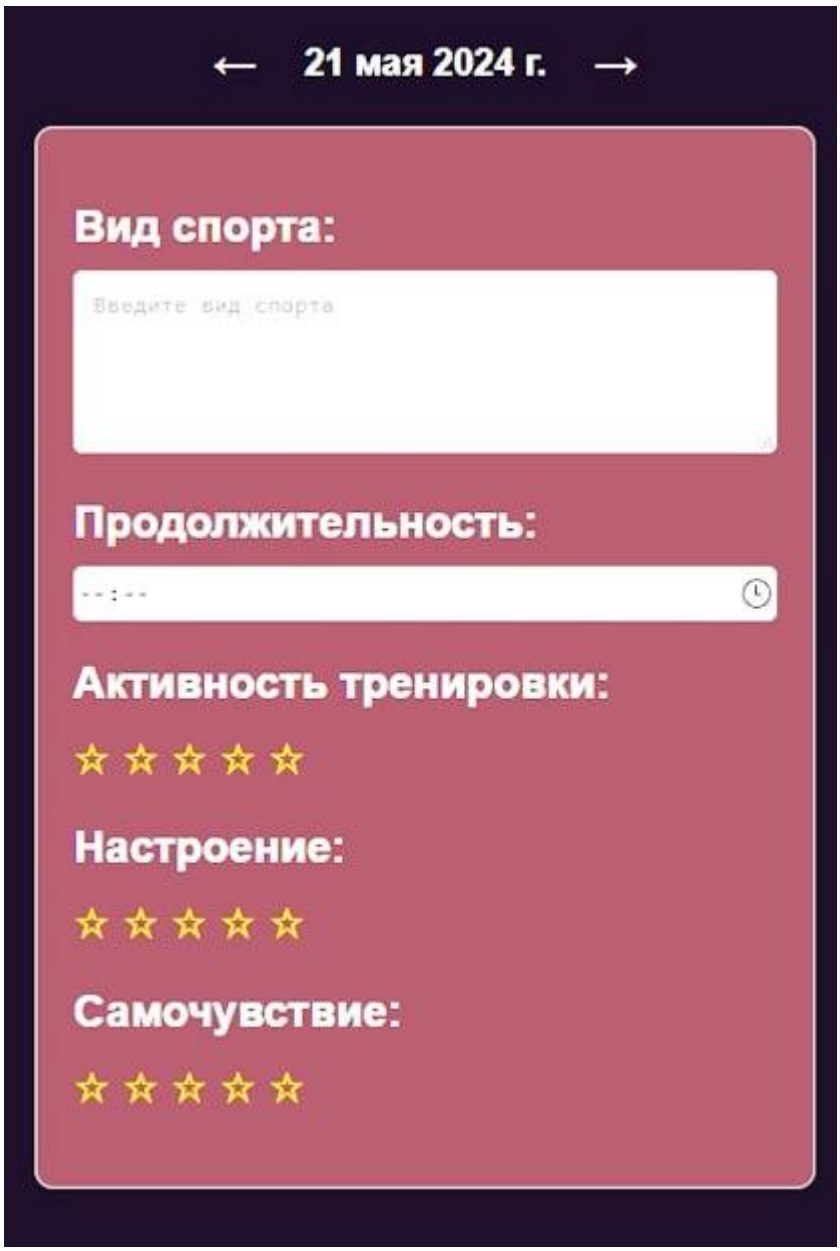

Рисунок 18 – Шаблон «Спорт»

На данной странице у пользователя есть такие возможности как:

– отмечать каким видом спорта пользователь занимался сегодня;

– отслеживать продолжительность своей тренировки;

– отмечать активность тренировки;

– отмечать своё настроение и самочувствие сегодня.

Интерфейс страницы новостей представлен на рисунке 19.

Рисунок 19 – Новости

На данной странице у пользователя есть такая возможность как:

– просмотр полезных статей из интернета о своих интересах.

Интерфейс страницы профиля пользователя представлен на рисунке 20.

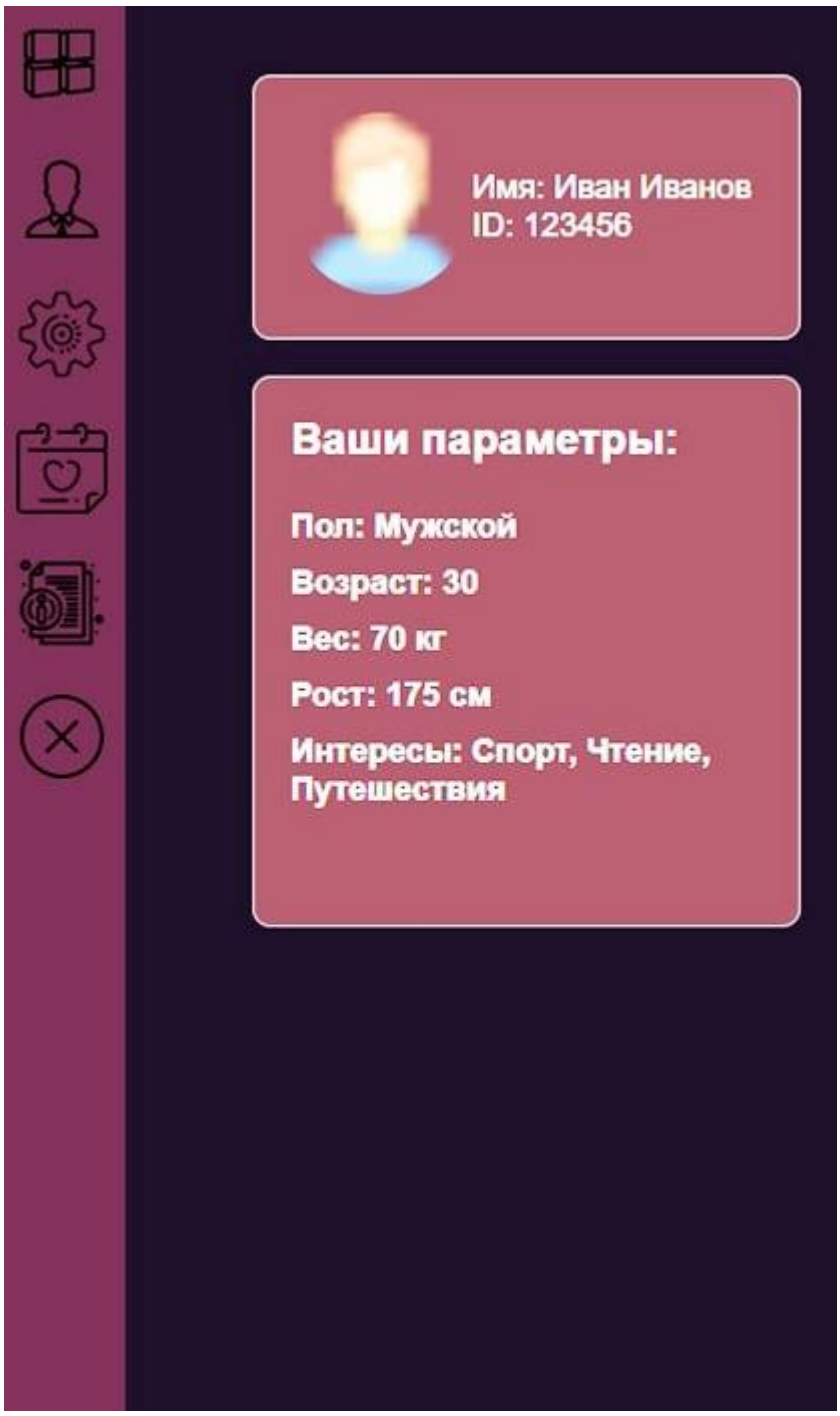

Рисунок 20 – Профиль пользователя

На данной странице у пользователя есть такие возможности как:

– просмотр своих личных параметров.

Интерфейс страницы расписания представлен на рисунке 21.

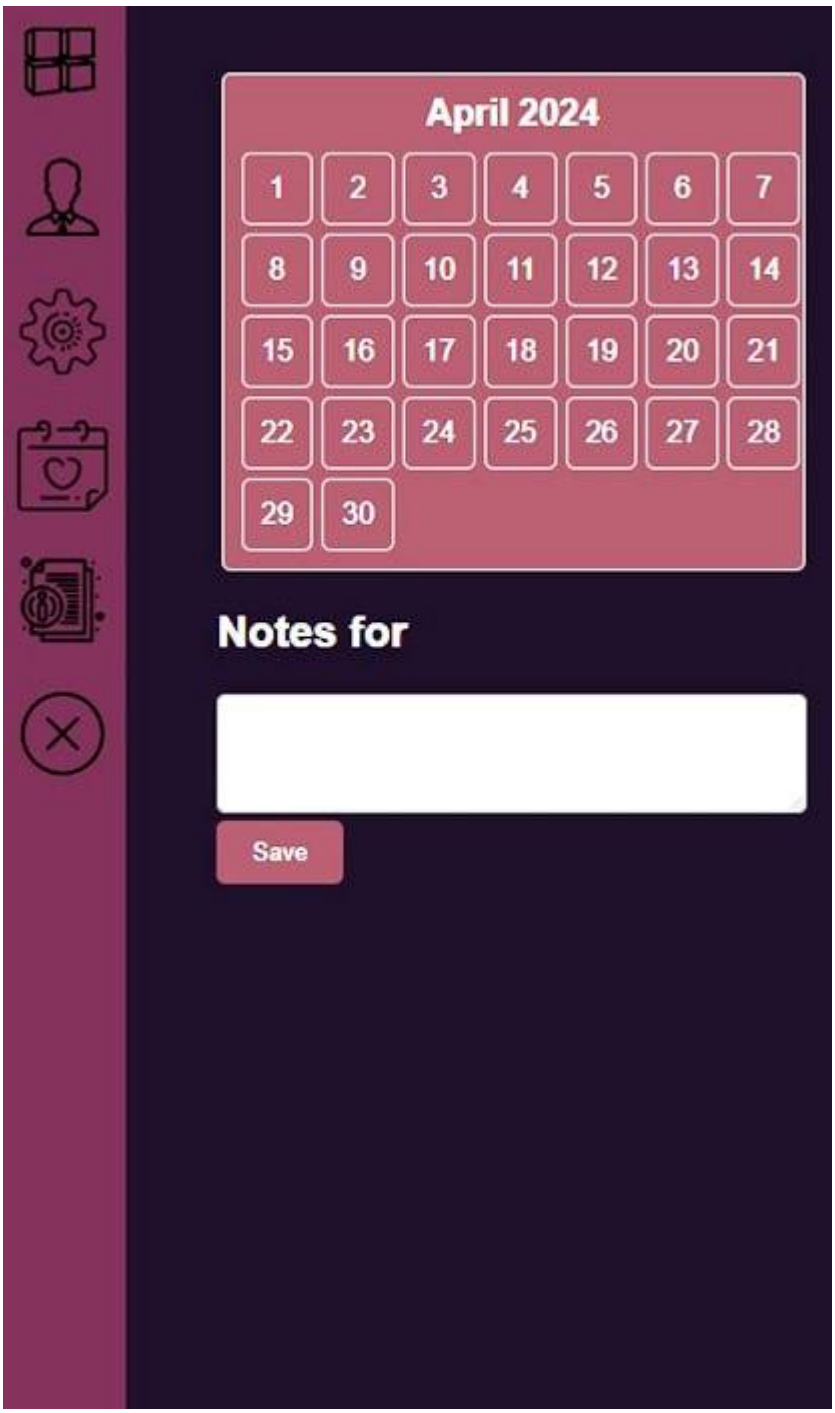

Рисунок 21 – Расписание

На данной странице у пользователя есть такая возможность как:

– запись дел на каждый день.

3.4 Выводы по главе

В данной главе описана реализация приложения, включая выбор инструментов (Visual Studio Code, Android Studio, HTML, CSS, JavaScript, Android WebView). Рассмотрены ключевые фрагменты кода, такие как инициализация WebView и функции loadPage, setupEventListeners, loadBlockContent, обеспечивающие загрузку и отображение контента. Также отмечено использование Android Jetpack для создания удобного интерфейса. Эти процессы и инструменты сформировали основу для успешной реализации приложения.

ЗАКЛЮЧЕНИЕ

В дипломной работе была разработана всесторонняя мобильная программа для управления задачами.

В первой главе проанализированы существующие решения, такие как "To-do List", "TickTime" и "Tusk", выявив ключевые функции: напоминания, виджеты и интеграция с календарем. Эти функции необходимы для удовлетворения потребностей пользователей и обеспечения конкурентоспособности.

Во второй главе рассмотрены аспекты проектирования. Диаграмма прецедентов показала ключевые сценарии взаимодействия. Описаны атрибуты сущностей и работа с базой данных, включая связующие таблицы. Представлена структура интерфейса с боковой панелью, обеспечивающая интуитивную навигацию, и разработан графический интерфейс по современным стандартам. Рассмотрено взаимодействие клиент-сервер для надежного распределения задач и обмена информацией.

В третьей главе описана реализация. Выбраны инструменты: Visual Studio Code, Android Studio, HTML, CSS, JavaScript и Android WebView. Рассмотрены ключевые фрагменты кода, обеспечивающие загрузку и отображение контента. Использован Android Jetpack для создания удобного интерфейса. Эти процессы и инструменты сформировали основу успешной реализации.

В результате была создана полноценная, конкурентоспособная мобильная программа, сочетающая удобство, богатый функционал и современный дизайн.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Visual Studio Code: официальный сайт – URL: https://code.visualstudio.com (дата обращения 13.03.2024).
2. Чемпионат: русскоязычный интернет ресурс о правилах тренировки – URL: https://www.championat.com (дата обращения 27.05.2024).
3. Центр общественного здоровья и медицинской профилактики: русскоязычный интернет ресурс с советами для ЗОЖ – URL: https://profilaktica.ru (дата обращения 27.05.2024).
4. Медицина обо мне: русскоязычный интернет ресурс с советами, которые помогут выспаться – URL: https://medaboutme.ru (дата обращения 27.05.2024).
5. Городская клиническая больница: русскоязычный интернет ресурс о принципах здорового питания – URL: https://6gkb.by (дата обращения 27.05.2024).
6. Виды клиент-серверной архитектуры: русскоязычный интернет ресурс об основных особенностях и видов архитектур клиент-сервер – URL: https://www.ittelo.ru/news/osnovnye-osobennosti-i-vidy-arkhitektur-klient-server/#tag-9 (дата обращения 12.05.2024).
7. Основы проектирования баз данных: русскоязычный интернет ресурс о программировании – URL: https://metanit.com/sql/tutorial/1.1.php (дата обращения 07.05.2024).
8. PlantUML: интернет ресурс с возможностью создания UML диаграмм – URL: https://plantuml.com (дата обращения 17.04.2024).
9. СТУ 7.5-07-2021. Стандарт университета. Система менеджмента качества. Общие требования к построению, изложению и оформлению документов учебной деятельности // Сибирский федеральный университет : официальный сайт. – URL: https://about.sfu-kras.ru/node/8127 (дата обращения: 07.06.2024).

Министерство науки и высшего образования РФ
Федеральное государственное автономное
образовательное учреждение высшего образования
«СИБИРСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Институт космических и информационных технологий

Кафедра вычислительной техники

УТВЕРЖДАЮ
Заведующий кафедрой
_____ О.В. Непомнящий
«20» 06 2024 г.

БАКАЛАВРСКАЯ РАБОТА

090301 Информатика и вычислительная техника

Мобильное приложение – трекер дня. Клиентская часть.

Руководитель	_____	20.06.24	доцент, канд. техн. наук	С.Н. Титовский
	подпись	*дата*	*должность, ученая степень*	
Выпускник	Бахт	20.06.24		И.С. Бахтеева
	подпись	*дата*		
Нормоконтролёр	_____	20.06.24	доцент, канд. техн. наук	С.Н. Титовский
	подпись	*дата*	*должность, ученая степень*	

Красноярск 2024