МИНОБРНАУКИ РОССИИ
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ
УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«ВОРОНЕЖСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»
(ФГБОУ ВО «ВГУ»)

Факультет Компьютерных наук

Кафедра программирования и информационных технологий

Мобильное приложение для для контроля времени и задач TimeAhead»

Курсовой проект

09.03.04 Программная инженерия

Профиль «Информационные системы и сетевые технологии»

Зав. кафедрой _____ С.Д. Махортов, д.ф.-м.н.,профессор __.__.2024

Обучающийся _____ Г.А.Асатрян

Обучающийся _____ М.В.Королицкий

Обучающийся _____ К.А.Чуркин

Руководитель _____ В.С. Тарасов, ст. преподаватель

Воронеж 2024

Содержание

Введение

Часто возникают ситуации, когда в голове имеется слишком много задач: нужно съездить в различные места, купить необходимые вещи или продукты, оплатить покупки и т.д. Но в течение дня часть из них забывается и, соответственно, не выполняется. Во избежании таких ситуаций следует добавить в жизнь важное понятие - time management.

Управление временем – это особый подход, при котором человек четко планирует свой день. Это позволяет повысить продуктивность и выполнить больше задач за меньший срок. Также можно научиться эффективно расходовать время, принимать быстрые и правильные решения и добиваться целей с меньшими усилиями.

Для планирования можно приобрести ежедневник и стикеры. В ежедневнике отмечать уже имеющиеся дела и повторяющиеся задачи, а на стикерах писать важные напоминания и приклеивать их на монитор или любое другое видное место.

Но со временем, когда задач и важных напоминаний становится больше, этот способ может стать неудобным. Технический прогресс не стоит на месте, и сейчас существует множество приложений, призванных справиться с хаосом в задачах и облегчить планирование.

1 Постановка задачи

1.1 Цели создания приложения

К целям создания приложения «TimeAhead» относятся:

— создание удобного и интуитивно понятного приложения для эффективного управления временем и задачами;

— предоставление подробной информации о том, как пользователь использует свое время, чтобы помочь ему в анализе и улучшении своих рабочих процессов и эффективности;

— расширение клиентской базы для привлечения рекламодателей;

— получение прибыли путем интеграции рекламы.

1.2 Задачи приложения

Разрабатываемый проект должен решать следующие задачи:

— ведение и просмотр записей о использовании времени. Пользователь может создавать записи о деятельности, указывать время начала и конца, категорию и описание;

— планирование задач. Пользователь может создавать задачи,указывая сроки, и отмечать их как выполненные;

— отслеживание привычек. Пользователь может создавать новые привычки, указывая периодичность;

— анализ использования времени. Пользователь может просмотреть статистику использования своего времени.

1.3 Требования к разрабатываемой системе

1.3.1 Функциональные требования

Авторизированный пользователь обладает следующими возможностями:

— добавление, редактирование и удаление активности (время начала и конца, название, описание, категория);

— просмотр созданных активностей по дням;

— добавление, редактирование и удаление задач (срок выполнения, напоминание, название, описание, категория);

— просмотр созданных задач с возможностью сортировки по времени;

— добавление с указанием дней и времени напоминания, названия и описания, редактирование и удаление записи о привычке;

— просмотр всех созданных привычек и просмотр по дням;

— создание своих категорий;

— просмотр статистики показывает уровень сформированности привычки.

Неавторизированный пользователь обладает следующими возможностями:

— добавление, редактирование и удаление активности (время начала и конца, название, описание, категория);

— регистрация или авторизация в аккаунт.

Администратор обладает следующими возможностями:

— создание общих категорий.

2 Анализ предметной области

2.1 Терминология (глоссарий) предметной области

Front-end – клиентская часть приложения. Отвечает за получение информации с программно-аппаратной части и отображение ее на устройстве пользователя;

Back-end – программно-аппаратная часть приложения. Отвечает за функционирование внутренней части приложения;

REST API – это архитектурный стиль для создания веб–сервисов. Он позволит приложениям взаимодействовать друг с другом и обмениваться данными через интернет;

Серверная часть – компьютер, обслуживающий другие компьютеры (клиентов) и предоставляющий им свои ресурсы для выполнения определенных задач;

Клиентская часть – компьютер, использующий ресурсы сервера и предоставляющий пользователю возможность взаимодействия с системой;

Фреймворк – готовый набор инструментов, который помогает разработчику быстро создать продукт;

Библиотека – это набор готовых функций, классов и объектов для решения каких-то задач;

СУБД (Система управления базами данных) – это совокупность программных и лингвистических средств общего или специального назначения, обеспечивающих управление созданием и использованием баз данных;

Объектно-реляционная СУБД – реляционная система управления базами данных, использующая в своей работе объектно-ориентированный подход: объекты, классы и наследование;

HTTP-протокол (HyperText Transfer Protocol) – сетевой протокол прикладного уровня.

2.2 Актуальность

Как показал опрос, проведенный аналитиками DISCOVERY Research Group, несмотря на знания различных методик планирования, 74% респондентов считают себя прокрастинаторами. А 57% опрошенных отметили, что им никогда не удаётся завершить все задачи, поставленные на день, что ожидаемо, так как 41% участников исследования сказали, что обычно ставят от 8 задач в день, тогда как статистика использования различных приложений показывает, что в среднем пользователи закрывают в день 6–8 задач.

Участники опроса называли разные причины своих неудач в управлении временем. Самая распространённая — неумение планировать свои задачи и контролировать их выполнение. В этом признались 20% респондентов.

Таким образом, в ситуации, когда необходимо держать в голове цели, спланировать порядок действий и отслеживать шаги, нужны помощники. Сегодня это электронные планировщики со множеством встроенных функций для тайм-менеджмента и управления целыми проектами.

Можно составлять планы, корректировать их одним кликом, отслеживать прогресс выполнения, передавать задачи другим исполнителям. Узнавать о важных событиях или назначенных встречах из уведомлений, отправленных на мессенджер, электронную почту или в виде push-сообщений на главный экран цифрового устройства. Благодаря им можно правильно распределять фокус и ничего не пропустить.

2.3 Обзор аналогов

Этап обзора аналогов является важной частью процесса разработки мобильного приложения. Этот шаг включает в себя изучение приложений,

аналогичных разрабатываемому, чтобы лучше понять ожидания пользователей, отраслевые тенденции и передовые методы разработки мобильных приложений. Этот обзор позволит собрать информацию и идеи, полезные для разработки высококачественного приложения, которое будет соответствовать потребностям целевой аудитории и выделяться на рынке.

2.3.1 LifeViewer

Есть задачи, итоги дня, идеи и мысли (заметки), фотографии, события и трекер привычек. И все это – на главном экране.

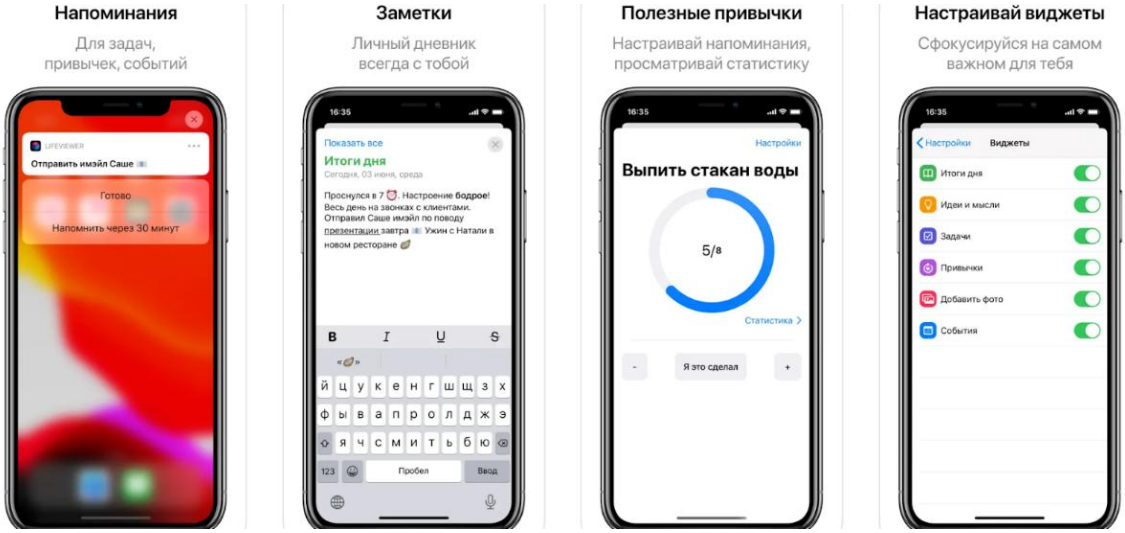

Рисунок 1 - Интерфейс LifeViewer

Нижняя панель меню распределена на дни, недели, месяцы и годы. Есть таймер, который показывает, сколько осталось времени до конца дня. Его можно отключить в настройках приложения.

Плюсы приложения: не надо скачивать дополнительные приложения, например тот же трекер привычек или заметки.

Минусы приложения: поначалу им сложно пользоваться из-за большого количества функционала, такой вывод сделан по итогам анализа отзывов пользователей по этому приложению. Также приложение доступно только в App Store. Поддерживается только IOS.

Доступно по подписке: 349 ₽ в месяц, 999 ₽ в год, а также 3090 ₽ – навсегда.

2.3.2 Brite

Это приложение схоже с LifeViewer по количеству функций. Отличие в том, что у Brite есть возможность выбирать, что нужно: сразу все функции, несколько или одна.

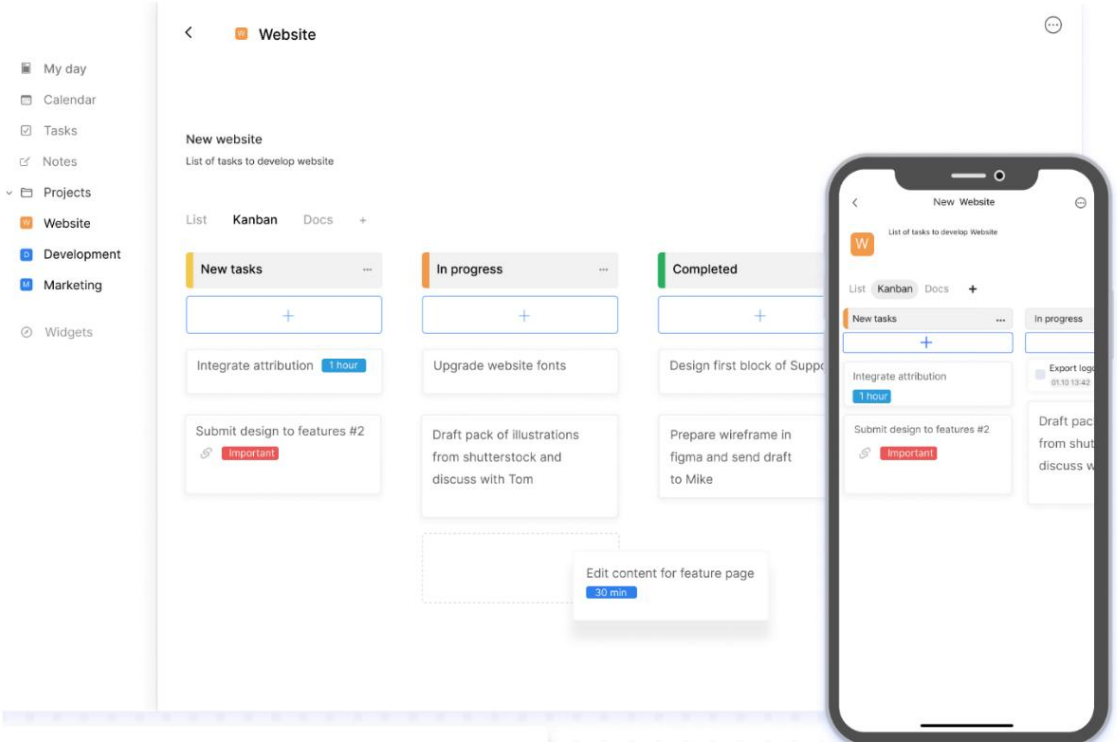

Рисунок 2 - Интерфейс Brite

У приложения есть и веб-версия, что очень удобно, если телефон не рядом. Виджеты в приложении делятся на:

— бесплатные: цели, рабочие/личные задачи, заметки, календарь;

— платные: привычки, тег и уровень сложности для задач, дневник, расходы, медитация, погода, идеи, фото дня, замеры тела и цитаты.

Плюсы приложения: Одно приложение заменяет 5–6 дополнительных. Поддерживает Web, Windows/MacOS и Android/IOS, а также Apple Watch.

Минусы приложения: большая часть функционала доступна только платно. Есть подписки: 249 ₽ в месяц, 1550 ₽ в год.

2.3.3 Any.do

Any.do давно на рынке планировщиков задач – с ноября 2011 года. Оно очень простое. В нем можно сделать необходимые списки и выполнять в них задачи. Также есть функция напоминания в «Вотсапе», что доступно при платной подписке.

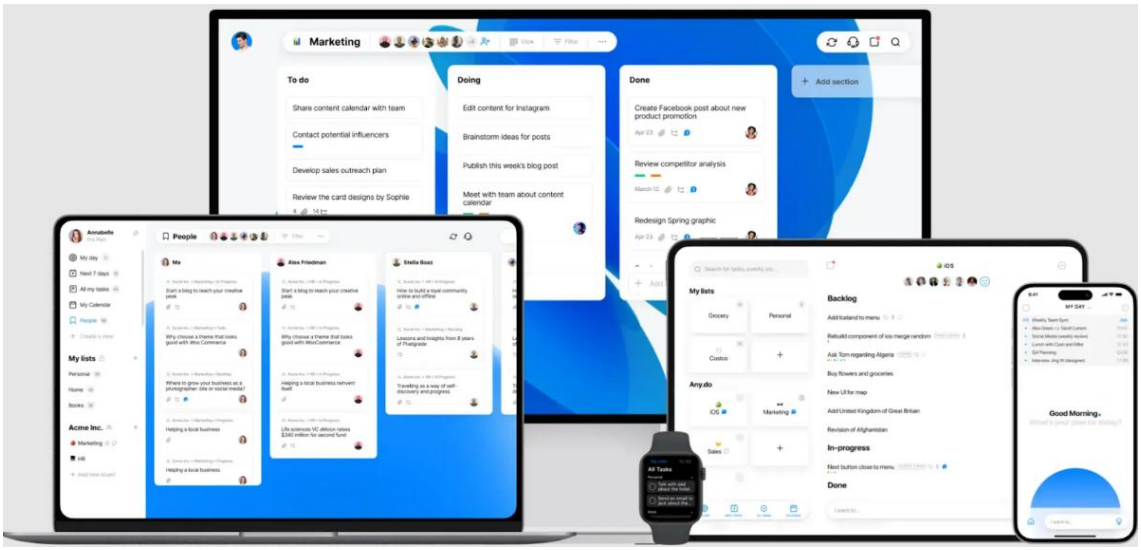

Рисунок 3 - Интерфейс Any.do

Плюсы приложения: удобный планировщик с календарем, который пользователи могут синхронизировать со встроенным. Поддерживает Web, Windows/MacOS и Android/IOS, а также Apple Watch.

Минусы приложения: из функционала присутствует только создание задач. Также есть ограничение на количество пользовательских категорий для них: можно создать максимум 5. Но если получить платную версию, то появляется возможность создавать любое количество категорий. Подписки: 459 ₽ в месяц, 1950 ₽ за полгода и 2790 ₽ в год.

2.3.4 TickTick

По функционалу схоже с Any.do: в приложении тоже можно составлять списки и вести календарь. Различие в том, что в TickTick можно

импортировать голосовых помощников – Сири, «Гугл-ассистента», Алексу (по подписке).

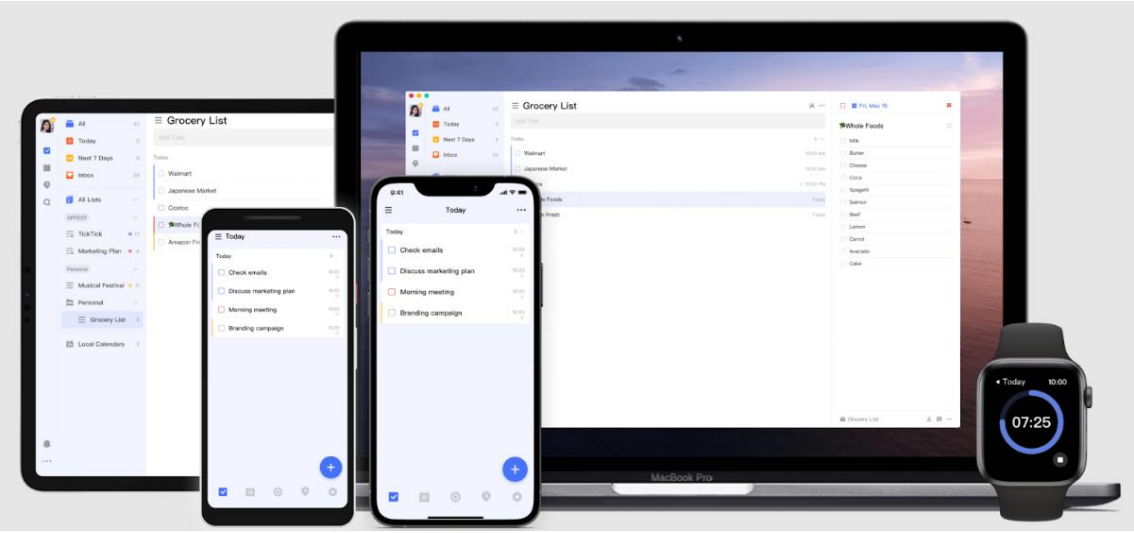

Рисунок 4 - Интерфейс TickTick

Плюсы приложения: удобный планировщик с календарем, который пользователи могут синхронизировать со встроенным. Можно расставить задачи по приоритетам.

Минусы приложения: есть возможность работы только со списком задач. Напоминания по задачам, фильтрация, добавление голосовых помощников доступно только по подписке: 199 ₽ в месяц, 1790 ₽ в год.

3 Реализация

3.1 Средства реализации

Система должна состоять из сервера приложения, реляционной базы данных, клиентской части.

Для реализации серверной части будут использоваться следующие средства:

— язык программирования Java 17;

— фреймворк Spring Boot 3;

— PostgreSQL;

— Flyway.

Java 17 является версией с долгосрочной поддержкой, а это значит, что она будет получать обновления, в том числе, связанные с безопасностью, в течение длительного времени. Также преимуществом является скорость работы: новые версии динамических компиляторов обеспечивают оптимизацию тех фрагментов кода, которые исполняется чаще.

Spring Boot — это фреймворк на основе Java с открытым исходным кодом, разработанный компанией Pivotal Software. Среди преимуществ можно отметить автоконфигурацию - это метод работы, позволяющий сократить количество действий, которые должны предпринимать разработчики. Это позволяет уделить больше внимания бизнес-логике.

PostgreSQL — это объектно-реляционная система управления базами данных, наиболее развитая из открытых СУБД в мире. Важная особенность этой базы данных заключается в том, что она обладает широким функционалом и при этом остается мощной, производительной и стабильной.

Flyway - это инструмент, который автоматизирует управление миграцией базы данных. Позволяет разработчикам эффективно управлять

схемой БД и переносить ее по мере ее развития на различных этапах, таких как проектирование, тестирование, производство и т.д.

Для реализации клиентской части мобильного приложения и сервисного веб-приложения будут использоваться следующие средства:

— язык программирования Dart;

— фреймворк Flutter.

Dart — язык программирования, который создали для использования в веб-разработке. Самым главным преимуществом является компилируемость в бинарный код, за счет чего достигается скорость выполнения операций сравнимая с Swift, Java, или Kotlin.

Flutter — это открытая кросс-платформенная технология от Google, предназначенная для создания приложений для iOS, Android, Web. Его основные преимущества заключаются в единой кодовой базе и высокой производительности.

3.2 Графическое описание работы системы

3.2.1 Диаграмма прецедентов

На рисунке 5 представлена UML диаграмма вариантов использования, иллюстрирующая функционал администратора.

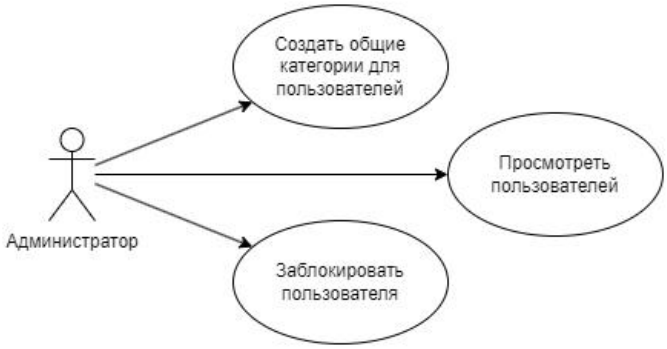

Рисунок 5 - Функционал для администратора

На рисунке 6 представлена UML диаграмма вариантов использования, иллюстрирующая функционал неавторизированного пользователя.

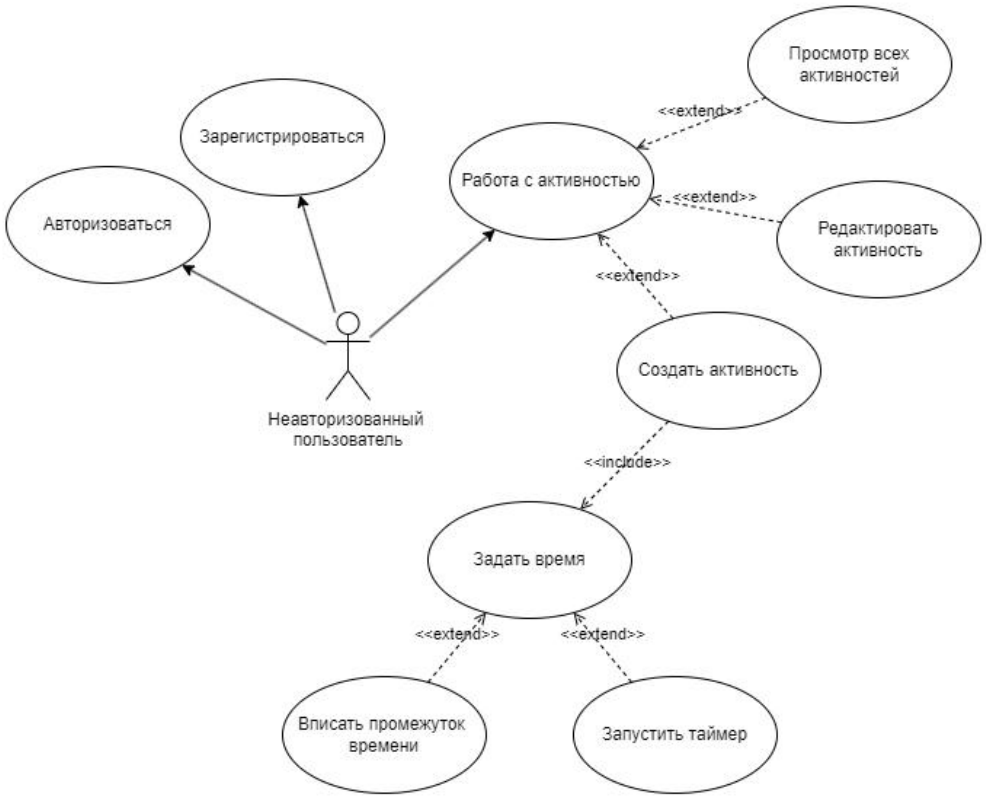

Рисунок 6 - Функционал для неавторизированного пользователя

На рисунке 7 представлена UML диаграмма вариантов использования, иллюстрирующая функционал авторизированного пользователя.

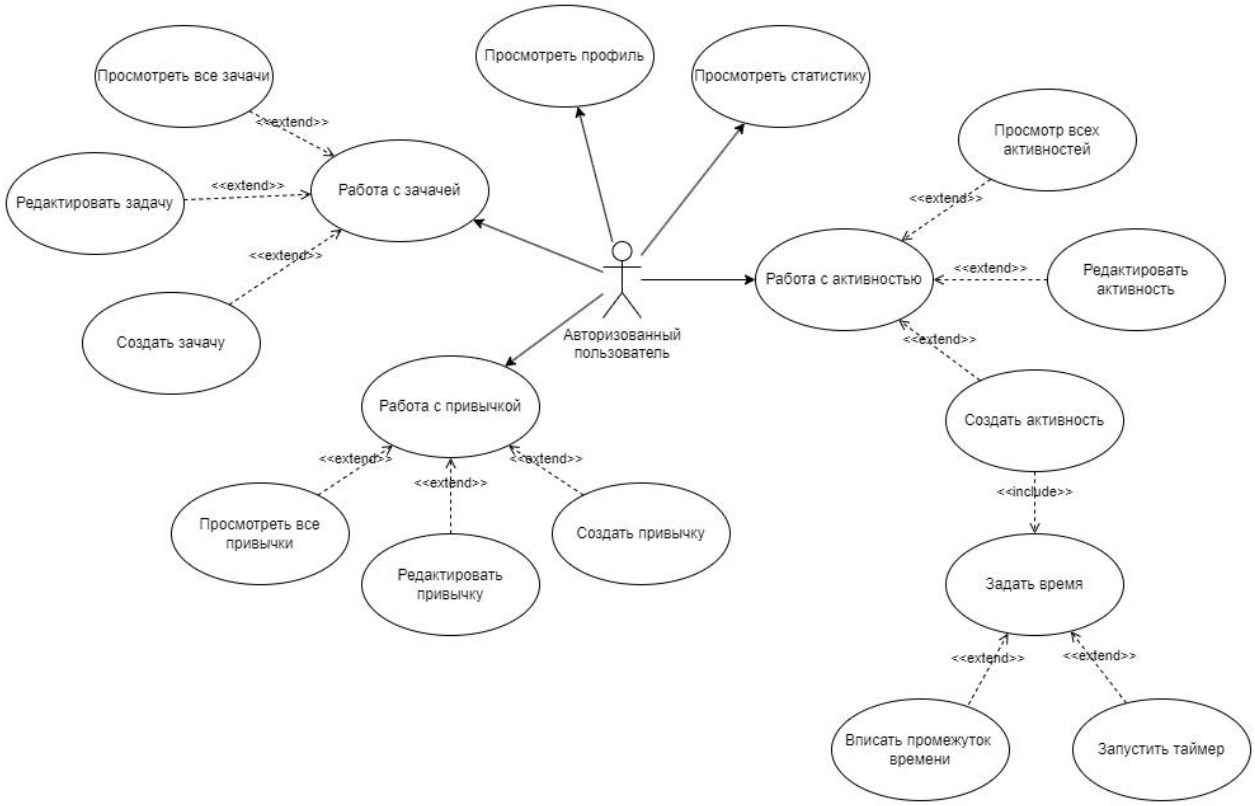

Рисунок 7 - Функционал для авторизированного пользователя

3.2.2 Диаграмма состояний

На рисунке 8 представлена UML диаграмма состояний, иллюстрирующая состояния сущностей задачи и привычки.

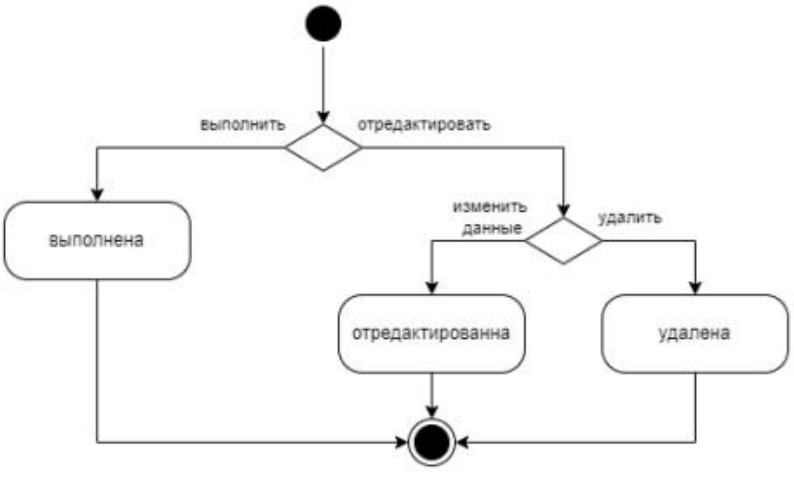

Рисунок 8 - Диаграмма состояний

3.2.3 Диаграмма последовательности

На рисунке 9 представлена UML диаграмма последовательности, иллюстрирующая жизненный цикл создания активности.

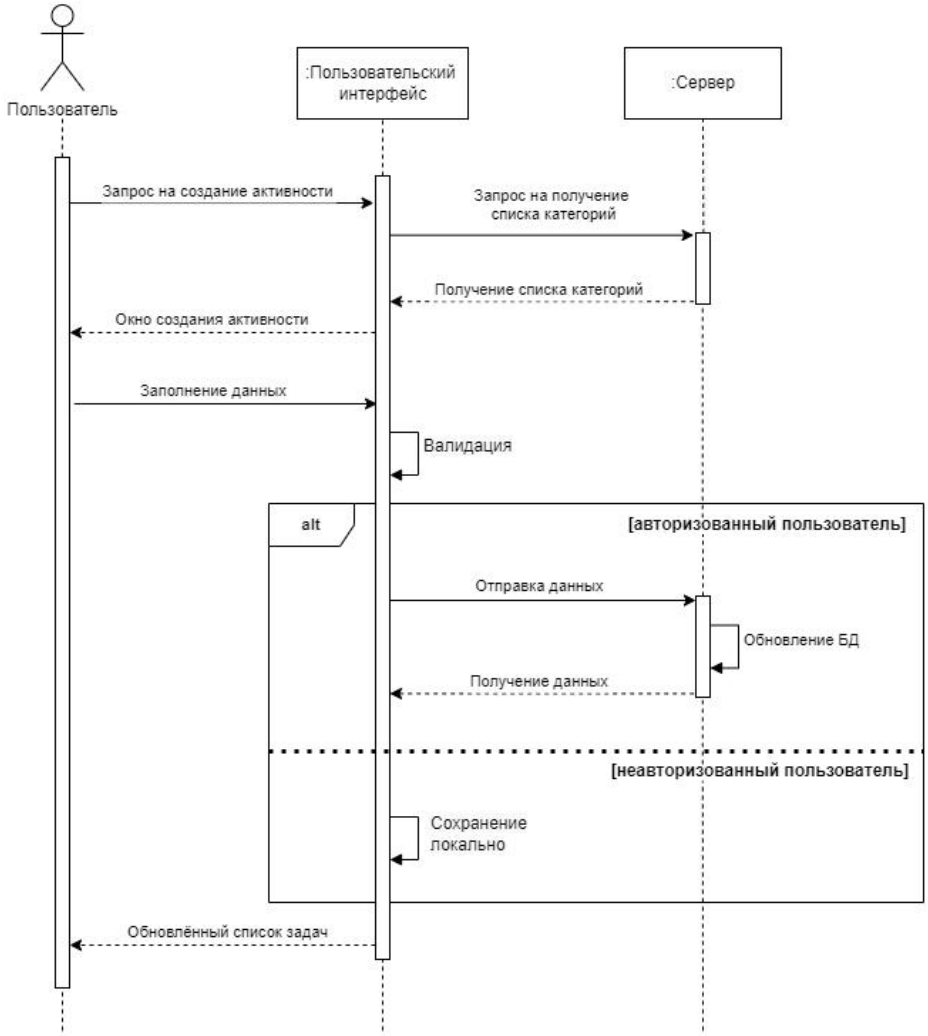

Рисунок 9 - Диаграмма последовательности для активности

3.2.4 ER диаграмма

На рисунке 10 представлена UML ER диаграмма, иллюстрирующая отношения сущностей.

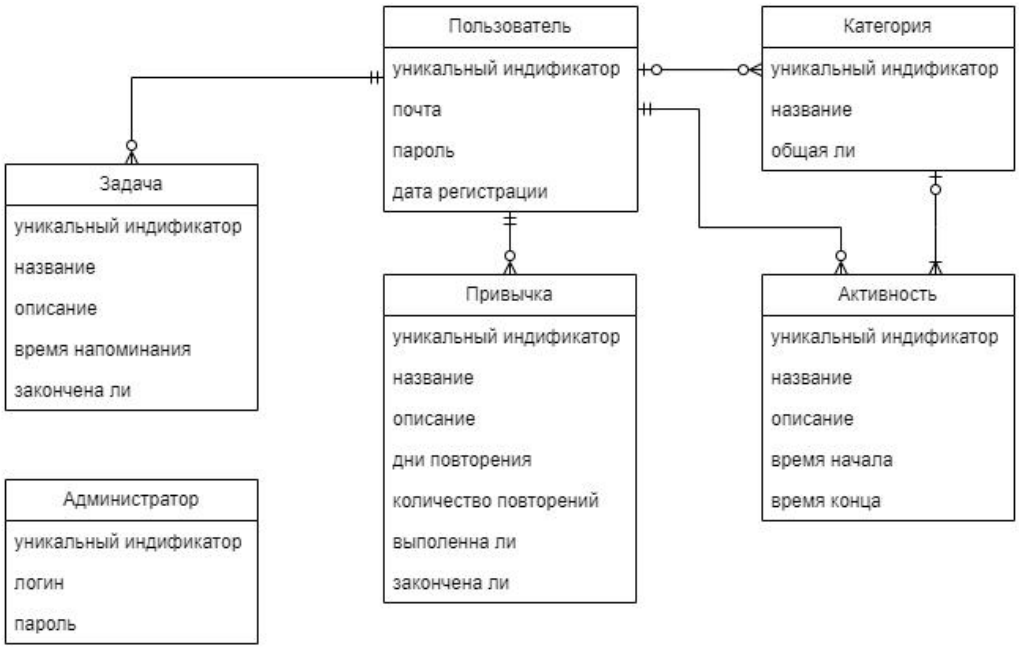

Рисунок 10 - ER диаграмма

3.2.5 Диаграмма развертываний

На рисунке 11 представлена UML диаграмма развертываний приложения.

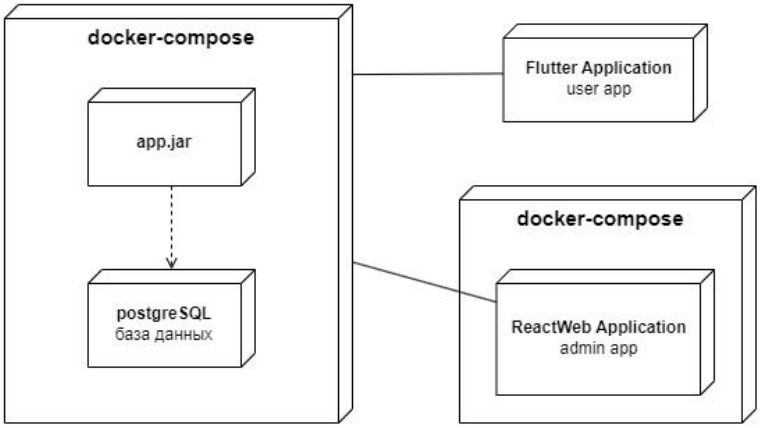

Рисунок 11 - Диаграмма развертываний

3.2.6 Диаграмма классов

На рисунке 12 представлена UML диаграмма классов сущностей.

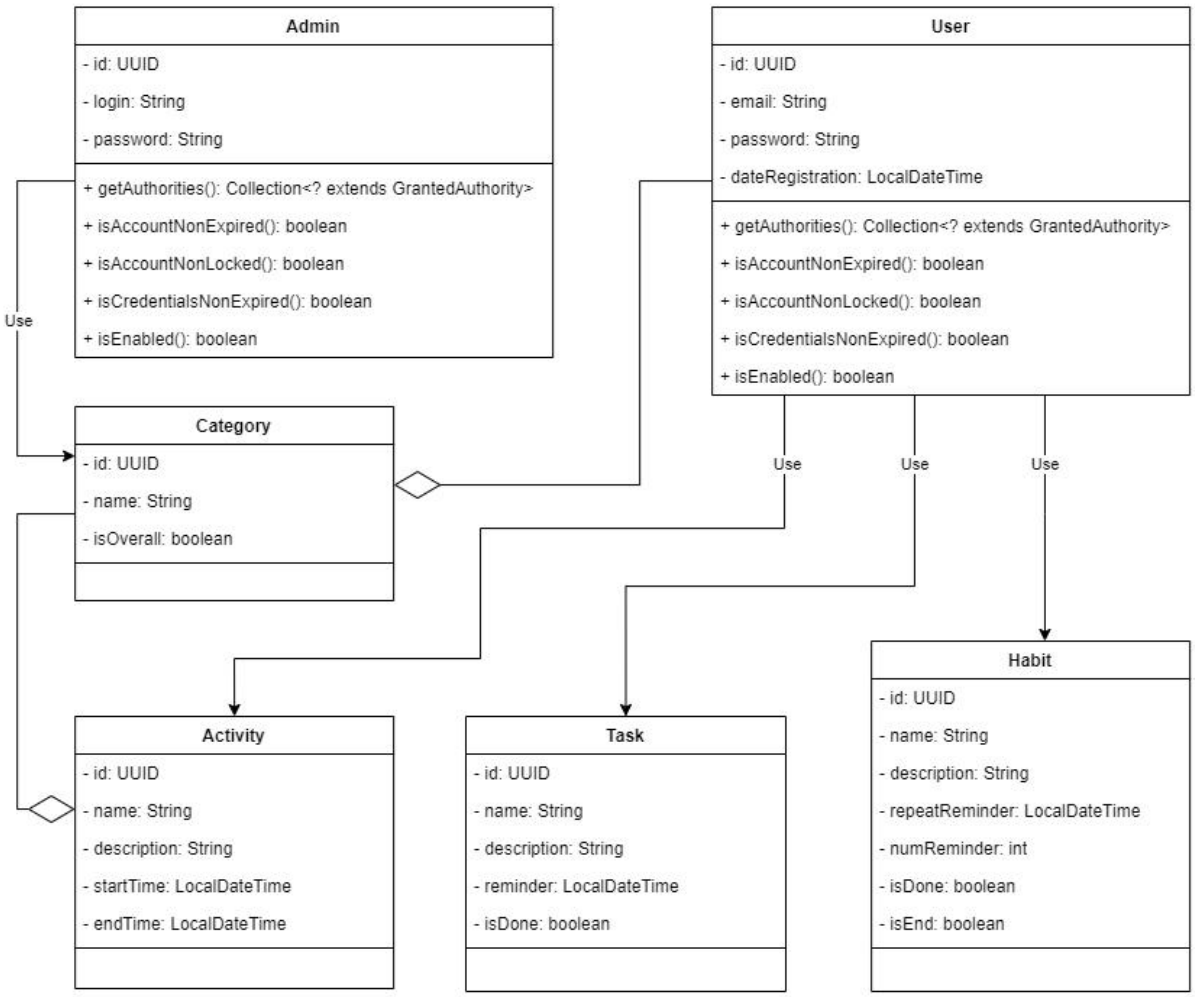

Рисунок 12 - Диаграмма класов сущностей

3.3 Архитектура приложения

Приложение имеет архитектуру, соответствующую модели клиент-серверного взаимодействия. Она представлена на рисунке 13.

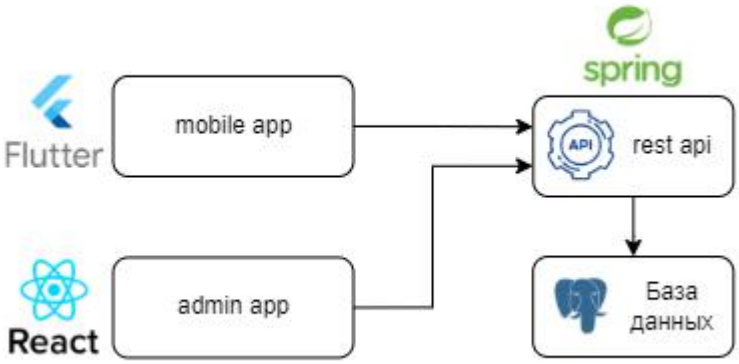

Рисунок 13 - Архитектура клиент-сервеного приложения

В качестве клиента выступает мобильное приложение, написанное с помощью фреймворка Flutter и браузерное приложение, написанное с помощью библиотеки React.

Для реализации серверной части использовался архитектурный подход REST API. Он позволяет клиенту обращаться к методам сервера с помощью HTTP-протокола, используя формат JSON.

3.4 Реализация серверной части приложения

3.4.7 Слой доступа к данным

Слой доступа к данным (Data Access Layer) - это слой абстракции, который управляет доступом к данным в приложении. С помощью него можно отделить зависимости между кодом, работающим с данными, и представлением, что делает код более универсальным и пригодным для повторного использования.

JPA Repository - это интерфейс, который предоставляет различные CRUD методы (Create, Read, Update, Delete) для работы с базой данных. Он автоматически генерирует SQL-запросы на основе методов, определённых в репозитории, что уменьшает объём необходимого кода.

Реализация слоя доступа к данным представлена на рисунке 14:

Рисунок 14 - Слой доступа к данным

3.4.8 Слой контроллеров

Слой контроллеров (Controller) является одним из трёх компонентов архитектуры MVC в Spring Boot Framework. Он отвечает за извлечение данных из запросов, передачу их внутрь бизнес-логики и формирование ответа на запрос.

Реализация слоя контроллеров представлена на рисунке 15:

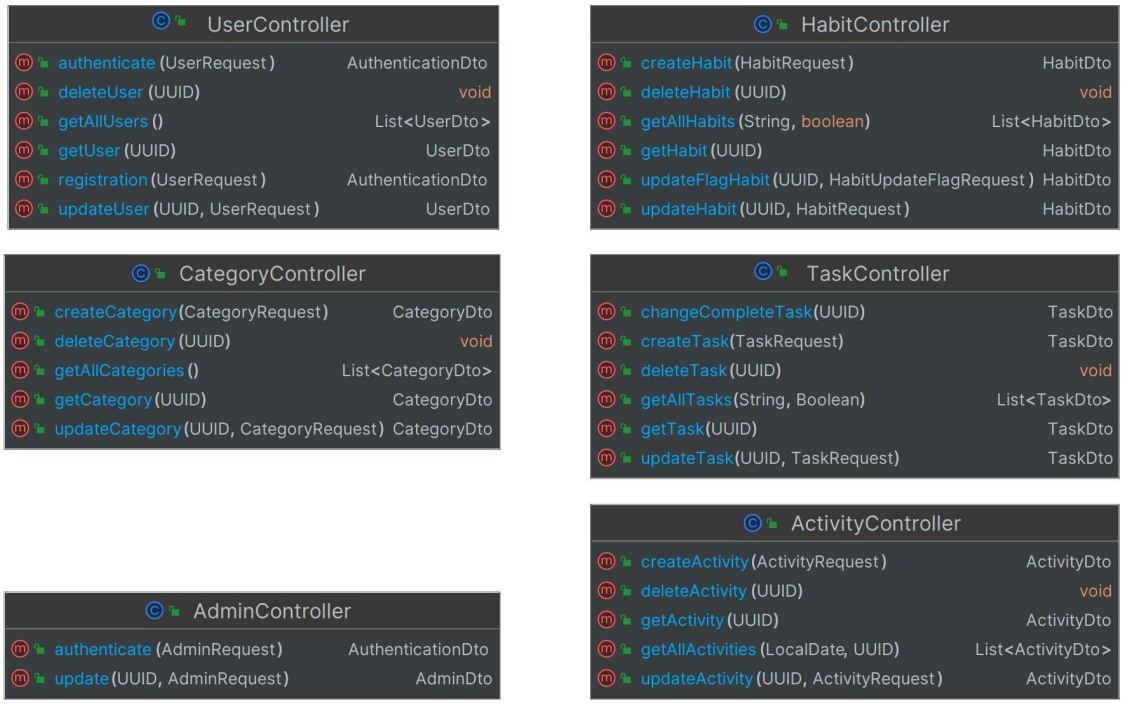

Рисунок 15 - Реализация слоя контоллеров

3.4.9 Слой модели

Слой моделей (Model) является одним из трёх компонентов архитектуры MVC в Spring Boot Framework. Он содержит классы, представляющие различные сущности в приложении.

Реализация слоя моделей представлена на рисунке 16:

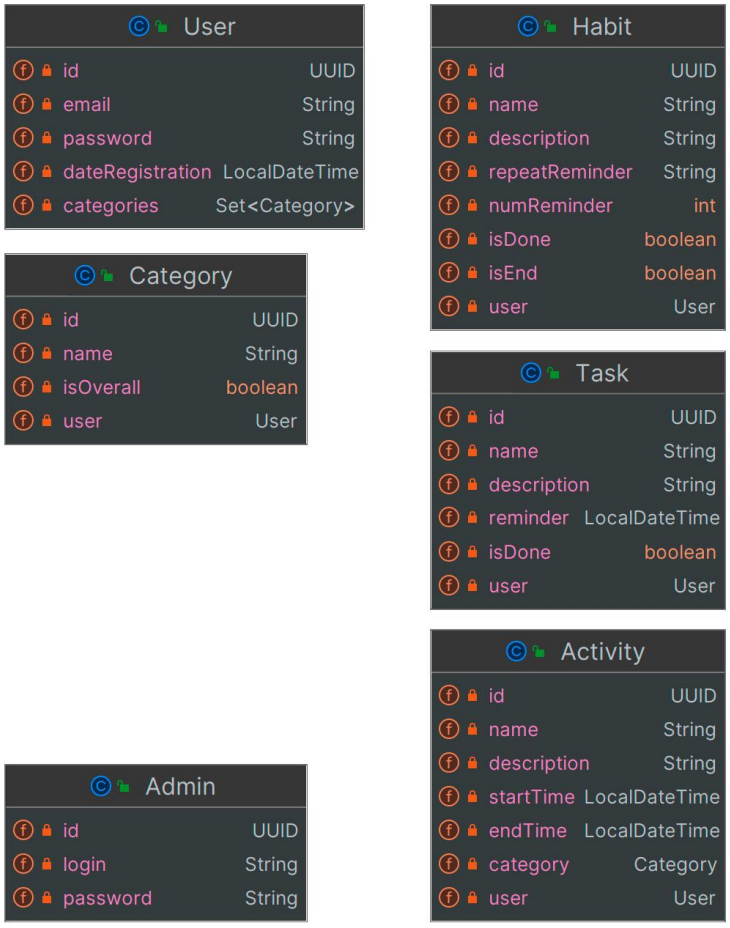

Рисунок 16 - Реализация слоя моделей

3.4.10 Слой бизнес-логики

Слой сервисов в бизнес-логике является промежуточным слоем между слоем контроллеров и слоем моделей в MVC-архитектуре приложения. Он отвечает за бизнес-логику, определяющую, как данные будут обработаны.

Реализация слоя бизнес-логики представлена на рисунке 17:

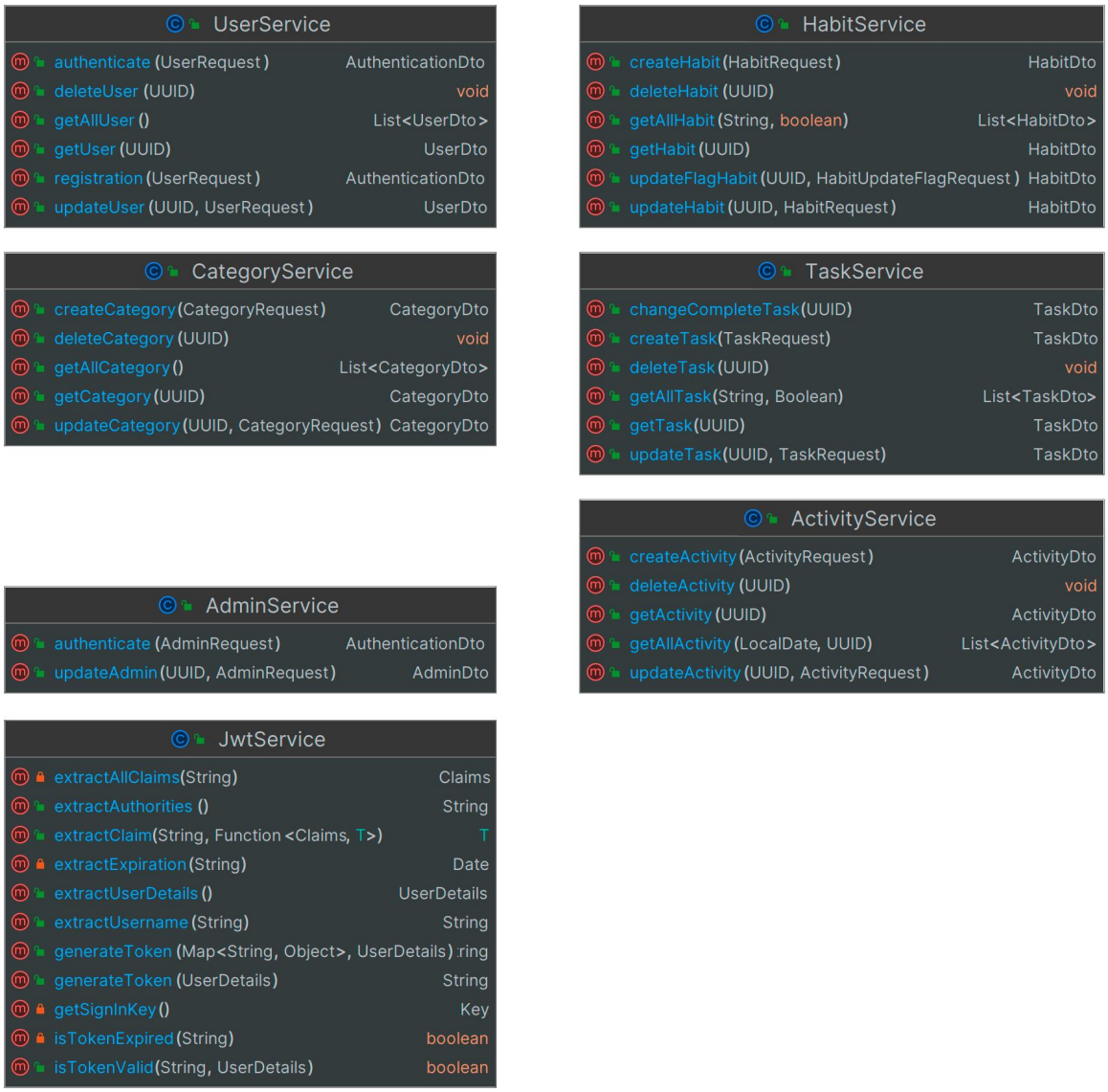

Рисунок 17 - Реализация слоя бизнес-логики

3.4.11 Механика работы приложения

Для корректной работы приложения были реализованы несколько сервисов.

UserService отвечает за регистрацию, авторизацию, получение и изменение данных пользователя. Основные методы сервиса:

— registration(). Отвечает за регистрацию нового пользователя, используя введенные им почту и пароль. Возвращает сгенерированный токен для прохождения аутентификации;

— authenticate(). Отвечает за аутентификацию пользователя. Возвращает сгенерированный токен для начала новой сессии в приложении;

— getUser(). Возвращает данные пользователя по его id;

— updateUser(). Обновляет почту и пароль пользователя по его id. Возвращает измененные данные.

HabitServer отвечает за создание, обновление, получение и удаление привычек пользователя, а также за изменение их статусов. Основные методы сервиса:

— createHabit(). Отвечает за создание новой привычки, используя введеные пользователем данные, формирует крон выражение для напоминаний. Возвращает созданную привычку;

— updateHabit(). Обновляет название, описание у существующей привычки по ее id и cron-выражение для напоминаний. Возвращает обновленную привычку;

— updateFlugHabit(). Управляет флагами состояния привычки: продолжает выполняться или же завершена, и напоминаниями. Так например, если привычка отмечена как завершенная, то уведомления о ней больше не будут приходить. Возвращает обновленную привычку;

— getAllHabit(). Возвращает список всех привычек пользователя с возможностью фильтрации по дням недели;

— deleteHabit(). Удаляет привычку по её id.

TaskService отвечает за создание, обновление, получение и удаление задач пользователя, а также за изменение их статусов. Основные методы сервиса:

— createTask(). Отвечает за создание новой задачи, используя введенные пользователем данные. Возвращает созданную задачу;

— updateTask(). Обновляет название, описание и время напоминания у существующей задачи по ее id. Возвращает обновленную задачу;

— changeCompleteTask(). Изменяет статус выполнения задачи и возвращает обновленную задачу;

— getAllTask(). Возвращает список всех задач пользователя с возможностью сортировки по времени напоминания и фильтрации по статусу выполнения;

— deleteTask(). Удаляет задачу по ее id.

ActivityService отвечает за создание, обновление, получение и удаление активностей пользователя. Основные методы сервиса:

— createActivity(). Отвечает за создание новой активности, используя введенные пользователем данные. Возвращает созданную активность;

— updateActivity(). Обновляет название, описание, время начала и окончания, категорию у существующей активности по ее id. Возвращает обновленную активность;

— getAllActivity(). Возвращает список всех активностей пользователя по заданной дате с возможностью фильтрации по категориям;

— deleteActivity(). Удаляет активность по её id.

CategoryService отвечает за создание, обновление, получение и удаление категорий пользователя. Основные методы сервиса:

— createCategory(). Отвечает за создание новой категории, используя введенные пользователем данные. Если пользователь является администратором, то категория создается как стандартная, то есть появляется у всех пользователей без возможности ее редактирования, иначе она привязывается к конкретному человеку. Возвращает созданную категорию;

— updateCategory(). Обновляет существующую категорию по ее id. Если категория стандартная, ее может редактировать только администратор, если личная, то ее может обновлять только пользователь. Возвращает обновленную категорию;

— getAllCategory(). Возвращает список всех категорий. Администратор получает все общедоступные категории, а пользователь получает все общедоступные и личные категории;

— deleteCategory(). Удаляет категорию по её id. Если категория общедоступная, ее может удалять только администратор, если личная, то ее может удалять только пользователь.

3.5 Реализация клиентской части приложения

3.5.12 Реализация активностей

Для работы с активностями были реализованы следующие основные классы, составляющие основу проекта управления активностями, обеспечивая полный цикл работы с ними.

Класс ActivityList представляет собой основной виджет, который отвечает за отображение списка активностей. Основные функции класса:

- getCategories(). Используется для получения списка категорий, которые могут быть присвоены активностям. Она извлекает категории из состояния _ActivityListState;
- createState(). Создает и возвращает объект состояния _ActivityListState, который содержит основную логику и данные для работы с активностями.

Класс _ActivityListState реализует состояние для ActivityList, где хранится вся логика и данные, необходимые для управления списком активностей. Основные функции класса:

- _deleteActivity(). Удаляет активность из списка по заданной дате и индексу. Эта функция обновляет состояние списка, удаляя соответствующую активность;
- _addActivity(). Добавляет новую активность с указанными параметрами. Эта функция обновляет состояние, добавляя новую активность в список.

Класс EditActivityScreen отвечает за предоставление интерфейса для редактирования существующей активности. Основные функции:

- createState(). Создает и возвращает объект состояния _EditActivityScreenState, который содержит логику и данные для редактирования активности.

3.5.13 Реализация задач

Для работы с задачами были реализованы следующие основные классы, составляющие основу проекта управления задачами, обеспечивая полный цикл работы с ними.

Класс MyHomePage представляет собой основной виджет, который отвечает за отображение списка задач. Класс _MyHomePageState реализует состояние для MyHomePage, где хранится вся логика и данные, необходимые для управления списком задач. Основные функции класса:

— _deleteTask(). Удаляет задачу из списка по индексу. Эта функция обновляет состояние списка, удаляя соответствующую задачу;

— _addNewTask(). Добавляет новую задачу с указанными параметрами. Эта функция обновляет состояние, добавляя новую задачу в список;

— _editTask(). Открывает экран редактирования задачи, позволяя пользователю изменять её;

— _viewTask(). Открывает экран с подробной информацией о задаче;

— _toggleTaskCompletion(). Переключает состояние выполнения задачи (завершена/незавершена);

— _sortTasks(). Сортирует задачи по дате выполнения в зависимости от состояния isAscending.

Класс TaskTile представляет собой виджет для отображения одной задачи в списке. Класс TaskDetailScreen используется для отображения подробной информации о задаче.

3.5.14 Реализация привычки

Класс HabitCreationScreen предоставляет интерфейс для создания или редактирования привычки.

В него входят следующие методы:

— initState(). Инициализирует состояние виджета. Если переданы данные о привычке, инициализирует поля с этими данными. Иначе устанавливает значения по умолчанию;

— createHabit(). Проверяет корректность введённых данных и возвращает объект HabitData с введёнными данными, если все проверки пройдены;

— showAlertDialog(). Отображает диалоговое окно с предупреждением, если какие-то данные не введены;

— toggleDay(). Переключает выбранный день недели. Если день уже выбран, он будет удалён из набора selectedDays. Если не выбран, он будет добавлен;

— selectAllDays(). Выбирает или отменяет выбор всех дней недели. Если все дни уже выбраны, они будут сброшены. Если не все дни выбраны, будут выбраны все дни недели.

Класс HabitData. Этот класс используется для хранения данных о привычке. Класс HabitTrackerScreen позволяет пользователю добавлять, просматривать и управлять своими привычками.

Класс HabitsList представляет список привычек и отвечает за их отображение и сортировку. В него входят следующие методы:

— getSortedHabits(). Возвращает список привычек, отсортированных по состоянию выполнения (сначала невыполненные, затем выполненные);

— buildHabitListTile(). Создает виджет для отдельной привычки;

— getSelectedDaysString(). Возвращает строку с перечислением выбранных дней недели.

4 Интерфейс приложения

4.1 Первый вход в приложение

При первом входе пользователь знакомится с функциональными возможностями приложения с помощью краткого обзора, экраны которого представлены на рисунке 18.

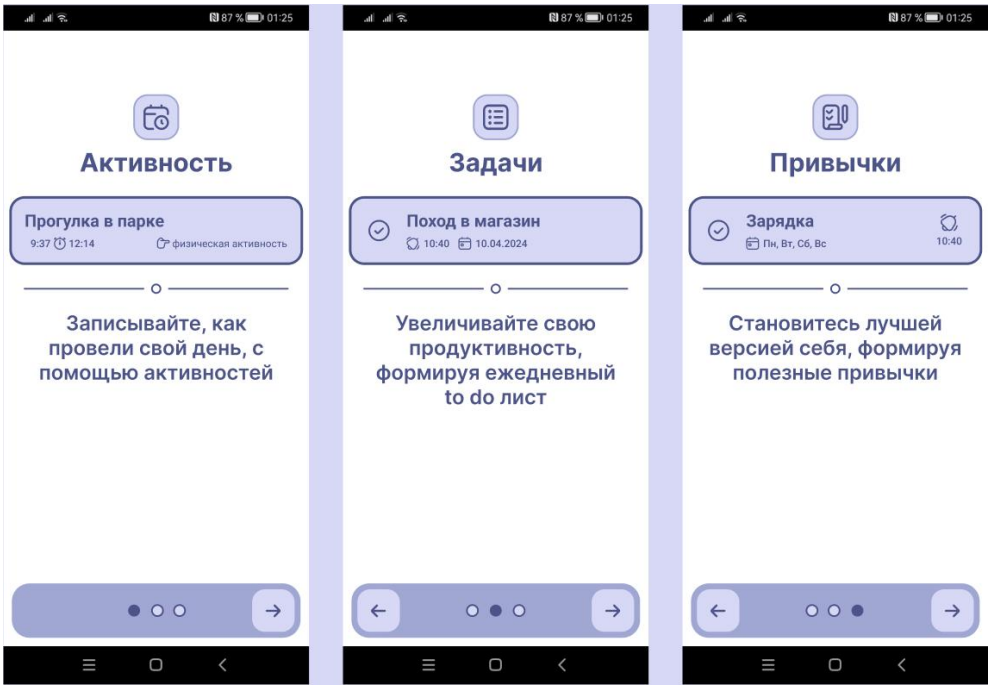

Рисунок 18 - Краткий обзор возможностей приложения

После последнего экрана пользователь переходит на экран с регистрацией.

4.2 Регистрация и авторизация

Экраны регистрации и авторизации представлены на рисунке 19.

Для регистрации пользователю необходимо ввести почту и пароль. После нажатия на кнопку «Создать профиль» он перенаправляется на экран входа. При авторизации также необходимо ввести почту и пароль. После входа в приложение пользователь попадает на экран с активностями.

Если пользователь не хочет проходить регистрацию, то он может выбрать «Продолжить как гость» и получить возможность работать только с активностями.

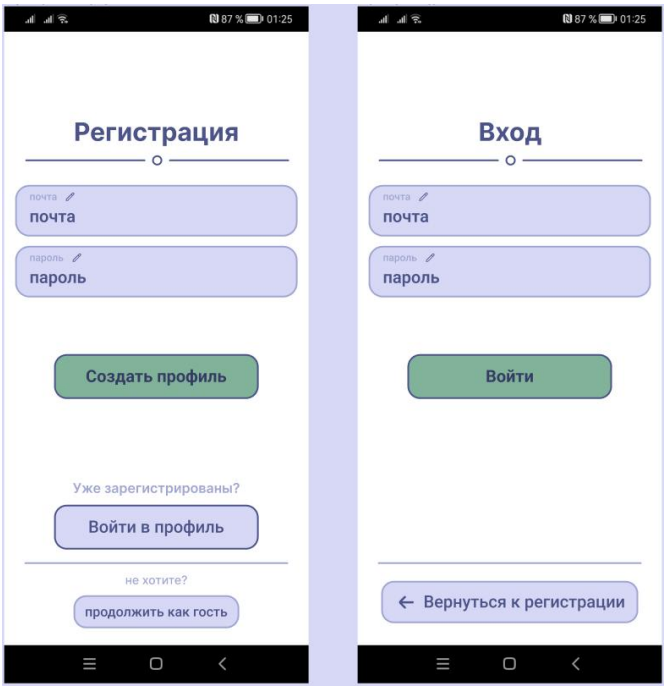

Рисунок 19 - Экраны регистрации и авторизации

4.3 Раздел активностей

Экраны для работы с активностями представлены на рисунке 20.

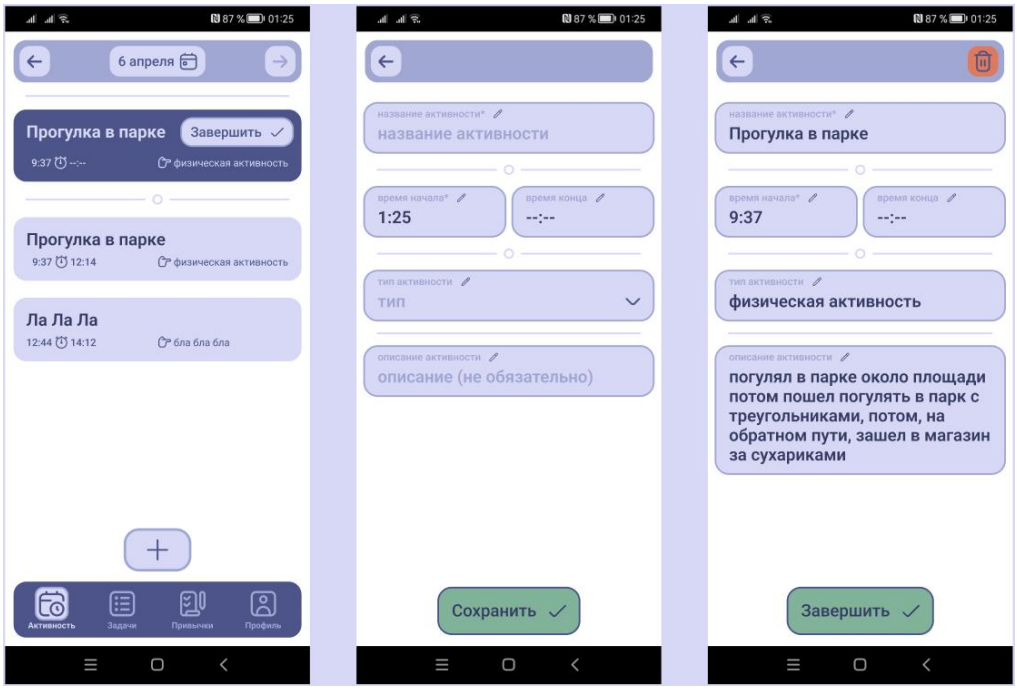

Рисунок 20 - Экраны для работы с активностями

На первом экране демонстрируется список активностей за текущий или выбранный день. Сначала располагаются те, которые пока не были закончены, затем выполненные.

Для создания новой активности необходимо ввести название и время начала. Время окончания можно ввести позже, описание же вводить необязательно.

Незавершенные активности выделенны более темным цветом. Для завершения необходимо перейти в режим редактирования, ввести время окончания и нажать кнопку «Сохранить».

4.4 Раздел привычек

Экраны для работы с привычками представлены на рисунке 21.

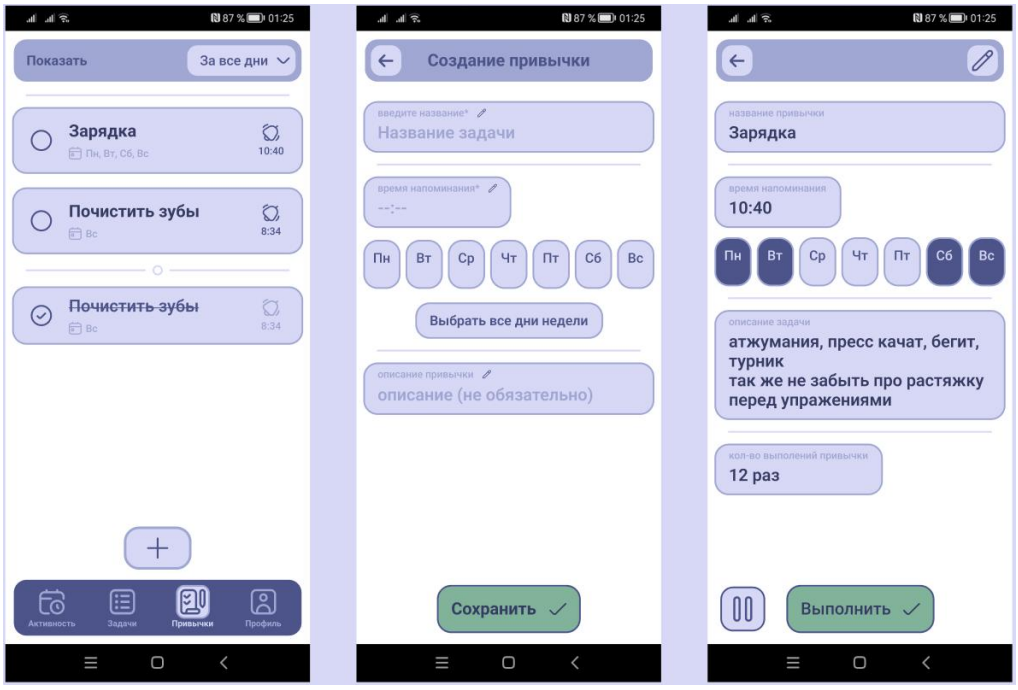

Рисунок 21 - Экраны для работы с привычкой

На первом экране демонстрируется список всех привычек пользователя с временем напоминания и днями, по которым она должна выполняться.

Для создания привычки необходимо нажать на иконку плюса и ввести название, время напоминания и выбрать дни, в которые должна она выполняться.

Незавершенную привычку можно либо отметить выполненной в конкретный день путем нажатия кнопки «Выполнить», что увеличит счетчик «Количество выполнений привычки», либо сделать ее завершенной путем нажатия на иконку паузы, т.е привычка будет считаться сформированной у пользователя.

4.5 Раздел задач

Экраны для работы с задачами представлены на рисунке N.

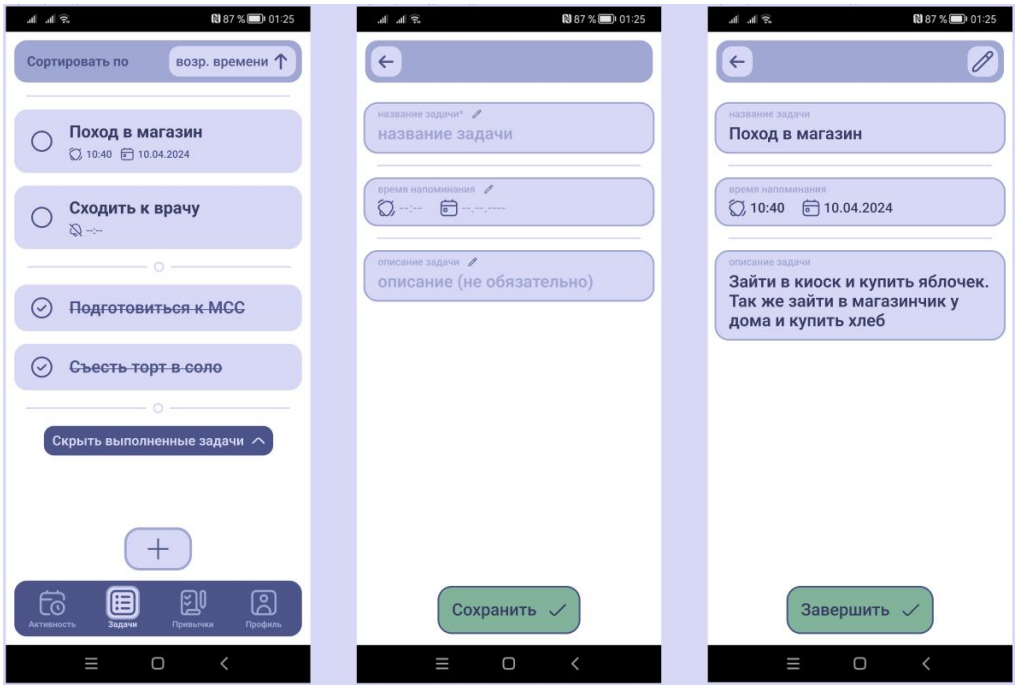

Рисунок 22 - Экраны для работы с задачами

На первом экране демонстрируется список всех задач пользователя. Их можно отсортировать по времени выполнения, а также скрыть завершенные.

Для создания новой задачи обязательно указать название. Время и дата напоминания, а также описание не является обязательным.

Для завершения задачи необходимо перейти в режим ее редактирования и нажать кнопку «Завершить».

4.6 Профиль пользователя

Экраны профиля пользователя представлены на рисунке N.

Перейдя в профиль, пользователь видит прогресс по своим привычкам, а также может изменить свои личные данные или добавить новую категорию для активностей.

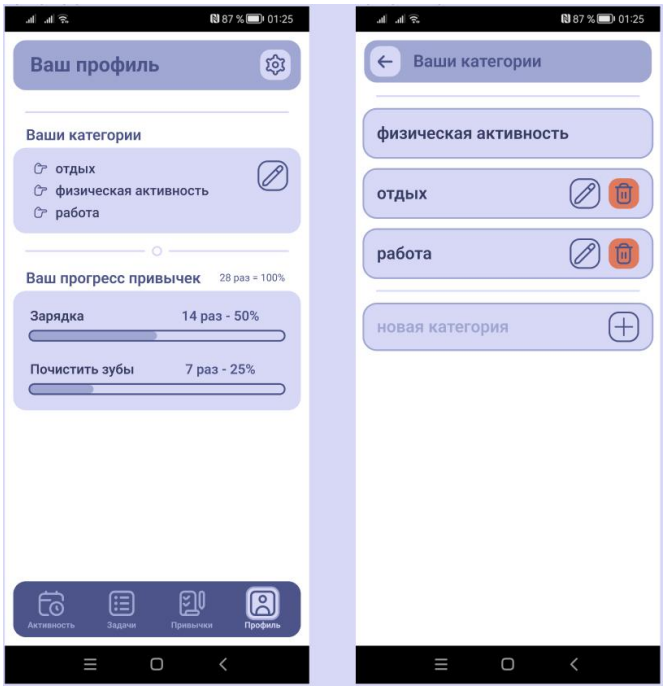

Рисунок 23 - Экраны профиля

Заключение

В ходе выполнения данной курсовой работы было реализовано приложение, выполняющее следующие задачи:

- ведение и просмотр записей о использовании времени. Пользователь может создавать записи о деятельности, указывать время начала и конца, категорию и описание;
- планирование задач. Пользователь может создавать задачи,указывая сроки, и отмечать их как выполненные;
- отслеживание привычек. Пользователь может создавать новые привычки, указывая периодичность;
- анализ использования времени. Пользователь может просмотреть статистику использования своего времени.

Список использованных источников

1. Почему Flutter является предпочтительным выбором для разработки: [сайт] – URL: https://stfalcon.com/ru/blog/post/why-flutter-is-the-preferred-choice-for-development (дата обращения 20.05.2024). – Текст : электронный.
2. Что такое Dart: [сайт] – URL: https://blog.skillfactory.ru/glossary/dart/ (дата обращения 21.05.2024). – Текст : электронный.
3. Flutter: плюсы и минусы использования кросс-платформенной технологии: [сайт] – URL: https://tproger.ru/articles/flutter-plyusy-i-minusy-ispolzovaniya-kross-platformennoj-tehnologii (дата обращения 21.05.2024). – Текст : электронный.
4. Про Flutter: [сайт] – URL: https://habr.com/ru/articles/430918/ (дата обращения 22.05.2024). – Текст : электронный.
5. Spring Boot – Flyway Database: [сайт] – URL: https://www.geeksforgeeks.org/spring-boot-flyway-database/ (дата обращения 25.05.2024). – Текст : электронный.
6. Миграции и контроль версий БД с помощью Flyway: [сайт] – URL: https://ealebed.github.io/posts/2020/flyway-version-control-for-db/ (дата обращения 25.05.2024). – Текст : электронный.
7. Spring Boot Documentation: [сайт] – URL: https://docs.spring.io/spring-boot/index.html (дата обращения 26.05.2024). – Текст : электронный.