

МИНОБРНАУКИ РОССИИ  
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ  
ОБРАЗОВАТЕЛЬНОЕ  
УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ  
«ВОРОНЕЖСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»

Факультет Компьютерных наук

Кафедра программирования и информационных технологий

Разработка мобильного приложения для анализа того, как пользователь  
проводит свой день и предлагает пути для повышения продуктивности  
«YourDay»

Курсовая работа

Направление: 09.03.04. Программная инженерия

Зав. Кафедрой \_\_\_\_\_ д. ф.-м. н, доцент С.Д. Махортов  
Руководитель \_\_\_\_\_ ст. преподаватель В.С. Тарасов  
Руководитель практики \_\_\_\_\_ Г.В. Прядченко  
Обучающийся \_\_\_\_\_ С.И. Илюнов, 3 курс, д/о  
Обучающийся \_\_\_\_\_ Д.Д. Середа, 3 курс, д/о  
Обучающийся \_\_\_\_\_ А.В. Гончаренко, 3 курс, д/о  
Обучающийся \_\_\_\_\_ А.Э. Долгушина, 3 курс, д/о  
Обучающийся \_\_\_\_\_ А.Э. Кузнецов, 3 курс, д/о  
Обучающийся \_\_\_\_\_ Я.В. Белозеров, 3 курс, д/о

Воронеж 2025

# СОДЕРЖАНИЕ

ВВЕДЕНИЕ.....	4
1 Постановка задачи.....	5
1.1 Цели курсовой работы.....	5
1.2 Задачи приложения .....	5
1.3 Требования к приложению.....	6
1.3.1 Функциональные требования .....	6
1.3.2 Нефункциональные требования .....	7
2 Анализ предметной области .....	10
2.1 Глоссарий.....	10
2.2 Обзор аналогов .....	11
2.3 StayFree.....	11
2.4 ActionDash.....	13
2.5 Экранное время + Блокировка.....	15
2.6 Digitox.....	17
3 Реализация.....	20
3.1 Средства реализации.....	20
3.2 Реализация серверной части .....	21
3.3 Основные модели в базе данных .....	21
3.4 Сервисный слой.....	23
3.5 Слой безопасности .....	25
3.6 Контроллеры.....	25
4 Развертывание серверной части .....	26
4.1 Реализация клиентской части .....	26
4.2 Обмен данными с сервером .....	28
4.3 Реализация интерфейса .....	29
4.3.1 Экран статистики .....	29
4.3.2 Экран блокнота .....	30
4.3.3 Экран списка целей.....	31

5 Перспективы развития .....	33
ЗАКЛЮЧЕНИЕ .....	34
СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ .....	35
ПРИЛОЖЕНИЕ А .....	36

## **ВВЕДЕНИЕ**

В текущую эпоху высокоразвитых технологий, мобильные устройства такие как смартфоны и планшеты стали неотъемлемой частью в жизни практически каждого человека, согласно данным аналитической платформы Statista, в 2024 году в России насчитывалось примерно 104,98 миллиона пользователей смартфонов, а к 2029 году их число, согласно прогнозам, достигнет 122,08 миллиона.

В связи с этим остро встает вопрос об эффективности использования смартфонов. В современном мире, где каждая минута на счету, умение эффективно распоряжаться своим временем становится не просто желательным навыком, а необходимостью. На первый взгляд устройство очень полезно, в нем уместается вся информация доступная в мире, мгновенная связь и коммуникация, однако с постепенным прогрессом этого вида техники, все чаще смартфоны зачастую становятся причиной потери времени, а также некачественного управления временем и рассеивания внимания. Поэтому актуальной темой на данный момент является анализ того, как человек проводит время за смартфоном, а также поиска путей повышения качества этого времяпровождения.

Целью исследования данной темы является разработка мобильного приложения для анализа того, как пользователь проводит день, и предлагает пути для повышения продуктивности. Оно поможет не только отслеживать текущую эффективность работы, но и даст пользователю рекомендации, опираясь на собранные данные. Это позволит оптимизировать рабочие процессы и выявлять «узкие» места, требующие дополнительного внимания или корректировки.

## **1 Постановка задачи**

### **1.1 Цели курсовой работы**

Целями выполнения курсовой работы созданию мобильного приложения являются:

- реализация системы, которая позволит пользователям отслеживать и анализировать свою активность в течение дня. Данные предоставляются на основе статистики по времени, проведенному в приложениях, количества шагов от шагомера, записям в блокноте и статистики по выполненным целям. Основываясь на этих данных и рекомендациях от искусственного интеллекта по составлению целей на день, который анализирует описанную выше информацию, пользователи смогут эффективно составлять цели на день и увеличивать качество времени, проведенного в течение дня;
- получение прибыли с помощью премиум подписки.

### **1.2 Задачи приложения**

Приложение позволит решать следующие задачи:

- получение статистики по времени, использования приложений;
- получение количества пройденных шагов за день;
- добавление записи в блокнот;
- добавление цели в список целей;
- осуществление ограничений приложений;
- запуск таймера фокусировки для повышения продуктивности;
- получение мотивационных уведомлений при достижении цели;
- получение периодических напоминаний о выполнении поставленных целей;

— активация автоматического продления премиум подписки.

### **1.3 Требования к приложению**

#### **1.3.1 Функциональные требования**

Данное приложение должно соответствовать следующим функциональным требованиям:

Неавторизованный пользователь должен иметь следующие возможности:

- авторизация в системе;
- регистрация в системе;
- восстановление пароля;
- просмотр статистики за три дня;
- возможность ограничить доступ к шести приложениям;
- возможность запустить таймер фокусировки до двух часов.

Авторизованный пользователь должен иметь следующие возможности:

- выход из профиля;
- просмотр страницы профиля;
- редактирование данных профиля;
- просмотр страницы, содержащей информацию о подписке и тарифных планах;
- оформление подписки;
- просмотр страницы с целями;
- ограниченное количество рекомендаций по составлению целей от искусственного интеллекта, две активных рекомендации в день;
- две активных цели в день, если цели не выполнены, рекомендаций от искусственного интеллекта не будет;
- создание цели;
- редактирование цели;

- просмотр страницы блокнота;
- создание заметок в блокноте за текущий день;
- редактирование заметок за текущий день;
- удаление заметок в блокноте за текущий день;
- просмотр истории заметок в период до тридцати дней;
- просмотр страницы со статистикой;
- просмотр расширенной статистики по времени, проведенному в приложениях;
- возможность ограничить доступ к шести приложениям;
- возможность запустить таймер фокусировки до двух часов;
- просмотр статистики за выбранный период, до тридцати дней.

Премиум пользователь должен иметь следующие возможности:

- продление премиум подписки;
- автопродление премиум подписки;
- выход из профиля;
- запуск таймера фокусировки более двух часов;
- установка разных системных звуков уведомлений;
- изменение цвета интерфейса;
- изменение цвета статистики;
- неограниченное количество рекомендаций по составлению целей от искусственного интеллекта;
- возможность ограничить доступ более чем шести приложениям;
- весь функционал предыдущих пользователей.

### **1.3.2 Нефункциональные требования**

Данное приложение должно соответствовать следующим нефункциональным требованиям:

- приложение должно быть совместимо с android 13 и выше;

- устройствами с диагональю экрана от 6.1 до 6.79 дюймов;
- время загрузки приложения не должно превышать трех секунд;
- все действия пользователя, а именно сохранение, редактирование и удаление целей должны выполняться за время не более одной секунды;
- приложение должно корректно обрабатывать ошибки и исключительные ситуации, например, потеря соединения с интернетом, предотвращая аварийное завершение работы;
- приложение должно поддерживать надежную аутентификацию пользователей через электронную почту и пароль;
- все данные пользователей должны шифроваться на стороне сервера и передаваться через защищенные каналы HTTPS;
- разрешение на отправку уведомлений;
- разрешение приложению быть всегда сверху;
- разрешение к статистике использования;
- доступ к функции «не беспокоить»;
- интерфейс должен быть понятным даже для новых пользователей без необходимости длительного обучения;
- приложение должно адаптироваться к различным размерам экранов в пределах указанных диапазонов;
- приложение должно оптимизировать использование батареи, избегая избыточного потребления ресурсов;
- функциональность искусственного интеллекта должна работать с меньшими задержками, при этом обеспечивая точность в девяносто процентов или выше для предлагаемых рекомендаций;
- приложение должно поддерживать возможность простого и быстрого обновления для добавления новых функций и исправления ошибок;
- для обеспечения функционирования приложения будут использоваться облачные серверы для серверной части системы,



оснащённые не менее чем 2 виртуальными ядрами, 2 ГБ оперативной памяти и SSD-накопителем объёмом 30 ГБ.

## 2 Анализ предметной области

### 2.1 Глоссарий

В данной курсовой работе используются следующие термины и сокращения с соответствующими определениями:

**Клиент-серверная архитектура** – это модель организации вычислительных систем, в которой задачи распределены между клиентами и серверами.

**REST API** – архитектурный стиль взаимодействия между клиентом и сервером через HTTP.

**JSON** – текстовый формат обмена данными, который используется для хранения данных и их передачи между различными системами и приложениями.

**Flutter SDK** – фреймворк, разработанный Google для создания кроссплатформенных мобильных приложений.

**Spring Boot** – фреймворк для создания приложений на языке Java, который упрощает работу с фреймворком Spring.

**JWT** – открытый стандарт (RFC 7519) для безопасной передачи информации между сторонами в формате JSON.

**Docker** – программная платформа для разработки, доставки и запуска контейнерных приложений

**CI/CD** – технология автоматизации тестирования и доставки новых модулей разрабатываемого проекта заинтересованным сторонам.

**Таймер фокусировки** – инструмент, позволяющий ограничить время на отвлекающие действия и повысить концентрацию.

**Ограничение приложений** – функция, позволяющая ограничить или заблокировать доступ к определённым приложениям.

**Шагомер** – функция для подсчёта количества шагов, пройденных пользователем.

**Премиум-подписка** — модель монетизации с расширенным функционалом.

**Искусственный интеллект** — технологии, которые учатся, анализируют и решают задачи очень быстро.

**Qwen / QwQ 32B** — модель искусственного интеллекта, использованная для генерации рекомендаций.

## 2.2 Обзор аналогов

Для создания эффективного приложения, для начала нужно провести анализ аналогов, который позволит выявить ключевые преимущества и недостатки, а также особенности приложений.

## 2.3 StayFree

Оценка пользователей 4,4 скачиваний 10 млн+

Сильные стороны:

- отсутствие рекламы;
- функция сна и концентрации;
- ограничение работы приложений;
- детализированная статистика использования за разные периоды времени.

Слабые стороны:

- блокировка не срабатывает после превышения лимита (сброс при перезапуске);
- непонятные настройки и ошибки при работе с приложениями без заданного лимита;
- избыточный набор разрешений;

— отсутствие умных функций на базе искусственного интеллекта и системы постановки целей.

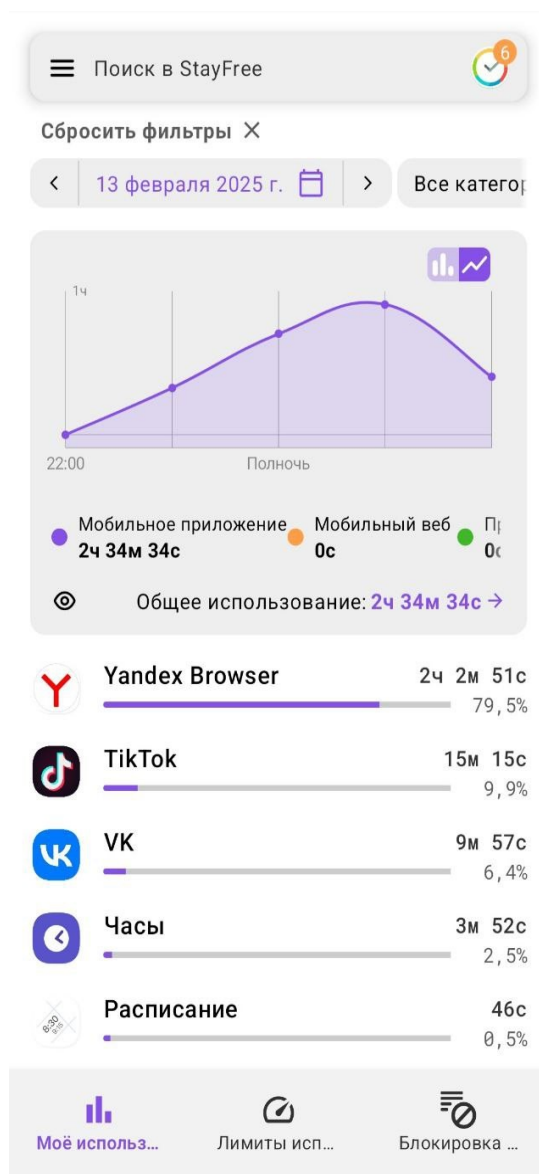


Рисунок 1 — Страница статистики приложения StayFree

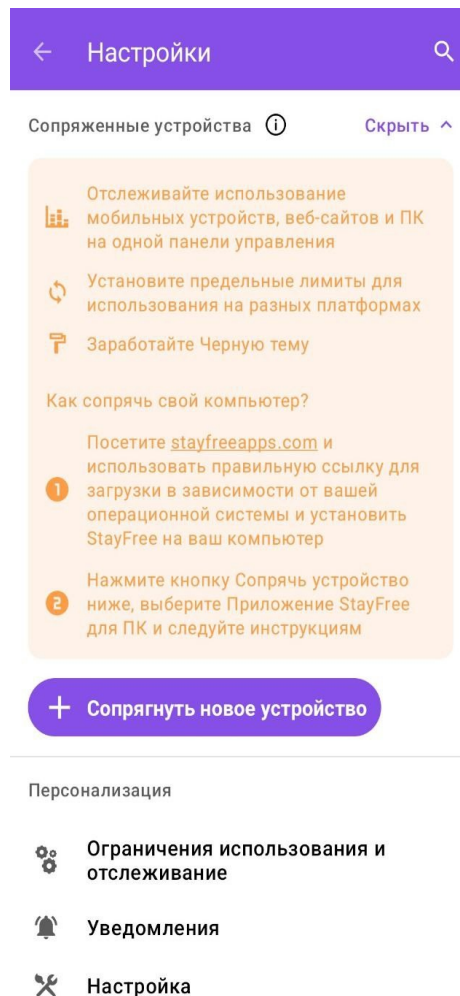


Рисунок 2 — Страница настроек приложения StayFree

## 2.4 ActionDash

Оценка пользователей 4,2 скачиваний 5 млн+

Сильные стороны:

- предоставляет статистику за неделю ещё до установки приложения;
- простой и понятный интерфейс;
- система достижений и возможность получения очков;
- ограничение времени использования приложений без рекламы и с поддержкой смены темы.

Слабые стороны:

- блокировка не активируется после истечения лимита;

— замедленная работа с подвисаниями и проблемы с выбором конкретных приложений для блокировки.

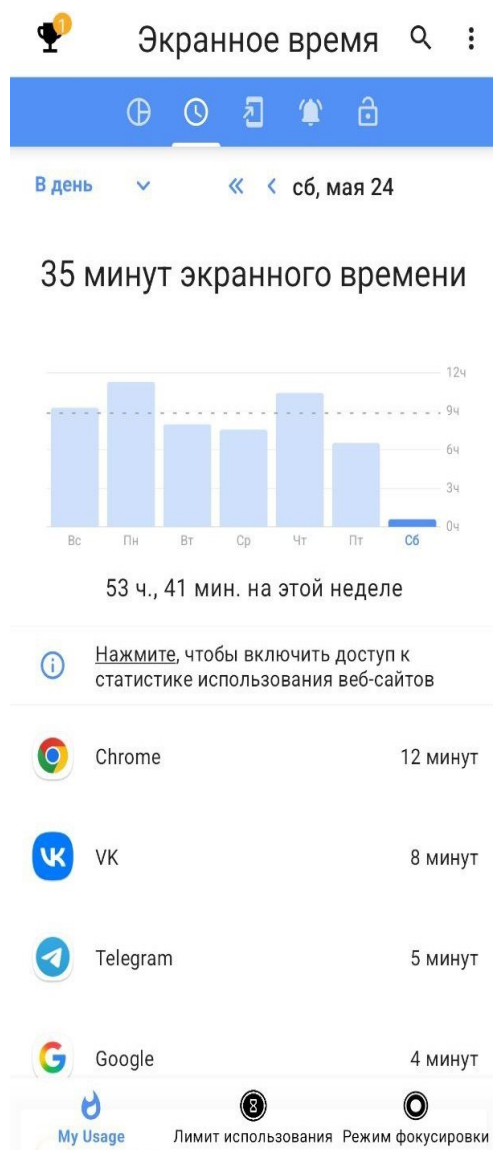


Рисунок 3 — Страница статистики приложения ActionDash

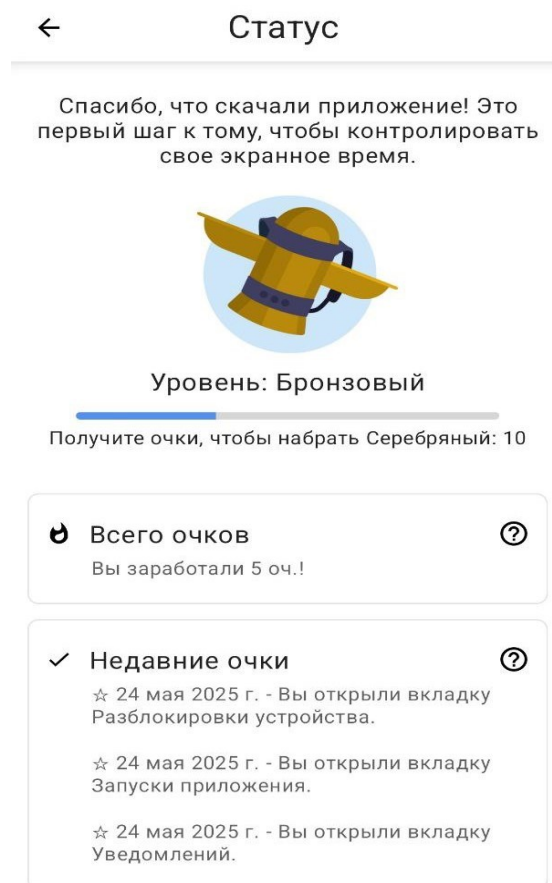


Рисунок 4 — Страница достижений приложения ActionDash

## 2.5 Экранное время + Блокировка

Оценка пользователей 4,2 скачиваний 1 млн+

Сильные стороны:

- ограничение времени в играх и ночное отключение телефона;
- удобный интерфейс с наличием базовой статистики и функций перерыва.

Слабые стороны:

- полностью платная модель, что ограничивает аудиторию;
- частые сбои в блокировке и подсчёте времени;
- отсутствие умных функций, списка целей и детализированной статистики по выполненным задачам.

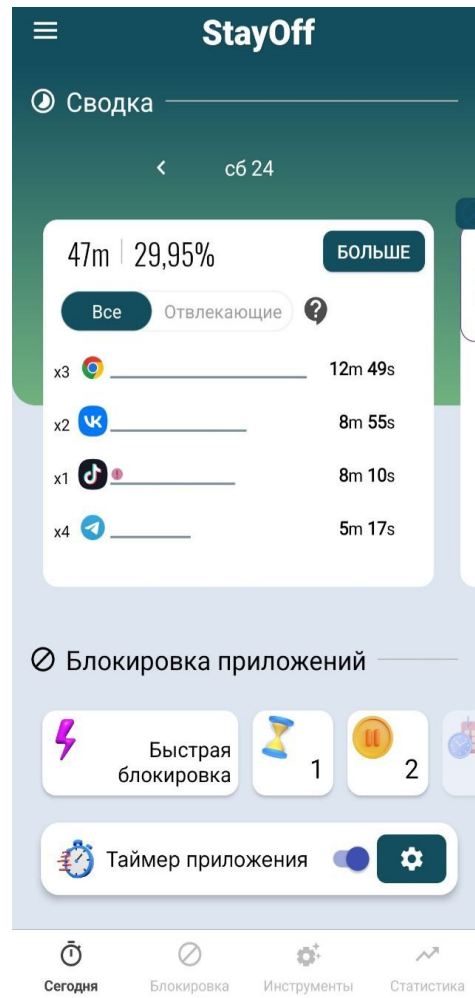


Рисунок 5 — Главная страница приложения Экранное время + блокировка



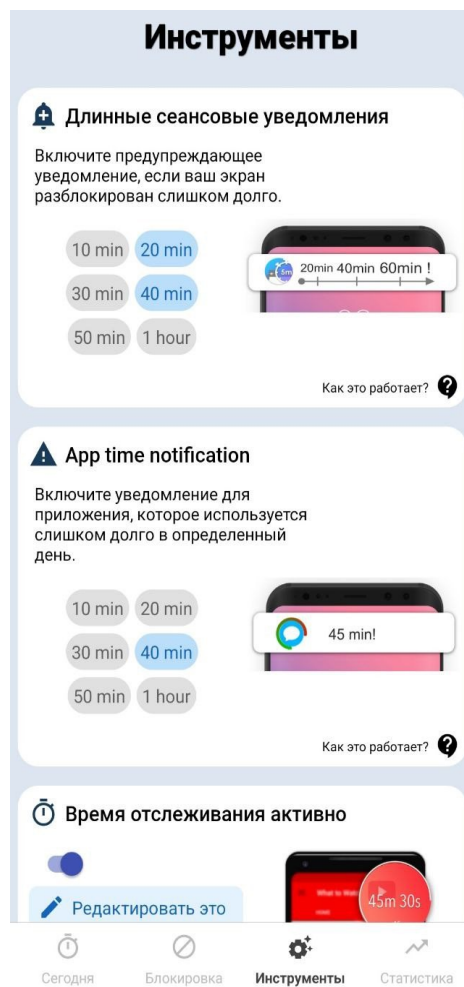


Рисунок 6 — Страница настроек приложения Экранное время + блокировка

## 2.6 Digitox

Оценка пользователей 4,6 скачиваний 1 млн+

Сильные стороны:

- категоризация приложений и подробный мониторинг использования;
- возможность установки лимитов, ежедневные и еженедельные отчёты.

Слабые стороны:

- отслеживание времени ограничено 10 днями;
- технические сбои в работе ограничений и трекинге;
- отсутствие умных функций на базе искусственного интеллекта, списка целей и статистики по выполненным задачам.

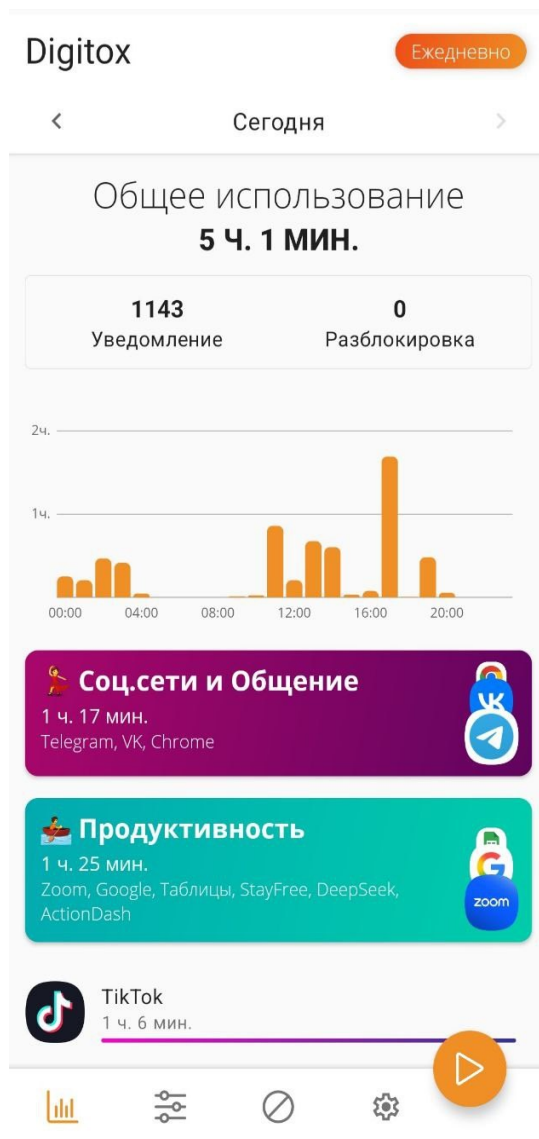





Рисунок 7 — Страница статистики приложения Digitox

## Настройки

Тема приложения



Widget

Показывать ежедневное использование

☐


Включить службу доступности

☐


Использование API доступности: Цель использования этой службы - наблюдать за действиями приложения и мгновенно показывать напоминания о чрезмерном использовании. Если вы не дадите это разрешение, вы не получите предупреждений о превышении лимита. Для этой службы данные не собираются.

Категории

PIN



Новый PIN-код

 Digitox

☐

Сторонние библиотеки

6.0.4








Рисунок 8 — Страница настроек приложения Digitox

## 3 Реализация

### 3.1 Средства реализации

Приложение разрабатывается на основе клиент-серверной архитектуры с использованием REST API для обмена данными между клиентом и сервером. Клиент запрашивает данные, а сервер их предоставляет. Запросы к API выполняются с использованием стандартных методов HTTP, а данные передаются в формате JSON.

Система имеет две части: Backend и Frontend. Backend отвечает за серверную часть приложения, она проводит обработку запросов от клиента и работу с базой данных, а frontend отвечает за внешний вид и пользовательский интерфейс, с которым взаимодействует пользователь.

Для реализации серверной части приложения будут использоваться следующие технологии:

- язык программирования Java: Java включает кроссплатформенность, безопасность, объектно-ориентированность и большое сообщество и поддержку;
- фреймворк Spring Boot: Spring Boot предоставляет ряд преимуществ, которые упрощают разработку приложений на языке Java. Автоматическая конфигурация, интеграция с другими инструментами, встроенные серверы, а также поддержка масштабируемости и безопасности;
- база данных MySQL: MySQL является надёжным, производительным, гибким и безопасным программным обеспечением, которое имеет обширное сообщество разработчиков и пользователей;
- инструмент для подготовки документации к API и проведения тестов API Swagger: Swagger является инструментом для создания, описания и тестирования API, основанный на спецификации OpenAPI. Он

позволяет программистам, техническим писателям и тестировщикам быстрее создавать, описывать и проверять программный интерфейс.

Для реализации клиентской части приложения будут использоваться следующие технологии:

— фреймворк Flutter SDK: главное преимущество Flutter это кроссплатформенность, нативные приложения для платформ разрабатываются на основе единой кодовой базы.

Для развертывания приложения будут использоваться следующие технологии:

— Docker: контейнеры Docker можно легко запускать в облачной инфраструктуре и на любом локальном устройстве.

Для рекомендаций по составлению списка целей будет использоваться API модели искусственного интеллекта Qwen:

— QwQ 32B (free): искусственный интеллект Qwen был выбран за свою универсальность, поддержку мультимодальных задач и открытый исходный код.

### **3.2 Реализация серверной части**

Серверная часть приложения разработана на языке Java с использованием фреймворка Spring Boot. Архитектура построена по принципам многослойного проектирования, где каждый слой отвечает за строго определённые задачи. Приложение предоставляет REST API, защищённый через JWT-аутентификацию. Для хранения данных используется реляционная СУБД MySQL. Развёртывание и автоматизация обновлений реализованы с помощью Docker и GitHub Actions (CI/CD).

### **3.3 Основные модели в базе данных**

На рисунке 9 показан слой сущностей.

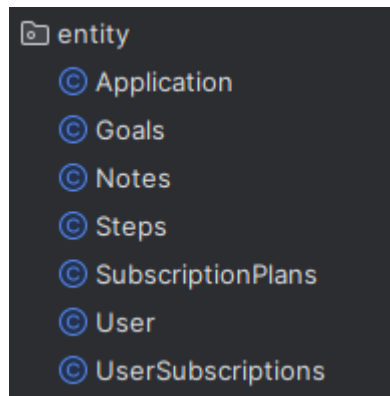


Рисунок 9 — Слой сущностей

- Сущность Application представляет собой статистическую запись об использовании мобильного приложения конкретным пользователем за определённую дату. Каждая запись хранит наименование приложения, его технический идентификатор (package name), дату использования и общее время активности в миллисекундах. Используется для формирования пользовательской аналитики и отчётности.
- Сущность Goals предназначена для хранения целей пользователя как созданных им самостоятельно, так и предложенных системой или куратором/тренером. Она фиксирует содержание цели, статус выполнения, источник создания и временные метки.
- Сущность Notes представляет собой заметку пользователя, содержащую заголовок, текст и дату/время, к которому она относится. Используется для хранения личных текстовых записей, напоминаний, идей и другой вспомогательной информации.
- Сущность Steps используется для хранения информации о количестве шагов, пройденных пользователем в конкретную дату.
- Сущность SubscriptionPlans описывает тарифные планы подписки, доступные пользователям в рамках платформы. Каждый план имеет уникальный код, название, срок действия, флаг премиальности и

стоимость. Эти данные используются для предоставления расширенного функционала по подписке.

- Сущность User представляет зарегистрированного пользователя платформы. Содержит личные данные, статусы подписки и временные метки.
- Сущность UserSubscriptions представляет собой факт оформления подписки пользователем, включая дату начала, окончания, сумму оплаты и параметры продления. Это связующая таблица между пользователем и тарифным планом, отражающая историю подписочной активности.

### 3.4 Сервисный слой

Сервисный слой проекта реализует основную бизнес-логику приложения. Именно на этом уровне происходит обработка пользовательских запросов, выполнение проверок, взаимодействие с репозиториями, а также формирование логики отклика. Сервисы изолируют контроллеры от работы с базой данных и позволяют повторно использовать логику в разных частях приложения. На рисунке 10 представлен сервисный слой.

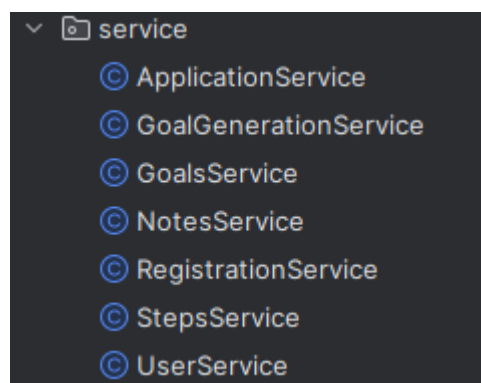


Рисунок 10 — Сервисный слой

В папке service реализованы следующие классы:

- `ApplicationService`: отвечает за обработку данных, связанных с использованием приложений. Предоставляет методы для сохранения информации о приложениях, времени использования, получения статистики по пользователям и дате.
- `GoalGenerationService`: реализует логику автоматической генерации целей с помощью Искусственного интеллекта.
- `GoalsService`: управляет пользовательскими целями. Содержит методы создания, обновления, удаления, фильтрации и изменения статусов целей (например, `ACTIVE`, `COMPLETED`). Также отвечает за учёт авторства цели — вручную созданная или сгенерированной искусственным интеллектом.
- `NotesService`: сервис для управления пользовательскими заметками. Обрабатывает добавление, редактирование, удаление и выборку заметок по времени, пользователю.
- `RegistrationService`: отвечает за регистрацию новых пользователей в системе. Обеспечивает проверку уникальности email, хэширование пароля, инициализацию пользовательского профиля и генерацию JWT-токена после успешной регистрации.
- `StepsService`: реализует логику подсчёта и хранения количества шагов пользователя. Предусмотрены функции добавления новых записей и получения статистики по дням.
- `UserService`: предоставляет методы для работы с данными пользователей: получение профиля по ID/email, редактирование, проверка наличия премиум-доступа, обновление подписки и взаимодействие с сервисом безопасности.



### 3.5 Слой безопасности

Слой security реализует систему аутентификации и авторизации в приложении на основе JWT-токенов. Он отвечает за защиту маршрутов, обработку токенов, фильтрацию запросов, управление сессиями и настройку публичных/защищённых эндпоинтов. Все компоненты интегрированы в систему с помощью Spring Security. На рисунке 11 представлен слой security.

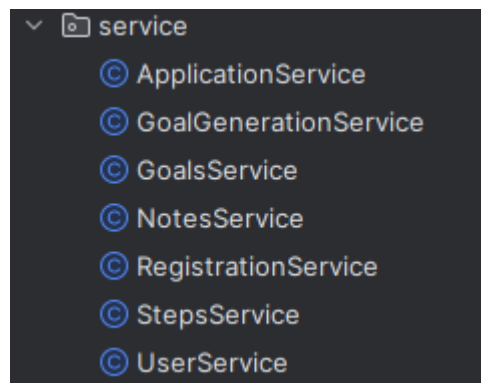


Рисунок 11 — Слой security

### 3.6 Контроллеры

Контроллеры реализуют уровень взаимодействия с клиентской частью приложения через REST API. Каждый контроллер обрабатывает входящие HTTP-запросы, валидирует данные, вызывает соответствующие методы сервисного слоя и возвращает результат в формате JSON. Все контроллеры размещены в пакете controller. На рисунке 12 показана структура controller.

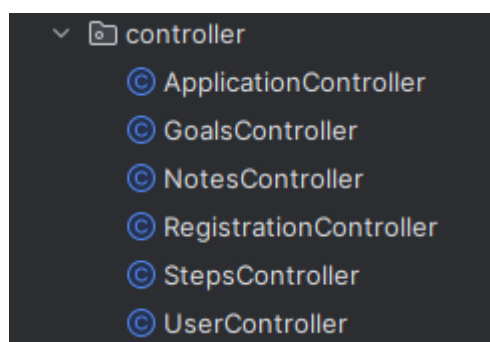


Рисунок 12 — Слой контроллеров

## **4 Развертывание серверной части**

Для удобства тестирования, масштабируемости и переноса проекта между средами, серверная часть приложения была контейнеризирована с использованием Docker. В качестве хостинг-платформы для развёртывания использован облачный сервер Timeweb Cloud.

### **4.1 Реализация клиентской части**

Клиентская часть приложения разработана на языке Dart с использованием фреймворка Flutter. Это кроссплатформенное решение обеспечивает поддержку Android-устройств и в перспективе – iOS. Интерфейс построен по принципу разделения представления, логики и навигации, что обеспечивает читаемость и масштабируемость кода. На рисунке 13 показана реализация компонентов клиентской части.

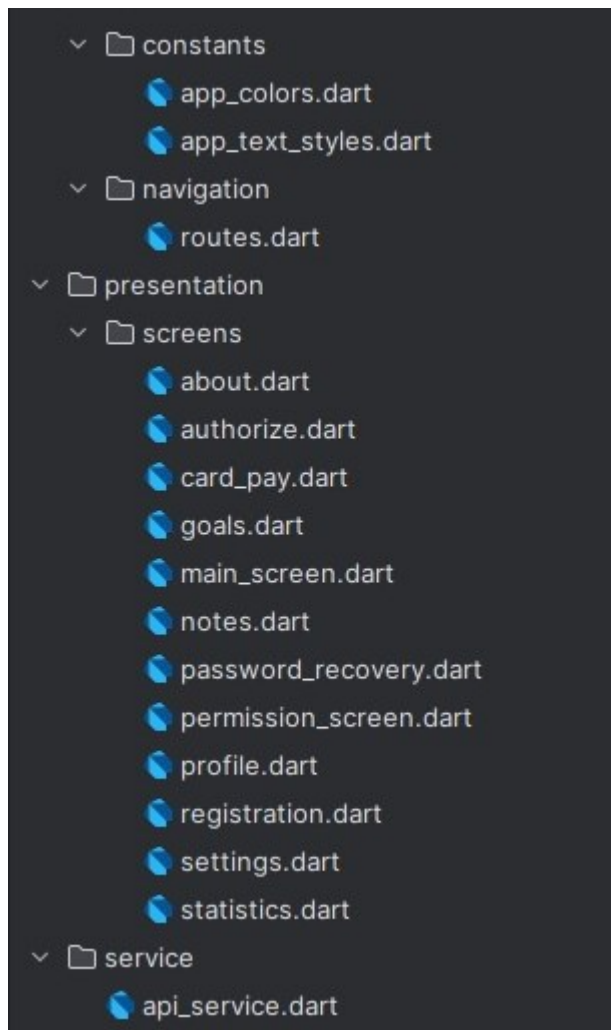


Рисунок 13 — Структура клиентской части

Папка constants содержит общие константы приложения:

- app\_colors.dart — определяет цветовую палитру и тему приложения;
- app\_text\_styles.dart — стили для текстов (размер, шрифт, жирность и т.д.), обеспечивают визуальное единообразие интерфейса.

Папка navigation:

- routes.dart — конфигурация маршрутов и навигации между экранами. Содержит определения маршрутов (например, /login, /main, /profile) и методы перехода между ними.

Папка screens Содержит все основные экраны пользовательского интерфейса:

- `about.dart` — информация о приложении;
- `authorize.dart` — экран авторизации (вход по email и паролю);
- `card_pay.dart` — экран выбора/оплаты подписки;
- `goals.dart` — цели пользователя (отображение, редактирование);
- `main_screen.dart` — главный экран после входа в систему;
- `notes.dart` — отображение и редактирование пользовательских заметок;
- `password_recovery.dart` — восстановление пароля;
- `permission_screen.dart` — управление разрешениями (например, доступ к шагам/приложениям);
- `profile.dart` — экран профиля пользователя;
- `registration.dart` — экран регистрации нового пользователя;
- `settings.dart` — настройки приложения;
- `statistics.dart` — статистика активности (шаги, приложения, цели).

Каждый экран реализует логику отображения интерфейса, обработки пользовательских действий и вызова API.

Папка `service`:

- `api_service.dart` — класс для отправки HTTP-запросов на backend. Содержит методы для авторизации, получения данных, отправки целей, заметок, шагов и т.д. Все взаимодействие с сервером централизовано через этот сервис.

## 4.2 Обмен данными с сервером

Взаимодействие клиента с серверной частью происходит через REST API. Используются стандартные методы:

- GET — для получения информации (цели, шаги, заметки);
- POST — для создания новых сущностей (регистрация, цель, заметка);
- PUT / PATCH — для обновления данных;

— DELETE — для удаления заметок, целей и т.д.

Аутентификация реализована через JWT-токены. После входа токен сохраняется и передаётся в заголовках всех последующих запросов. Обработка авторизации и сохранение токена реализованы в `api_service.dart` (см. рисунок 13).

## **4.3 Реализация интерфейса**

### **4.3.1 Экран статистики**

На данном экране (см. рисунок 14) изображена статистика, пользователю доступно отображение количества пройденных шагов и приложений, которые чаще всего используются. Над каждой из круговых диаграмм отображена кнопка, при нажатии на которую пользователь переходит на страницу с более подробной статистикой. В самом верху отображается календарь, при нажатии на одну из дат, страница отобразит статистику за выбранный день.

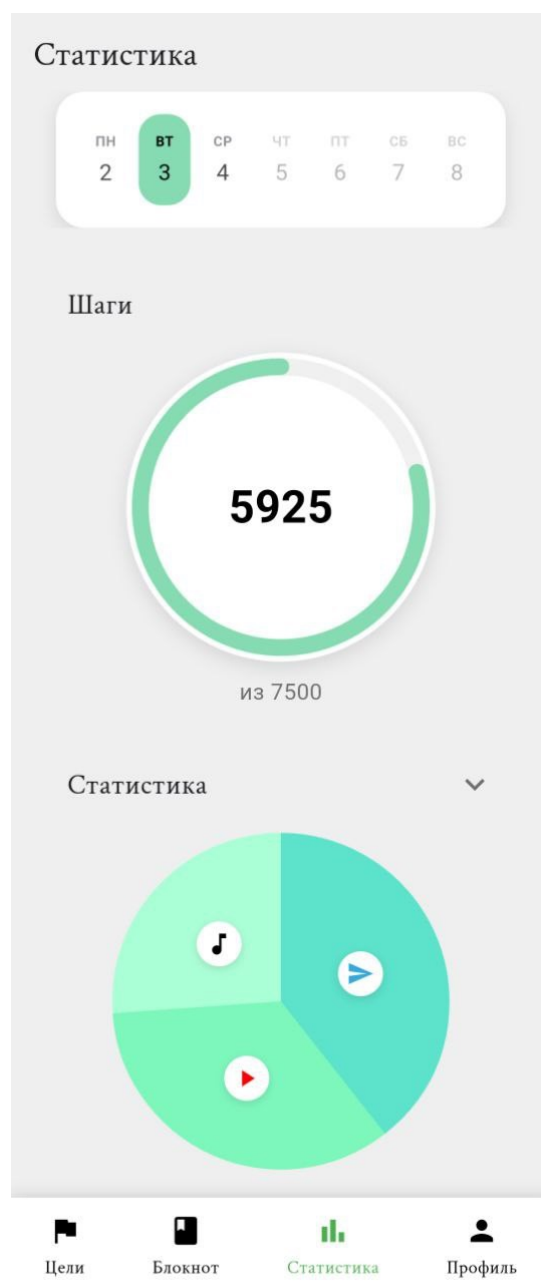


Рисунок 14 — Страница статистики

#### 4.3.2 Экран блокнота

На экране отображена форма с названием дня недели (см. рисунок 15), в которой ведутся заметки. В самом низу отображена кнопка, при нажатии на которую можно добавлять новые записи в блокнот.

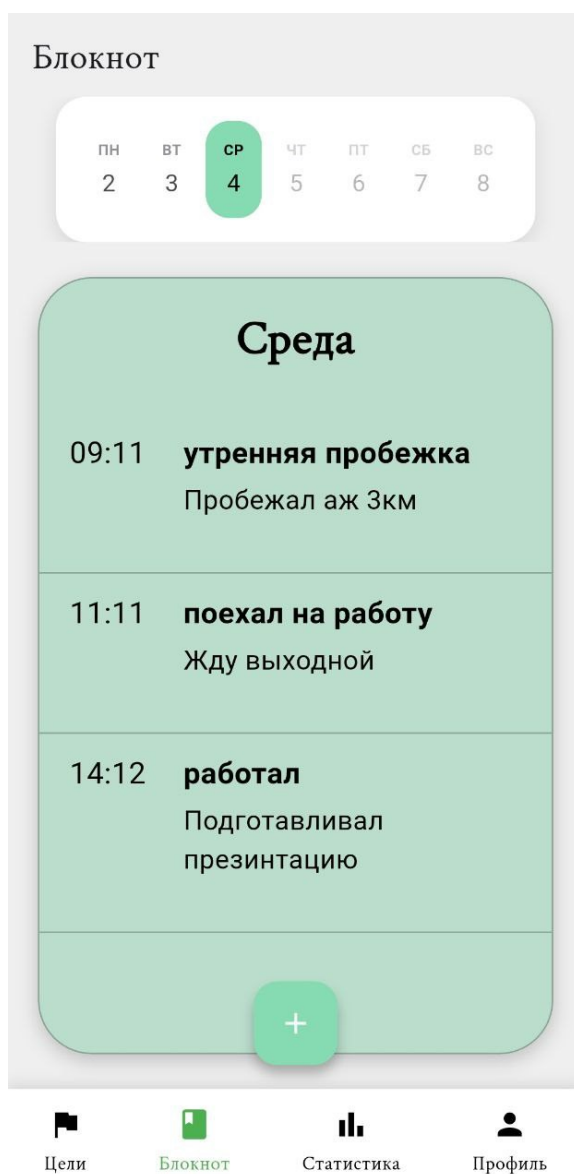


Рисунок 15 — Страница блокнота

### 4.3.3 Экран списка целей

На экране отображается список поставленных целей (см. рисунок 16), выполненные цели отмечаются галочкой, под списком отображается кнопка при нажатии на нее пользователь может поставить новую цель.

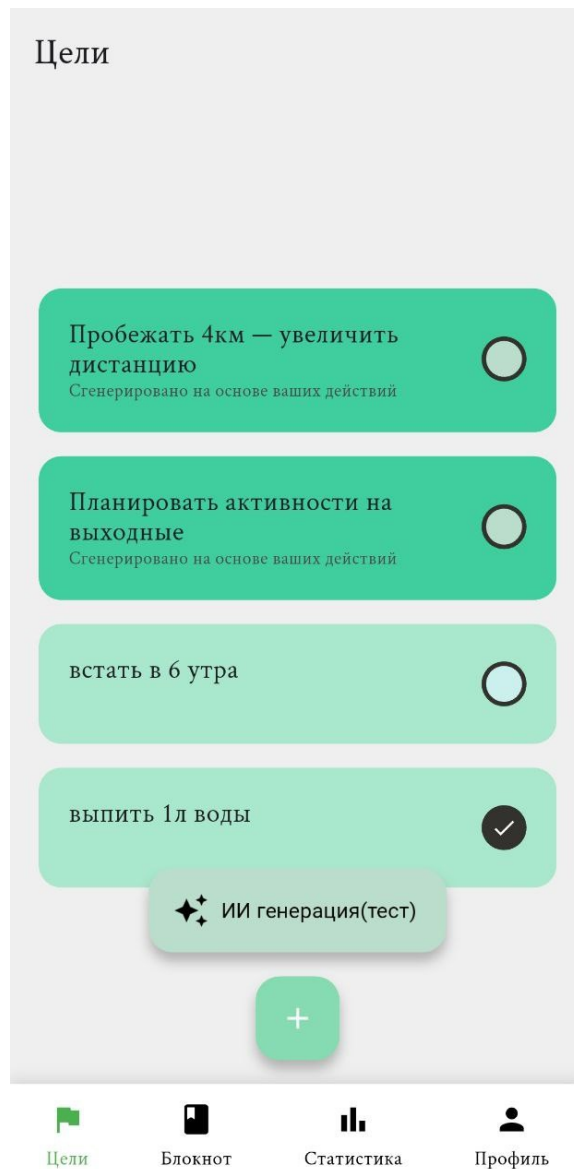


Рисунок 16 — Страница списка целей



## 5 Перспективы развития

В перспективах развития проекта имеются следующие необходимые улучшения:

- запуск приложения в App Store для расширения аудитории – начало разработки запланировано на 31.08.2025;
- возможность добавлять друзей в приложении и делиться с ними целями – начало разработки запланировано на 01.08.2025 и 08.08.2025;
- внедрение реферальной программы – начало разработки запланировано на 03.09.2025.

А чтобы мотивировать действующих клиентов советовать приложение своим друзьям и близким, им за это будут полагаться бонусы. Эти изменения повысят вовлеченность пользователей и укрепят позиции проекта на рынке.

## ЗАКЛЮЧЕНИЕ

В рамках выполнения курсовой работы была достигнута поставленная цель — разработано мобильное приложение «YourDay», предназначенное для анализа активности пользователя в течение дня и предоставления персонализированных рекомендаций по повышению продуктивности.

В процессе разработки были применены одни из современных технологий к ним относятся язык программирования Java и фреймворк Spring Boot для создания надёжного серверного приложения, база данных MySQL для хранения информации, а также фреймворк Flutter для создания кроссплатформенного клиентского интерфейса. Особое внимание было уделено вопросам безопасности, производительности и удобства пользовательского взаимодействия.

Пользователь в приложении имеет возможность отслеживать время, проведённое в приложениях, количество шагов, ставить цели и получать рекомендации от искусственного интеллекта. Приложение реализовано в виде клиент-серверной архитектуры с применением REST API.

В дальнейшем планируется расширить функциональность приложения, повысить точность работы интеллектуальной системы и адаптировать продукт под платформу iOS. Таким образом, проект «YourDay» обладает высоким потенциалом для развития.

## СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Flutter. Документация по Flutter SDK [Электронный ресурс]. – Режим доступа: URL: <https://docs.flutter.dev> – заглавие с экрана (Дата обращения 01.04.2025)
2. Spring Boot. Документация по Spring Boot [Электронный ресурс]. – Режим доступа: URL: <https://docs.spring.io/spring-boot> – заглавие с экрана (Дата обращения 20.03.2025)
3. MySQL. Официальная документация [Электронный ресурс]. – Режим доступа: URL: <https://dev.mysql.com/doc> – заглавие с экрана (Дата обращения 24.03.2025)
4. Docker. Документация по Docker [Электронный ресурс]. – Режим доступа: URL: <https://docs.docker.com> – заглавие с экрана (Дата обращения 15.05.2025)
5. JWT. JSON Web Token (JWT) Introduction [Электронный ресурс]. – Режим доступа: URL: <https://jwt.io/introduction> – заглавие с экрана (Дата обращения 05.04.2025)
6. Swagger/OpenAPI. Документация Swagger [Электронный ресурс]. – Режим доступа: URL: <https://swagger.io/docs/> – заглавие с экрана (Дата обращения 15.05.2025)
7. Qwen/QwQ 32B. GitHub репозиторий и документация [Электронный ресурс]. – Режим доступа: URL: <https://github.com/QwenLM> – заглавие с экрана (Дата обращения 27.04.2025)

## ПРИЛОЖЕНИЕ А

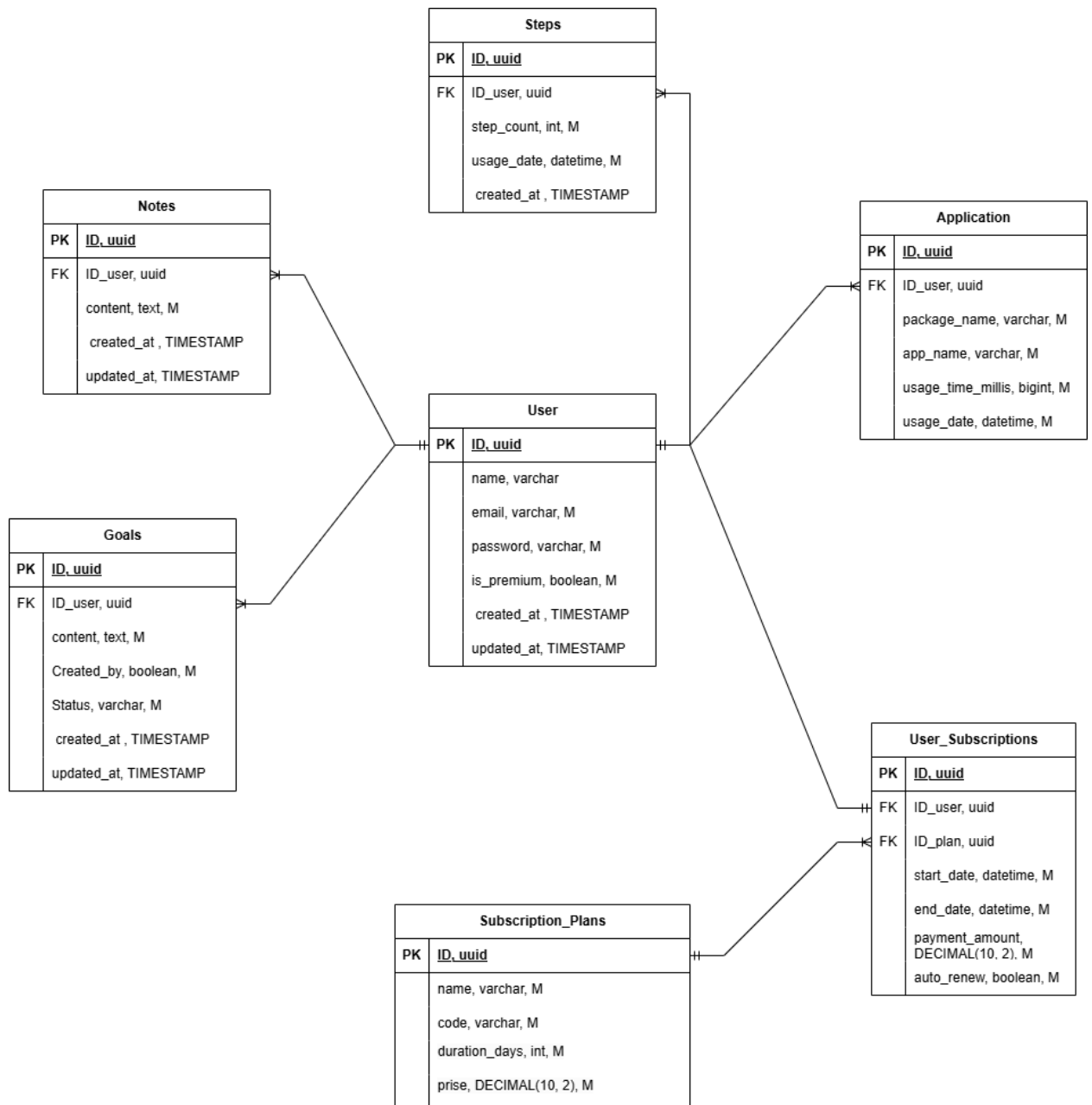


Рисунок А.1 — ER-диаграмма

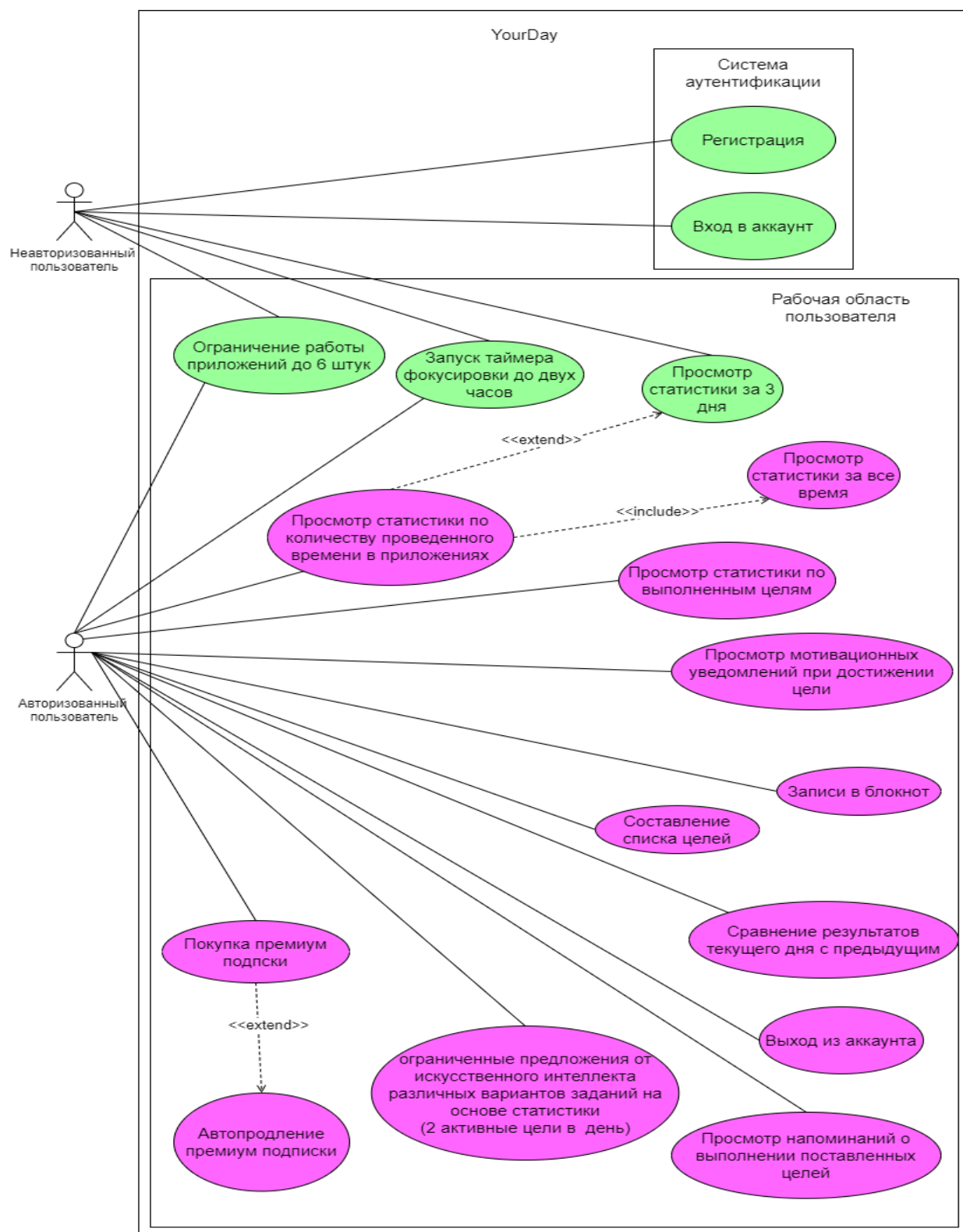


Рисунок А.2 — Диаграмма прецедентов неавторизованный и авторизованный пользователь



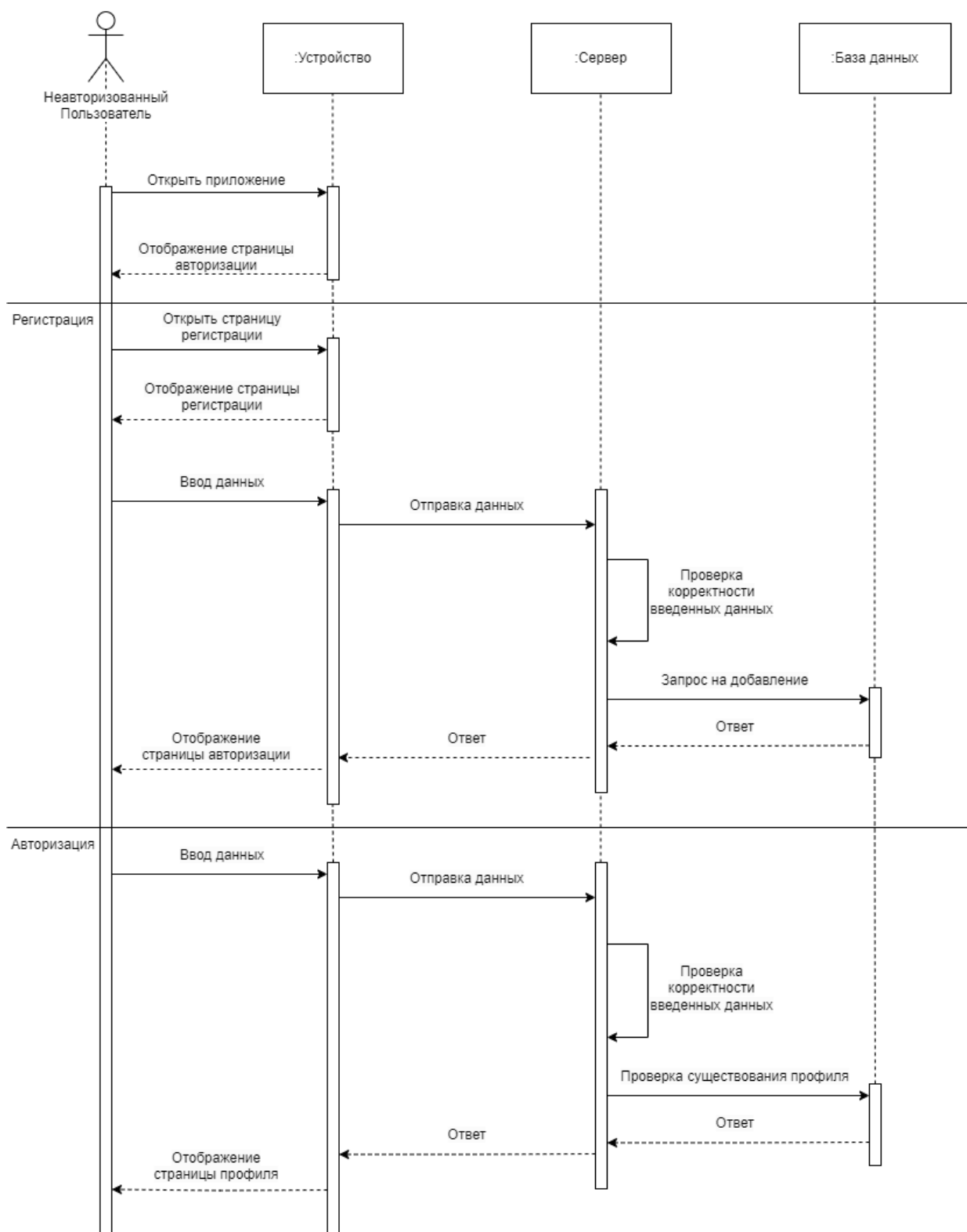


Рисунок А.4 — Диаграмма последовательности неавторизованный пользователь

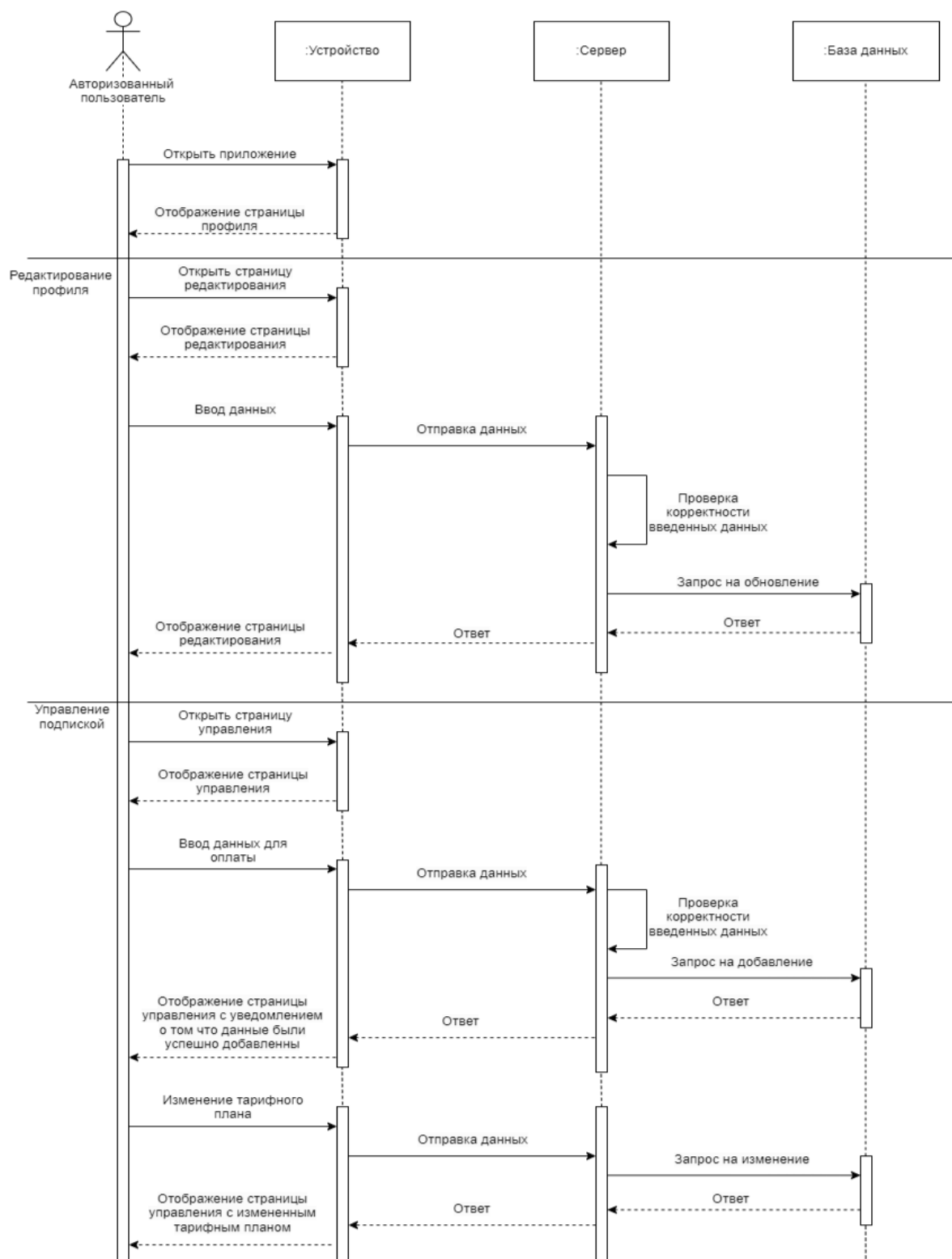


Рисунок А.5 — Диаграмма последовательности авторизованный  
ПОЛЬЗОВАТЕЛЬ



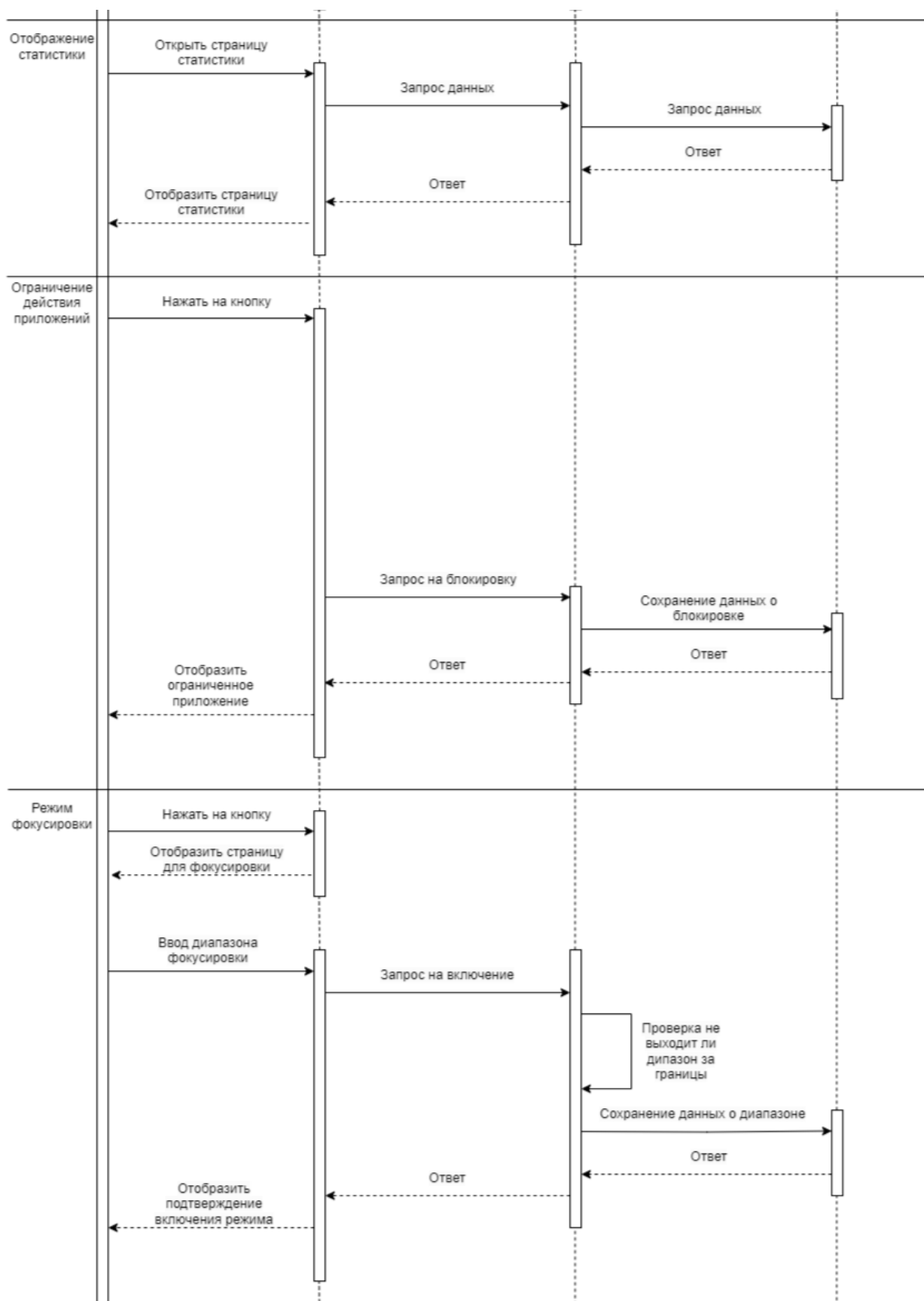


Рисунок А.6 — Продолжение диаграммы последовательности  
авторизованный пользователь

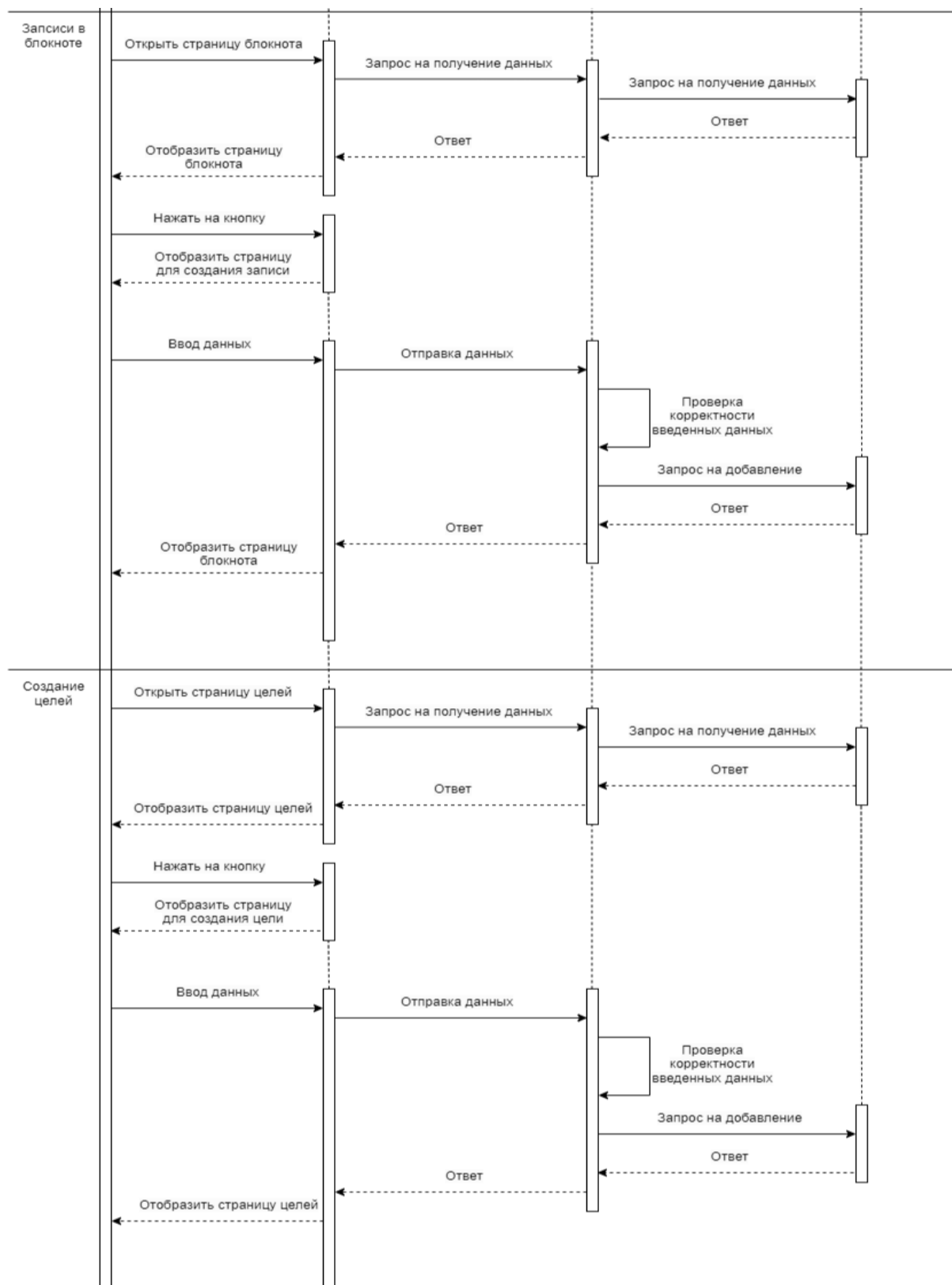


Рисунок А.7 — Продолжение диаграммы последовательности  
авторизованный пользователь

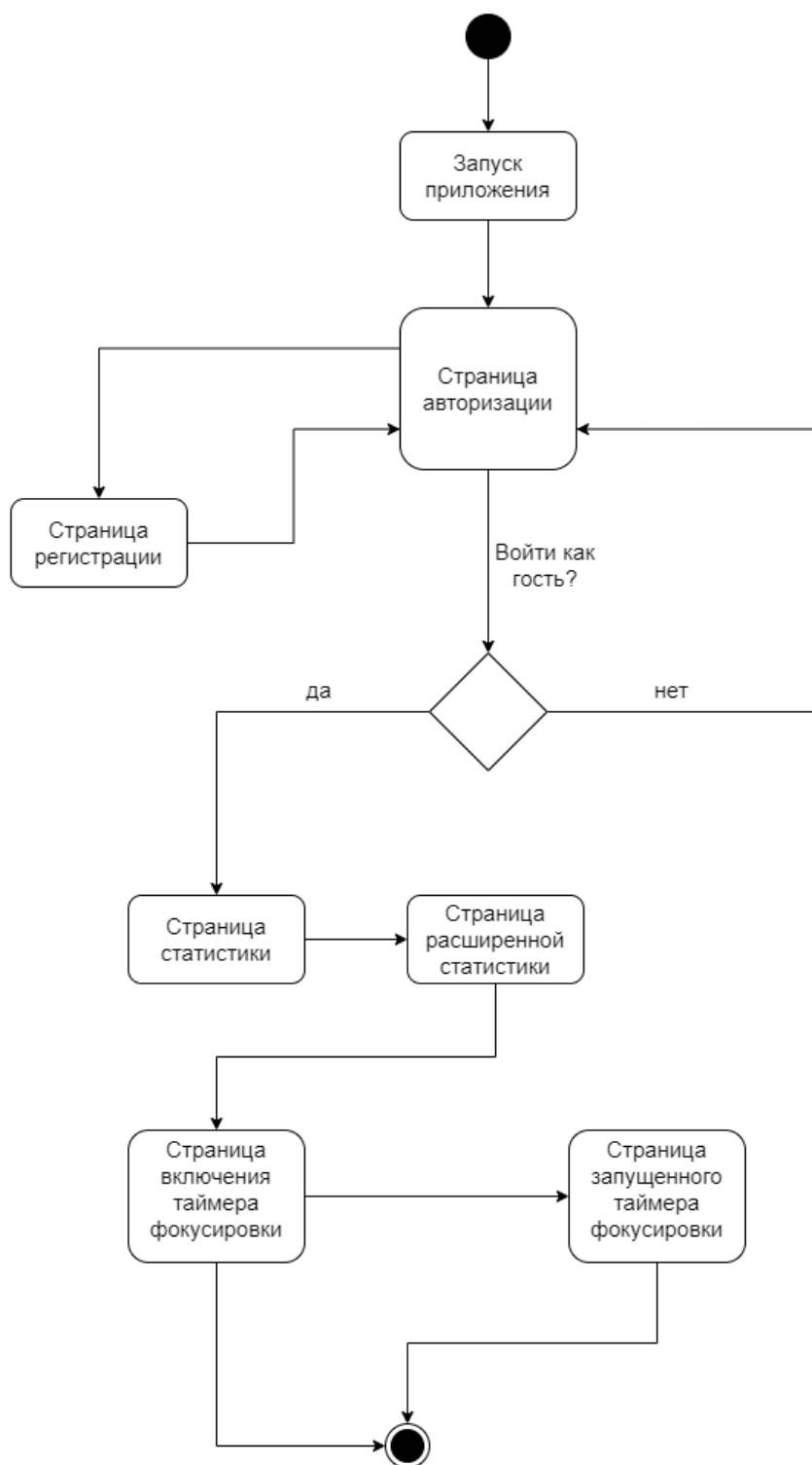


Рисунок А.8 — Диаграмма состояний неавторизованный пользователь

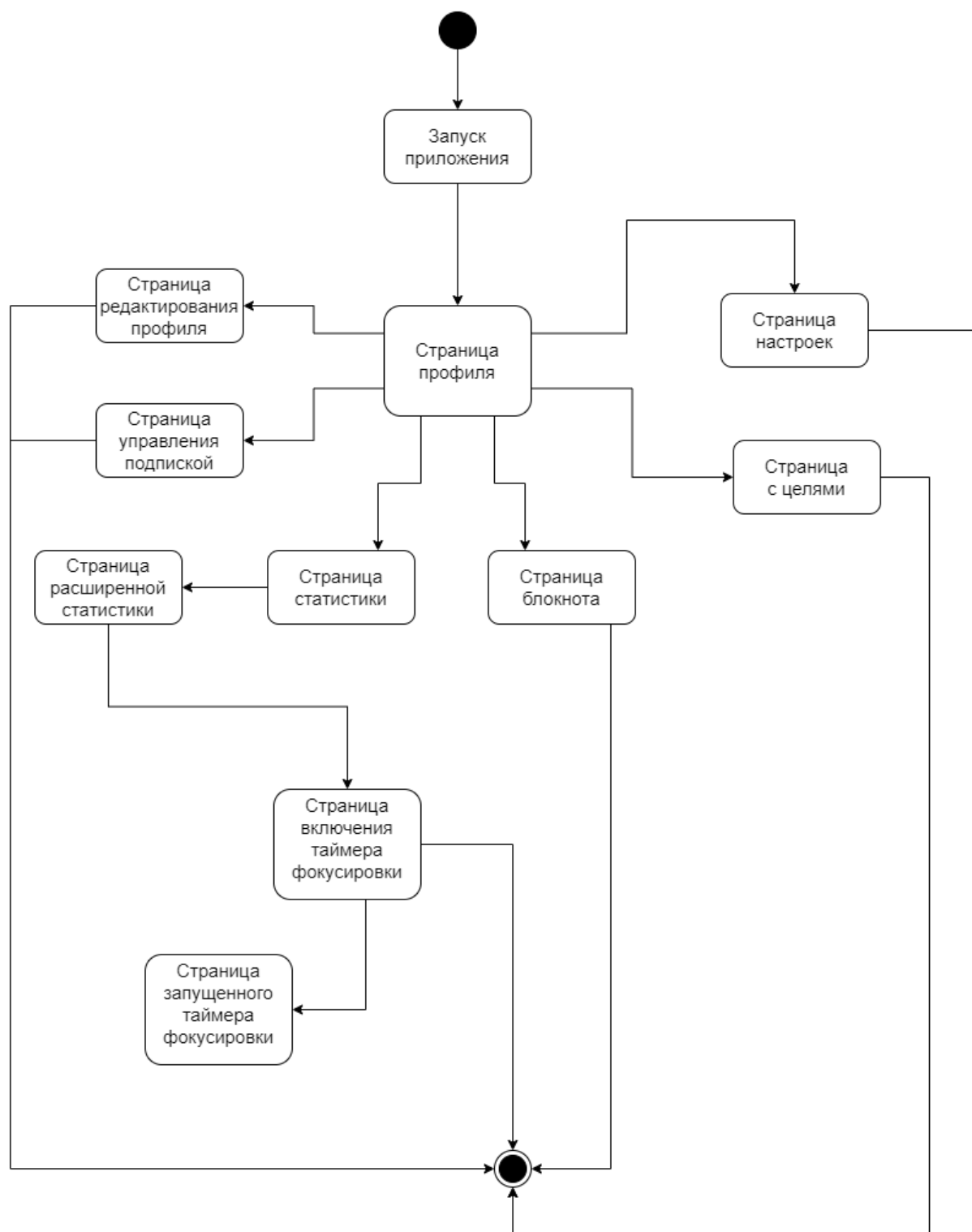


Рисунок А.9 — Диаграмма состояний авторизованный пользователь



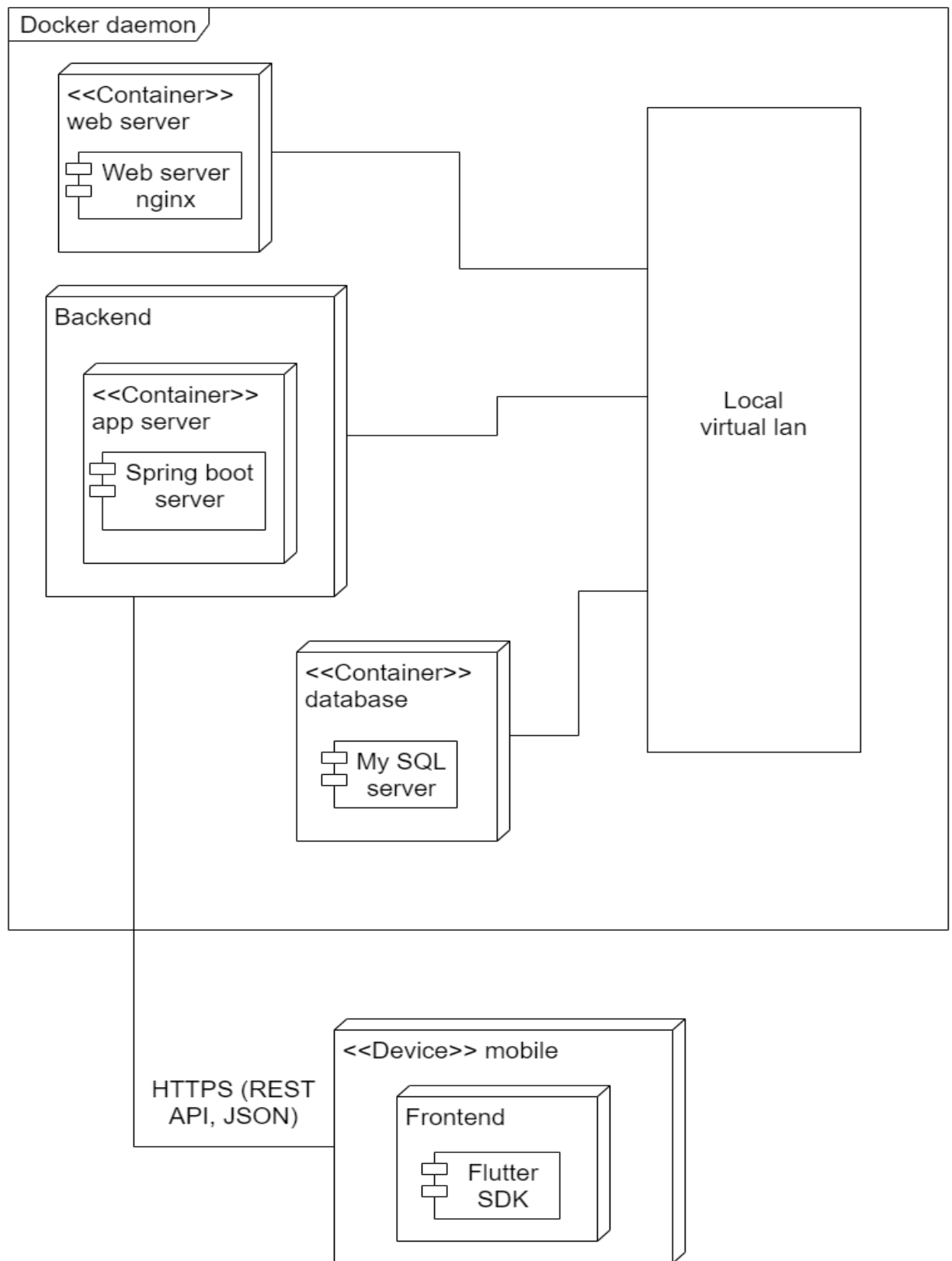


Рисунок А.11 — Диаграмма развертывания