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 прослушивания подкастов «TikTalk»

Курсовой проект

по дисциплине

Технологии программирования

09.03.04 Программная инженерия

Информационные системы и сетевые технологии

Преподаватель _____ В.С. Тарасов, ст. преподаватель __.__.20__

Обучающийся _____ К.П. Луговской, 3 курс, д/о

Обучающийся _____ П.Н. Негуляев, 3 курс, д/о

Обучающийся _____ А.А. Сазонов, 3 курс, д/о

Обучающийся _____ М.Т. Сошич, 3 курс, д/о

Руководитель _____ Е.Д. Проскуряков, ассистент

Воронеж 2024

Содержание

Введение

Последние несколько лет рынок мобильных приложений в России стремительно развивается. Исследование потребительского поведения в данной сфере проводилось в августе 2021 года в формате панельного онлайн-опроса. Всего было опрошено 300 человек в возрасте от 18 до 55 лет из разных городов России с населением от 500 тысяч жителей. В темах опроса также фигурировало прослушивание подкастов на мобильном приложении[1].

В числе преимуществ такого формата участники опроса отметили удобство доступа к разнообразному контенту в любое время и в любом месте. Возможность слушать подкасты во время занятий спортом, поездок или выполнения домашних дел стала особенно важной для пользователей, что положительно оценили 45% участников опроса.

Другая причина выбора мобильных приложений для подкастов среди опрошенных людей — экономия времени. Нет необходимости скачивать отдельные файлы или искать новые эпизоды вручную. Благодаря подпискам и автоматическим обновлениям новые выпуски подкастов становятся доступными сразу после их выхода. Это отметили 52% респондентов.

Участники опроса также отметили возможность простого поиска и сравнения подкастов на разных платформах. Для этого достаточно использовать несколько приложений и подписок, чтобы выбрать самые интересные и полезные подкасты. Удобство такого подхода отметило 68% опрошенных.

Целью данной работы является разработка мобильного приложения TikTalk для прослушивания подкастов. Особенностью данного приложения будет наличие функции общей ленты подкастов, ленты подписок и возможностью создания альбомов. Также будет внедрена система для быстрой загрузки и легкой записи подкастов для авторов контента.

1 Постановка задачи

1.1 Цели создания приложения

Целями создания мобильного приложения для IOS «TikTalk» являются:

— создание приложения для поиска, прослушивания, записи и загрузки подкастов в приложении для людей с повышенной занятостью;

— получение прибыли путем интеграции рекламы;

— увеличение количества пользователей, которые активно используют приложение.

1.2 Задачи приложения

Разрабатываемый проект должен решать следующие поставленные задачи:

— давать возможность подписываться на любимых авторов и формировать собственную ленту;

— искать подкасты и пользователей;

— просматривать подкасты в главной ленте приложения;

— записывать и загружать собственные подкасты;

— обрезать подкасты с помощью встроенного инструмента;

— редактировать данные своего аккаунта после авторизации в системе;

— подписываться на интересующих пользователей, оценивать и сохранять подкасты.

1.3 Требования к разрабатываемой системе

1.3.1 Функциональные требования

В мобильном приложении пользователь будет иметь одну из ролей:

— неавторизованный пользователь(гость);

— авторизированный пользователь.

Также для работы с веб-страницей модерации и ответа на жалобы будет введена роль администратора.

К разрабатываемому приложению выдвигаются следующие функциональные требования:

Авторизированный пользователь обладает следующими возможностями:

— просмотр главной ленты приложения;

— просмотр собственного профиля;

— редактирование собственного профиля, изменение персональных данных;

— загрузка подкастов из файлов устройства или запись подкаста в приложении;

— публикация подкастов в открытый доступ;

— осуществление поиска в специализированной вкладке приложения;

— возможность поставить отметку «нравится»;

— просмотр списка подкастов, на котором у пользователя стоит отметка «нравится»;

— подписки на профили интересующих авторов;

— просмотр списка профилей, на которых совершена подписка;

— снятие метки «нравится»;

— снятие подписки;

— поиск в специализированной вкладке приложения;

— подача жалобы на подкаст.

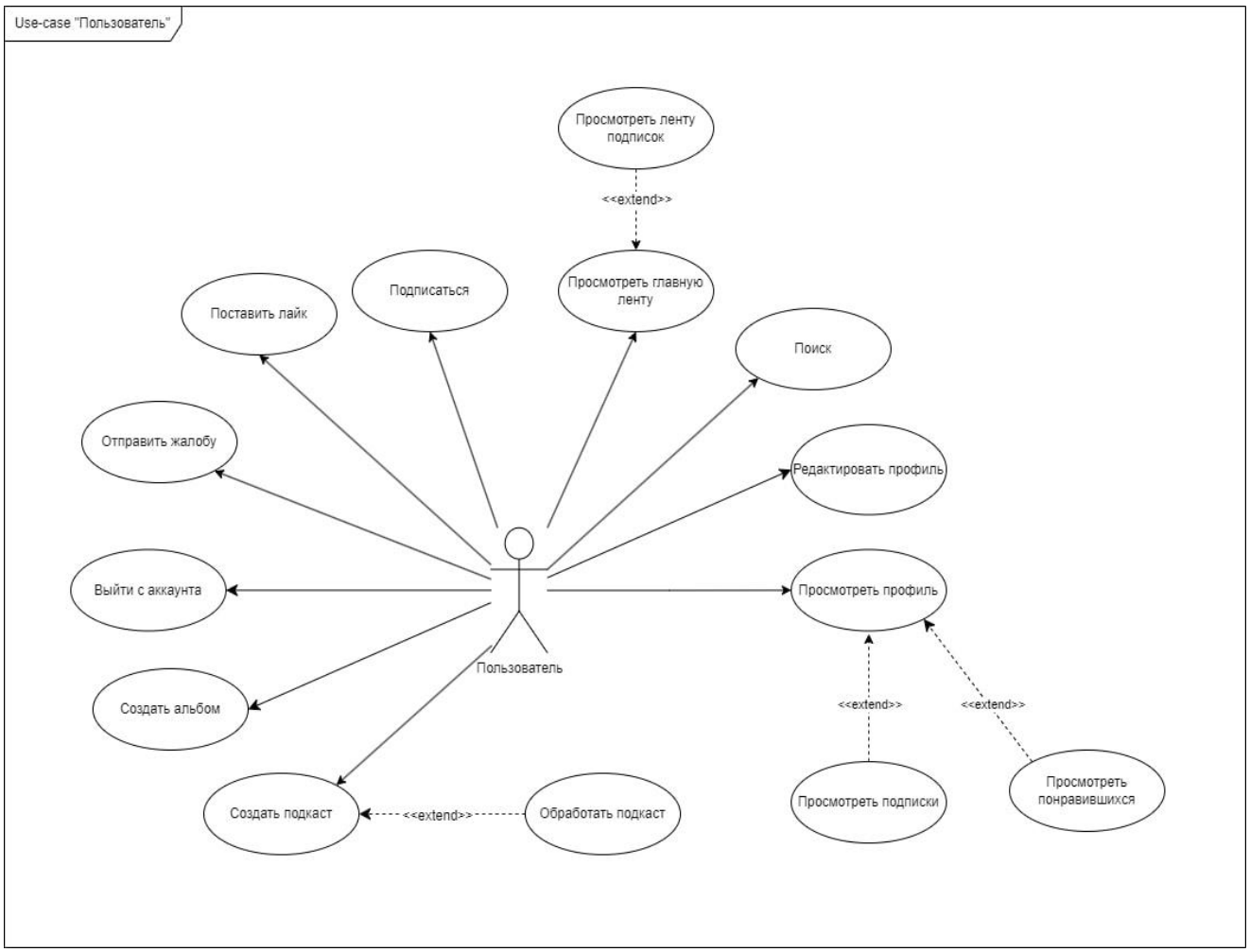

Рисунок 1 — Use-case пользователя

Неавторизированный пользователь (гость) обладает следующими возможностями:

— просмотр главной ленты приложения;

— авторизация;

— поиск в специализированной вкладке приложения;

— регистрация.

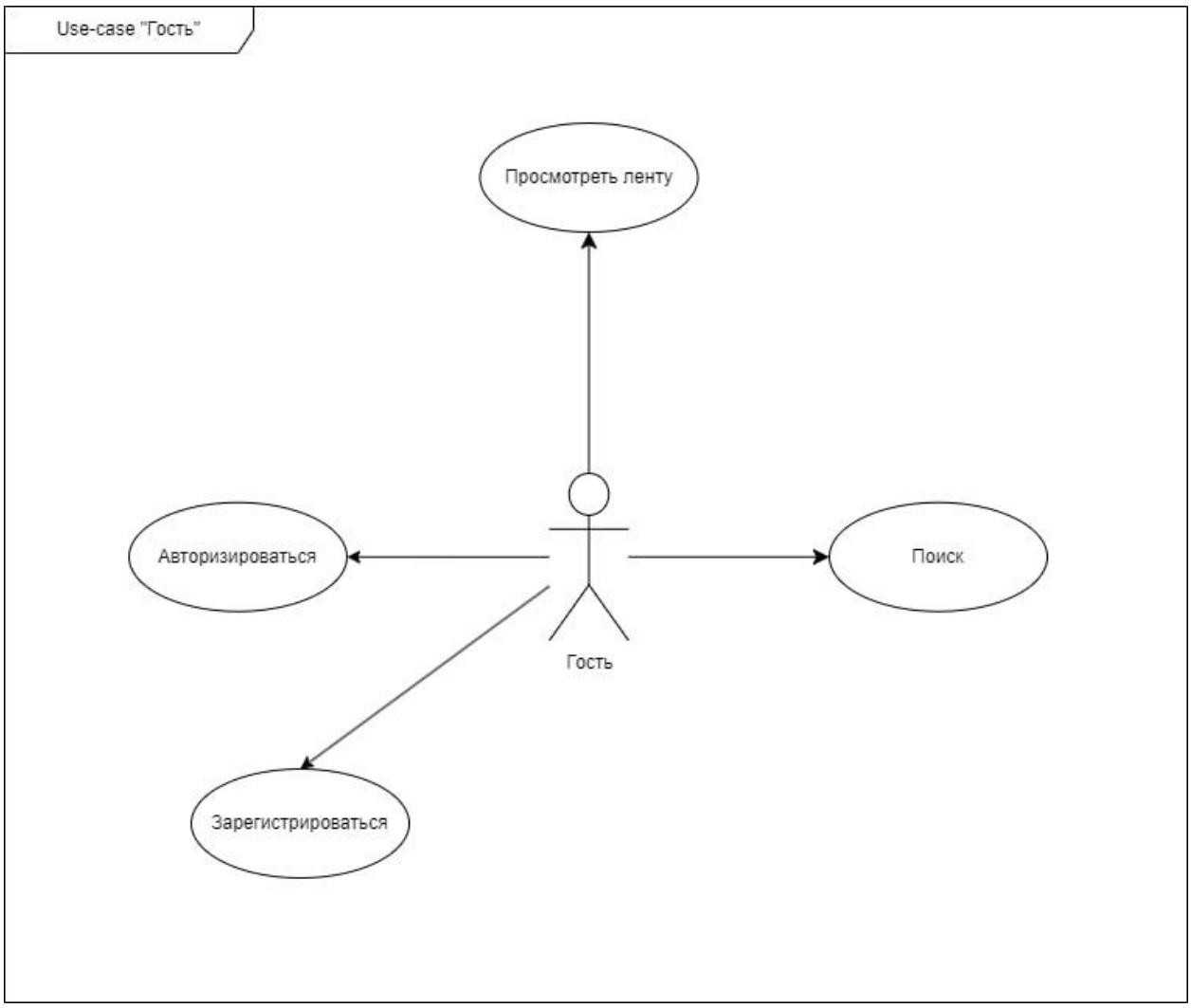

Рисунок 2 — Use-case гостя

Администратор обладает следующими возможностями:

— вход в веб-версию панели модерации;

— удаление загруженного пользователями контента, на который пришли жалобы.

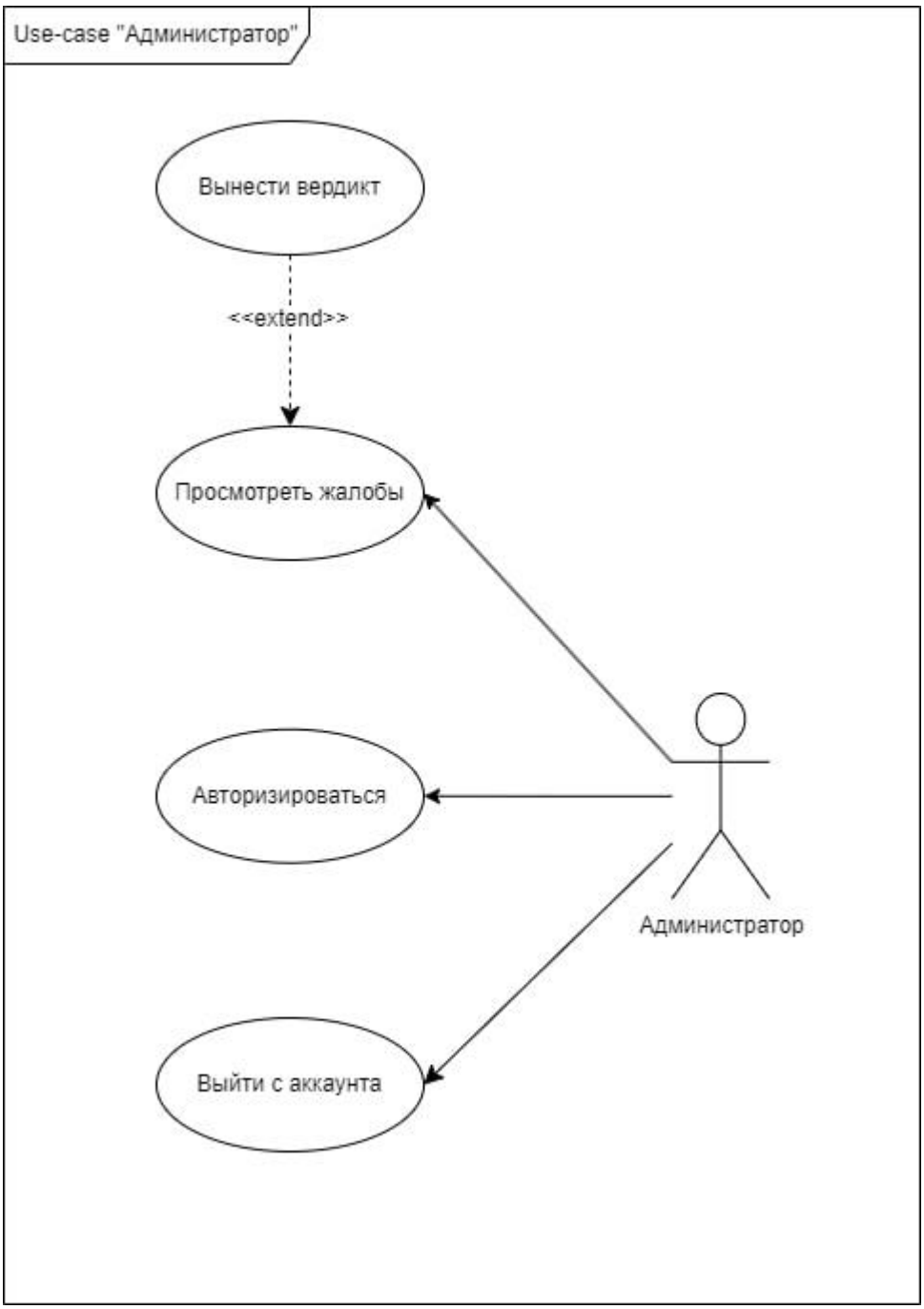

Рисунок 3 — Use-case администратора

1.3.2 Требования к интерфейсу

Оформление и верстка экранов приложения должны соответствовать следующим требованиям:

— все экраны приложения должны быть оформлены в едином стиле;

— все экраны приложения должны быть оформлены в соответствии с принципами «Human Interface Guidelines».

1.3.3 Требования к безопасности

Система должна соответствовать триаде CIA:

— конфиденциальность;

— целостность;

— доступность.

Конфиденциальность означает, что только авторизованные лица могут просматривать конфиденциальную информацию. Данные, отправляемые по сети, не должны быть доступны посторонним лицам. Основной способ избежать этого - использовать методы шифрования для защиты данных.

Целостность означает сохранение исходного состояния данных, информации или системы. В контексте безопасности информации целостность гарантирует, что данные остаются нетронутыми и неизменными, не подвергаются несанкционированным модификациям, и остаются точными и достоверными в течение всего времени их хранения и передачи.

В контексте триады CIA доступность относится к гарантированию доступности информации и ресурсов системы для авторизованных пользователей в любое время, когда они этого требуют. Это означает, что информация должна быть доступна в нужном пользователю месте и время без каких-либо препятствий или задержек.

Обмен данных между клиентом и сервером должен осуществлять по протоколу HTTPS.

1.4 Задачи, решаемые в процессе разработки

Были поставлены следующие задачи:

— анализ предметной области;

— анализ аналогов;

— выбор языка программирования для написания back-enda и front-enda;

— создание доски заданий и репозитория на GitHub;

— создание макетов дизайна приложения;

— создание диаграмм: прецендентов, активностей, последовательностей, развертывания, сотрудничества, состояний, ER, развертывания, компонентов, классов;
— написание технического задания в соответствии с ГОСТ 34.201-20;
— разработка основных функций приложения;
— разработка особенностей приложения, которые бы выделяли его на рынке среди аналогов;
— реализация основных функций приложения с использованием REST API;
— реализация интерфейса приложения;
— написание курсовой работы по проекту.

2 Анализ предметной области

2.1 Терминология

Подкаст – вид аудио- или видеоконтента, когда один или несколько ведущих обсуждают разные темы.

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.

База данных – упорядоченный набор структурированной информации или данных, которые обычно хранятся в электронном виде в компьютерной системе.

Docker – это платформа для разработки, доставки и запуска приложений в контейнерах. Контейнеры Docker позволяют упаковывать приложения и все их зависимости в единый стандартизированный блок, который может быть легко перемещен между различными окружениями без изменения кода.

Авторизация – процесс проверки прав доступа пользователя к определенным ресурсам, функциям или данным в информационной системе.

Анонимный пользователь (неавторизированный) – обобщенное понимание пользователя, незарегистрированного в системе, под которым можно получить ограниченный доступ к отдельным объектам системы.

Авторизированный пользователь – посетитель приложения, который ранее проходил процесс регистрации и на данный момент зашел под своей учетной записью.

Администратор – специалист, который отвечает за поддержку работоспособности приложения, а также управляет модерацией контента.

Модерация – контроль выполнения требований, установленных владельцем приложения.

GitHub – это веб-сервис для хостинга проектов на базе системы контроля версий Git. Он предоставляет возможность разработчикам работать

с кодом, совместно его редактировать, отслеживать изменения и управлять версиями проектов.

Метрики – показатель для оценки работы маркетинга, продаж и бизнеса в целом.

2.2 Актуальность

На текущий момент мобильные приложения для прослушивания подкастов приобретают все большую актуальность благодаря нескольким ключевым факторам. Во-первых, популярность аудиоконтента растет, так как люди стремятся эффективно использовать свое время, например, слушая подкасты во время поездок, занятий спортом или выполнения домашних дел. Аудиоформат предоставляет удобный способ получения информации и развлечения, не отвлекая от повседневных задач.

Во-вторых, с развитием технологий смартфоны стали основным устройством для потребления контента, включая подкасты. Увеличение числа создателей контента и разнообразие тем подкастов также способствует росту интереса к этому формату. Люди могут легко найти подкасты, соответствующие их интересам, будь то новости, образовательные программы, интервью или развлекательные шоу.

В условиях интенсивного образа жизни и постоянной нехватки времени подкасты становятся важным источником знаний, новостей и вдохновения. Все эти факторы поддерживают высокий спрос на специализированные приложения для прослушивания подкастов, делая их актуальными и востребованными в настоящее время.

2.3 Обзор аналогов

2.3.1 PocketCast

PocketCast – это приложение для прослушивания подкастов, которое позволяет пользователям легко находить, загружать и слушать подкасты на своих мобильных устройствах. Оно обеспечивает пользовательский интерфейс, который облегчает поиск и сортировку подкастов по категориям, а также настройку уведомлений о новых выпусках. Приложение также

поддерживает функцию загрузки подкастов для прослушивания в офлайн-режиме. Кроме того, PocketCast предоставляет возможность автоматического удаления загруженных подкастов для освобождения места на устройстве. Однако отсутствует фильтр для настройки автоматического удаления подкастов, что не всегда удобно. PocketCast также не предлагает быструю загрузку и запись подкастов для людей с ограниченным временем. Из основных недостатков приложения можем выделить:

— ограниченные возможности модерации контента, такие как отсутствие инструментов для фильтрации или блокировки подкастов с нежелательным или оскорбительным содержанием, что может привести к негативному пользовательскому опыту;
— сложный интерфейс, например, неочевидные пути для выполнения определенных действий или запутанное расположение функциональных элементов, что может вызывать затруднения при использовании приложения.

Рисунок 4 — Интерфейс PocketCast

2.3.2 Яндекс.Музыка

Яндекс.Музыка – это приложение для прослушивания музыки и подкастов, которое предлагает пользователям доступ к огромной библиотеке аудиоконтента. С помощью этого приложения пользователи могут находить и слушать музыку и подкасты на своих мобильных устройствах.

Одной из функций приложения является возможность прослушивания подкастов. Пользователи могут находить подкасты по различным категориям, таким как новости, спорт, технологии, развлечения и многое другое. Приложение также предлагает функцию автоматического обновления

подкастов, что позволяет пользователям всегда быть в курсе последних выпусков.

Также стоит отметить, что приложение позволяет пользователям создавать свои собственные плейлисты и делиться ими с друзьями. Это делает Яндекс.Музыку отличным инструментом для поиска и прослушивания подкастов.

Но, функция подкастов в Яндекс.Музыке не является основной, так что в связи с этим возникают следующие проблемы:

— ограниченный выбор подкастов: хотя приложение предлагает широкий выбор подкастов, некоторые пользователи могут столкнуться с ограниченным выбором в определенных категориях;
— реклама: в бесплатной версии приложения пользователи могут столкнуться с рекламой, которая может быть навязчивой и отвлекающей;
— ограничения в бесплатной версии: бесплатная версия приложения имеет некоторые ограничения, такие как ограничение на количество загрузок и прослушивание в офлайн-режиме.

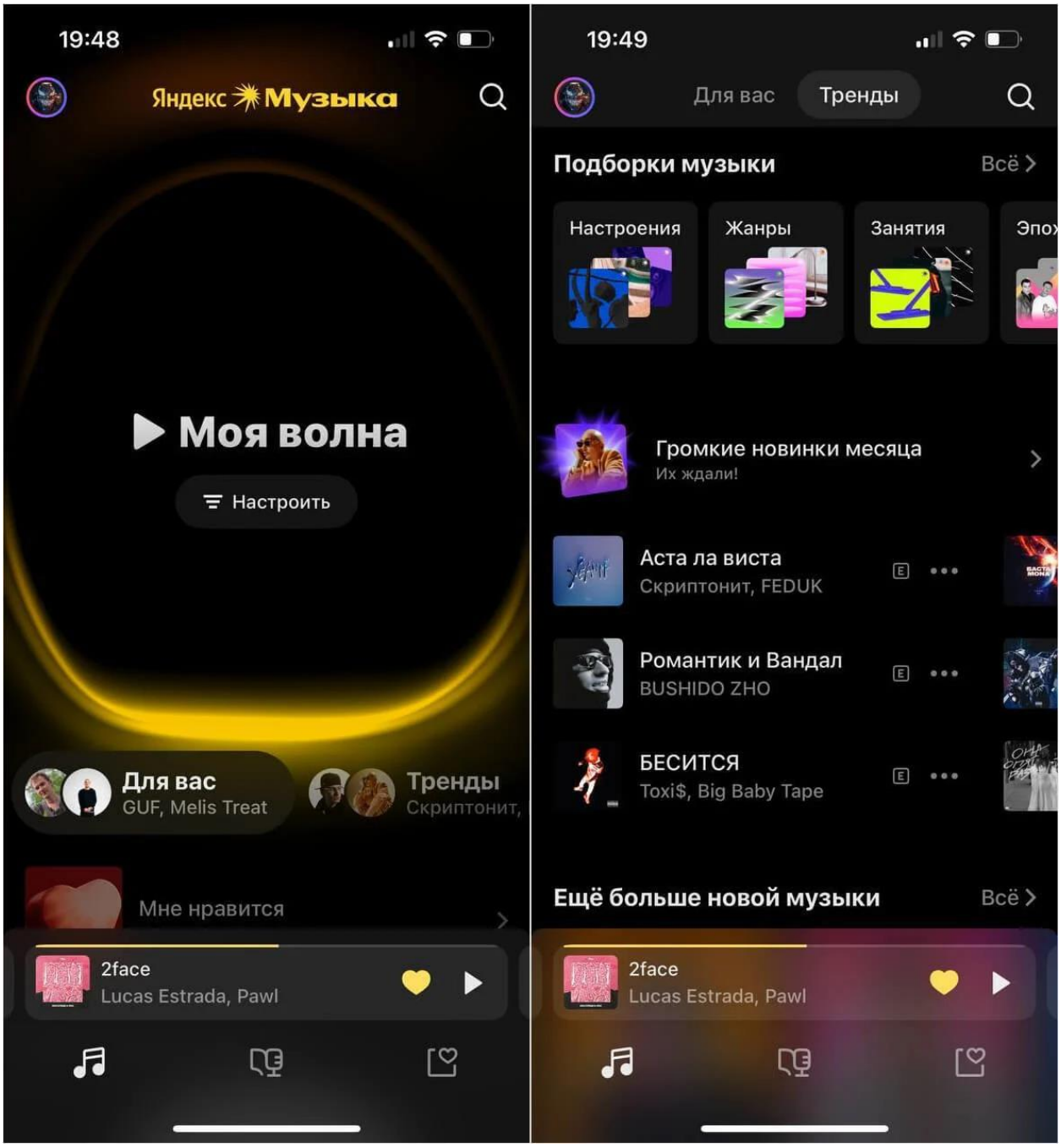

Рисунок 5 — Интерфейс Яндекс.Музыки

2.3.3 YouTube.Shorts

YouTube.Shorts – это новое приложение от YouTube, которое позволяет пользователям создавать и просматривать короткие видеоролики. Приложение предлагает широкий выбор контента, включая развлекательные, образовательные и информационные видео. Пользователи могут записывать и редактировать свои видео, добавлять музыку и эффекты, а также делиться ими с другими пользователями. YouTube.Shorts также предлагает функцию

рекомендаций, которая помогает пользователям находить контент, который может их заинтересовать. Но, YouTube.Shorts является приложением для просмотра коротких видео, а не прослушиванию подкастов, так же он является лишь дополнительным сервисом основного приложения, поэтому пользователи, приходящие за конкретным типом контента, предпочтут наше приложение, в отличии от YouTube.Shorts. Поэтому мы возьмем во внимание удобство загрузки контента в приложении, а также сосредоточимся на адаптации для прослушивания подкастов короткого формата. К недостаткам приложения можно отнести:

— отсутствие специализированных инструментов для создания и редактирования аудио-контента, что делает приложение менее привлекательным для пользователей, желающих создавать и распространять аудио-контент, такой как подкасты;
— недостаточное внимание к функциональности, связанной с прослушиванием подкастов, например, отсутствие функций управления подписками, настройки скорости воспроизведения или возможности скачивать эпизоды для прослушивания в офлайн-режиме, что делает YouTube.Shorts менее конкурентоспособным среди пользователей, ищущих приложение специально для прослушивания подкастов.

Рисунок 6 — Интерфейс YouTube.Shorts

3 Реализация

3.1 Средства реализации

Для реализации серверной части будут использоваться следующие средства:

— язык программирования Java 21 версии;

— фреймворк Spring Boot 3;

— СУБД PostgreSQL 16;

— Keycloak 23;

— MinIO.

Для реализации клиентской части мобильного приложения будут использоваться следующие средства:

— язык программирования Swift 5;

— фреймворк UIKit.

Для реализации веб-приложения для модерации будут использоваться следующие средства:

— язык программирования JavaScript;

— фреймворк Vue.js 3.4.

Для развертывания приложения будет использоваться Docker.

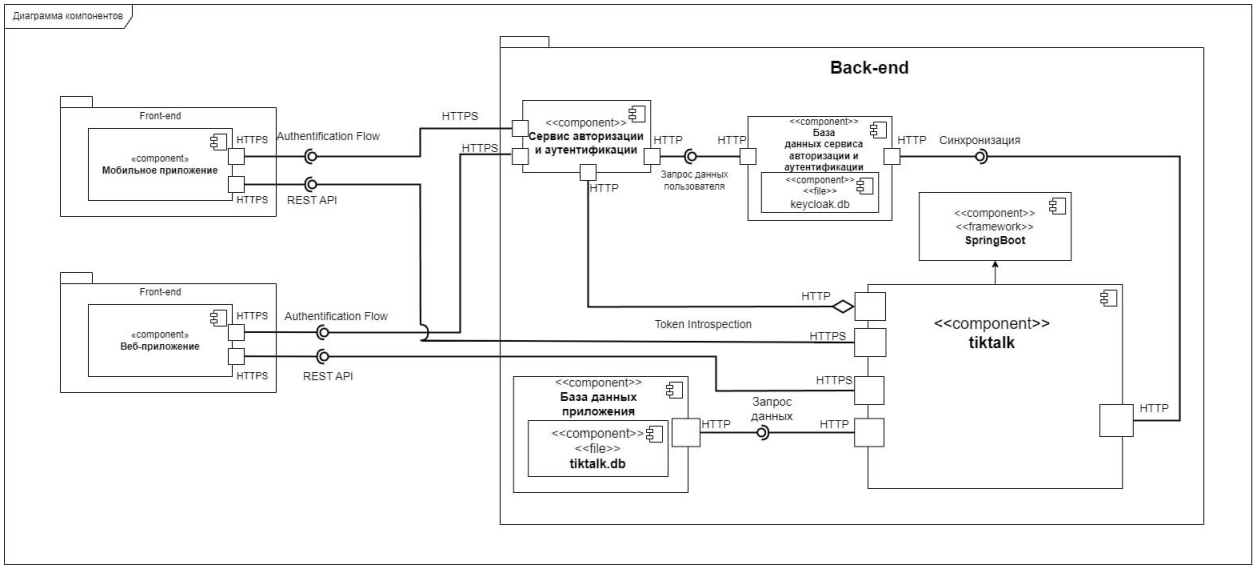

Рисунок 7 — Диаграмма компонентов

3.1.1 Java

Для разработки серверной части приложения применялись язык программирования Java 21 и фреймворк Spring Boot 3[2].

Java характеризуется простым синтаксисом, богатой стандартной библиотекой для работы с различными структурами данных и сетью, а также удобными инструментами сборки, такими как Maven, которые позволяют быстро подключать нужные зависимости.

Spring Boot 3 упрощает создание веб-приложений благодаря мощной системе аннотаций на языке Java.

3.1.2 PostgreSQL

Для хранения данных используется реляционная база данных PostgreSQL 16. За счёт производительности в долгосрочной перспективе позволит значительно увеличить клиентскую базу без необходимости смены технологии.

3.1.3 Keycloak

Для аутентификации пользователей используется Keycloak 23. В будущем этот сервис обеспечит нам возможность лёгкой реализации входа через сторонние сервисы.

3.1.4 Swift

Для реализации клиентской части мобильного приложения используется язык Swift 5. Это современный язык программирования, разработанный Apple для создания приложений на платформах iOS, macOS, watchOS и tvOS. Swift сочетает в себе простоту и производительность, что делает его отличным выбором для разработки приложений.

UIKit – это фреймворк, предоставляющий инструменты и инфраструктуру для разработки графических интерфейсов на iOS и tvOS.

3.1.5 JavaScript

Для реализации веб-приложения для администраторов был выбран язык JavaScript и фреймворк Vue.js.

JavaScript – это высокоуровневый, интерпретируемый язык программирования, который является одним из основных компонентов веб-технологий наряду с HTML и CSS. Он был разработан для добавления интерактивности и динамического поведения веб-страницам.

Vue.js – это прогрессивный фреймворк для создания пользовательских интерфейсов, разработанный для упрощения разработки веб-приложений. Он акцентирует внимание на представлении (view) и может интегрироваться в проекты постепенно, по мере необходимости.

3.1.6 Docker

Для развёртывания приложения использовался Docker.

Docker представляет собой инструмент контейнеризации приложений, обеспечивающий их переносимость и ускоряющий процесс разработки. Благодаря изоляции процессов, приложения не могут воздействовать на внешнюю среду, что повышает уровень безопасности.

Docker Compose позволяет легко настроить запуск нескольких контейнеров с помощью декларативного описания запускаемых сервисов и сетей.

3.1.7 AppMetrica

Для сбора информации о работе приложения, пользовательских действиях и проведения аналитики применяется сервис AppMetrica. Благодаря удобному SDK[3] и гибким настройкам, этот сервис позволяет легко организовать отправку пользовательских событий и создавать аналитические отчёты на основе собранных данных.

3.2 Реализация back-end

Разработан backend приложения на основании архитектуры MVC, добавлен Swagger для документации API.

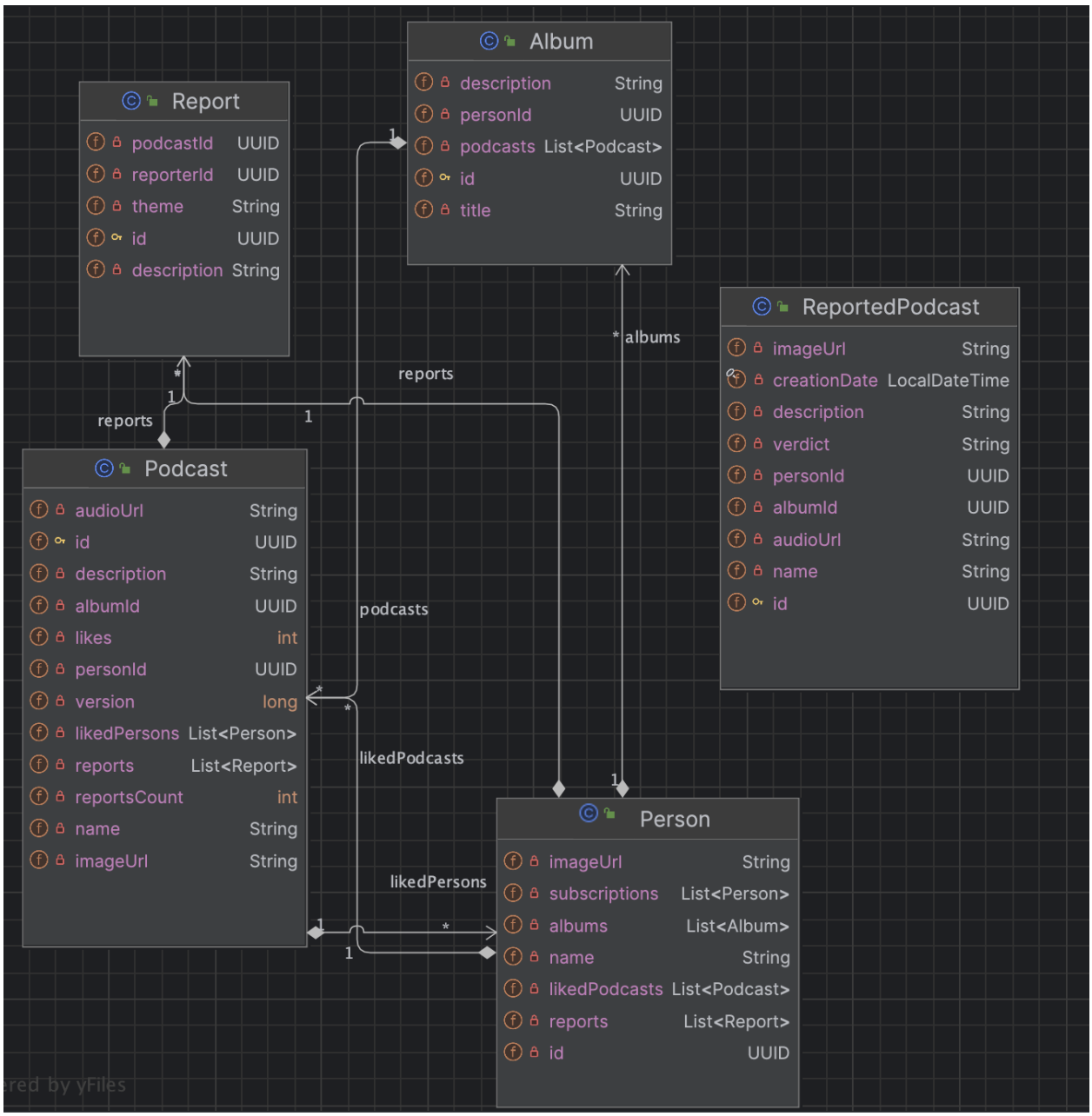

Рисунок 8 — Диаграмма классов

Развернута база данных PostgreSql, спроектирована модель базы данных, включающая сущности и связи между ними, а также настроен Liquibase для накатывания миграций. С помощью фреймоворка Spring Security и стороннего сервиса Keycloak реализованы аутентификация и авторизация с использованием технологии JWT токенов. Отдельно развернута база данных Keycloak хранящая в себе важные данные пользователей. Настроена реализация S3 хранилища – MinIo, в нее сохраняются все изображения и аудиофайлы. Собран образ backend приложения и по нему запущен контейнер.

Также в контейнеры были помещены: база данных PostgreSql, база данных Keycloak, Keycloak, MinIO, хранилище MinIO.

3.2.1 Развертывание

Развертывание приложение будет происходит в Doker в соответствии с диаграммой развертывания

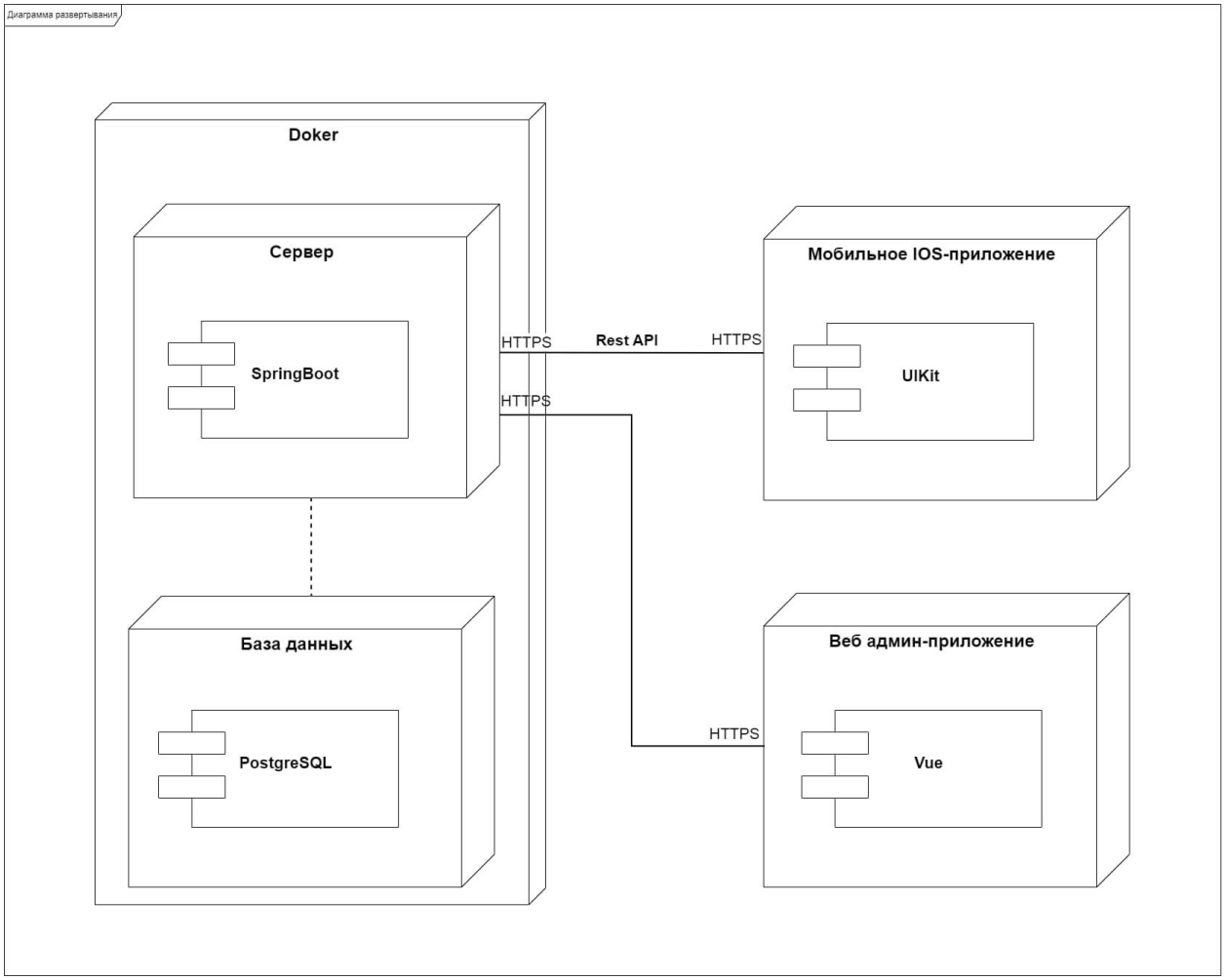

Рисунок 9 — Диаграмма развертывания

3.3 Реализация mobile

Для реализации мобильного приложения использовались фреймворк UIKit и архитектура MVP+R (Model View Presenter + Router).

Архитектура MVP+R (Model-View-Presenter + Router) разделяет ответственность приложения на четыре компонента: Model, View, Presenter и Router. Model управляет данными и бизнес-логикой, включая взаимодействие с внешними источниками данных. View отвечает за отображение данных и пользовательский интерфейс, реагируя на действия пользователя. Presenter

действует как посредник между Model и View, обрабатывая данные от Model и передавая их в View, а также обрабатывая действия пользователя. Router управляет навигацией между экранами и созданием новых модулей приложения, обеспечивая независимость логики навигации от других компонентов.

Преимущества архитектуры MVP+R:

— разделение обязанностей: каждый компонент имеет четко определенные обязанности, что упрощает понимание и поддержку кода;
— тестируемость: благодаря разделению логики на независимые компоненты, каждый из них можно тестировать отдельно;
— поддерживаемость: изменения в одном компоненте минимально влияют на другие компоненты, что облегчает внесение изменений и добавление новых функций;
— переиспользуемость: компоненты, такие как Presenter и Model, могут быть легко переиспользованы в различных частях приложения.

3.3.1 Model

Model – это компонент архитектуры, отвечающий за бизнес-логику и управление данными приложения. Она инкапсулирует данные и обеспечивает методы для их обработки и управления. Model взаимодействует с внешними источниками данных, такими как базы данных, сетевые API и локальные хранилища, и предоставляет данные Presenter для отображения их в View.

Рисунок 10 — Модель подкаста

3.3.2 View

View представляет собой визуальный компонент приложения, который отвечает за отображение данных и взаимодействие с пользователем через UI. Основными элементами View в UIKit являются UIView и UIViewController. В архитектуре MVP+R, View максимально простая, без бизнес-логики, работает с Presenter.

Основные функции View:

— View получает данные от Presenter и обновляет пользовательский интерфейс в соответствии с этими данными;

— View обрабатывает действия пользователя, такие как нажатия кнопок, ввод текста и жесты. Эти события передаются Presenter для дальнейшей обработки;

— View отображает индикаторы загрузки, сообщения об ошибках или успехе и другие формы обратной связи;

— View не содержит бизнес-логику, такую как обработка данных или сложные вычисления. Они вынесены в Presenter.

```
extension LikedPodcastsViewController: UITableViewDelegate {
    func tableView(_ tableView: UITableView, didSelectRowAt indexPath: IndexPath) {
        presenter.showPodcast(index: indexPath.row)
    }
}
```

Рисунок 11 — Пример выбора подкаста и вызова экрана

3.3.3 Presenter

Presenter – это компонент архитектуры, который играет роль посредника между View и Model. Он отвечает за обработку пользовательских действий, получение данных из Model, их подготовку и передачу в View для отображения. Presenter не имеет прямой зависимости от UIKit.

Основные функции Presenter:

— Presenter принимает и обрабатывает действия пользователя, которые происходят в View. Эти действия могут быть инициированы различными взаимодействиями, такими как нажатия кнопок, ввод текста и другие жесты;

— Presenter запрашивает необходимые данные у Model. После получения данных он обрабатывает их и подготавливает для отображения во View;

— Presenter передает подготовленные данные во View для отображения. Он также может инициировать обновление интерфейса View в ответ на изменения данных или состояния;

— Presenter взаимодействует с Router для выполнения навигационных задач.

```
func getPodcast() {
    do {
        let podcast = try castPodcastModelToPodcast(self.podcast)
        viewController?.config(podcast: podcast)
    } catch (let error) {
        viewController?.showErrorAlert(title: "Ошибка", message: error.localizedDescription)
    }
}
```

Рисунок 12 — Пример преобразования модели подкаста и передача в View

3.3.4 Router

Router в архитектуре MVP+R отвечает за навигацию и управление переходами между экранами в приложении. Он отделяет логику навигации от Presenter и View.

Основные функции Router:

— Router осуществляет переходы между различными экранами и модулями приложения;

— Router отвечает за создание и настройку новых экземпляров View, Presenter и Model для целевого экрана;

— Router также может управлять возвратом данных на предыдущие экраны.

```
func showAuthorFrom(_ viewController: UIViewController, author: AuthorModel) {
    let presenter = AuthorPresenter(author: author)
    let authorViewController = AuthorProfileViewController(presenter: presenter)
    presenter.viewController = authorViewController
    viewController.navigationController?.pushViewController(authorViewController, animated: true)
}
```

Рисунок 13 — Пример перехода на экран автора

3.3.5 Service

Service представляет собой компонент, который отвечает за выполнение определенных функций, таких как работа с сетью, доступ к базе данных.

```
func getAlbumById(_ id: UUID, completion: @escaping (Result<AlbumModel, Error>) -> Void) {
    lastTask?.cancel()
    let request = albumByIdRequest(id: id)
    let task = urlSession.objectTask(for: request, completion: { (result: Result<AlbumModel, Error>) in
        switch result {
        case .success(let album):
            completion(.success(album))
        case .failure(let error):
            completion(.failure(error))
        }
    })
    lastTask = task
    task.resume()
}
```

Рисунок 12 - Пример выполнения GET запроса для альбома.

3.3.6 Интерфейс приложения

Интерфейс приложения выполнен в соответствии с принципами «Human Interface Guidelines».

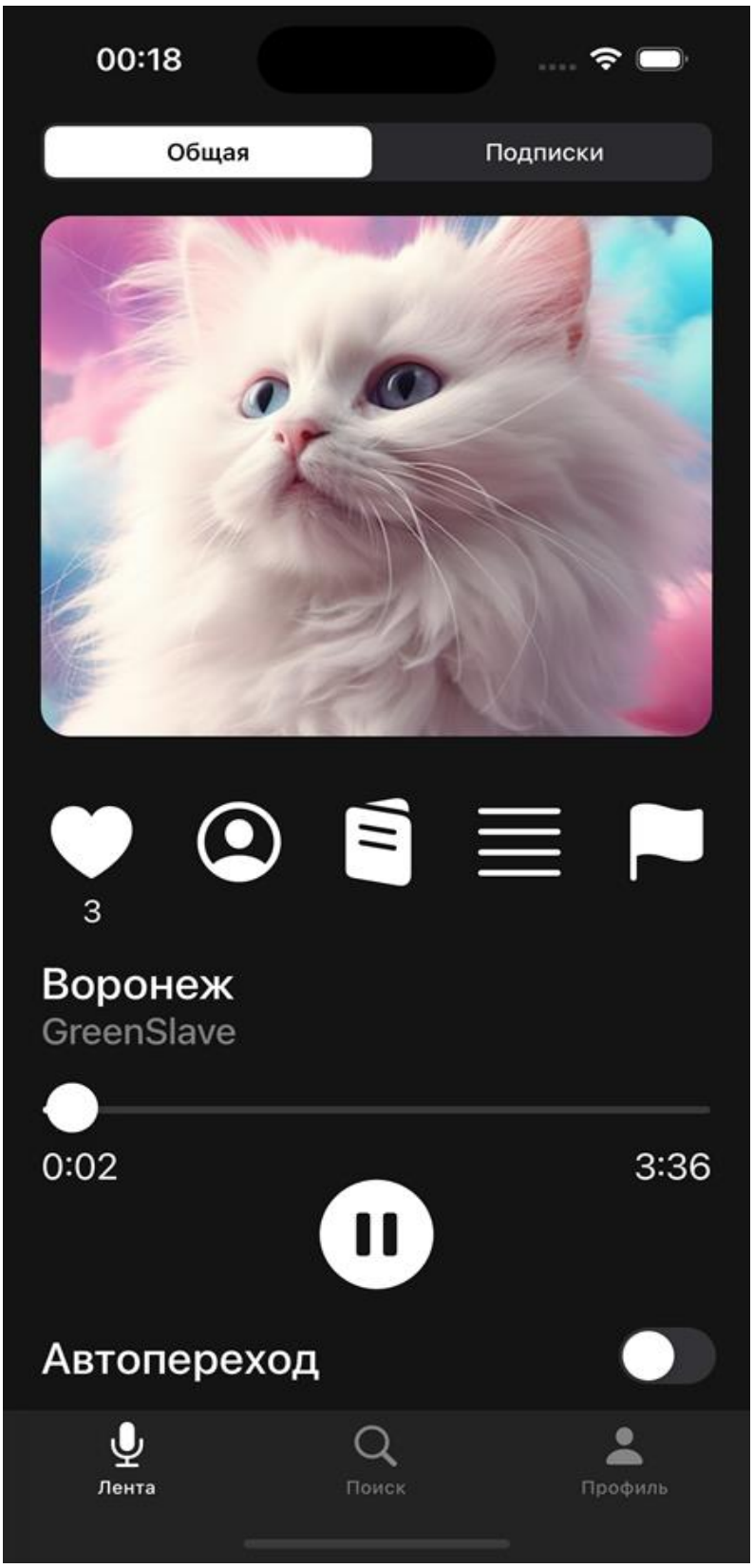

Рисунок 14 — Экран ленты

Примеры остальных экранов веб-приложения будут предоставлены в приложении А.

3.4 Реализация front-end

Для реализации front-end веб приложения использовался фреймворк Vue.js и архитектура MVVM.

Архитектура MVVM разделяет ответственность приложения на 3 компонента: Model, View, ViewModel. Model управляет данными и бизнес-логикой. View отвечает за отображение данных и пользовательский интерфейс, реагируя на действия пользователя. ViewModel – это слой который обрабатывает данные из Model и передает их в View.

3.4.1 Model

Model – это слой архитектуры, отвечающий за бизнес-логику и управление данными приложения. Model взаимодействует с внешними источниками данных (API, базы данных). Model представляет из себя централизованное хранилище с которым можно взаимодействовать с помощью мутаций, действий и геттеров.

Рисунок 15 — Хранилище Vuex

3.4.2 View и ViewModel

View и ViewModel во Vue являются сами компоненты. Компонент – это файл vue в котором содержатся script, template и style секции. Компонент обрабатывает действия пользователя и меняется данными с Model, а также отвечает за отображение данных.

3.4.3 Интерфейс веб-приложения

Итоговый интерфейс веб-приложения для администрации будет выглядеть следующим образом.

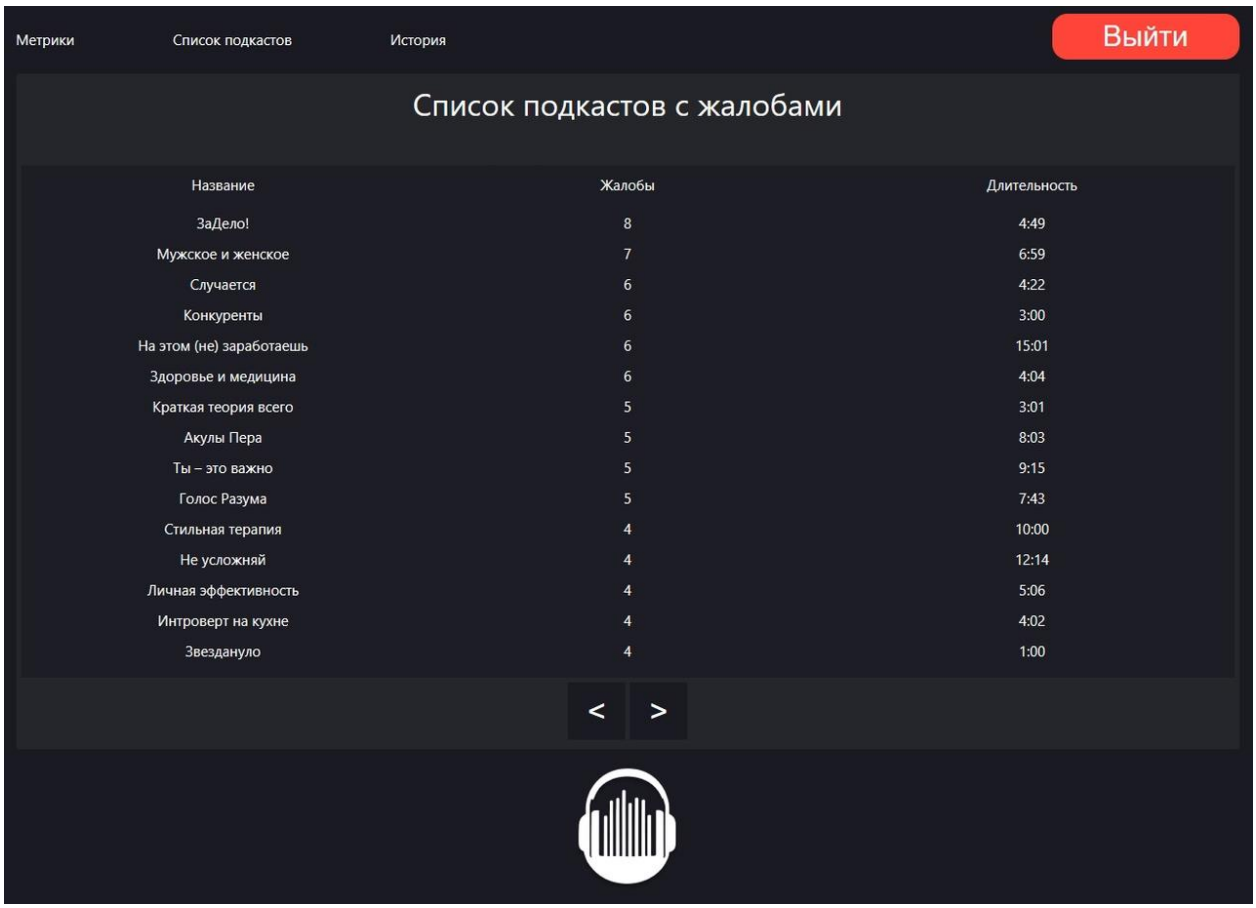

Рисунок 16 — Экран списка жалоб

Примеры остальных экранов веб-приложения будут предоставлены в приложении Б.

Заключение

В результате работы был проведен анализ предметной области, разработано мобильное приложение, помогающие пользователю находить нужные ему подкасты и позволяющие становиться автором контента. А также веб-приложения для администрации, которое поможет сотрудникам очищать приложение от неподобающего контента. Были выполнены следующие задачи:

— разработано mobile-приложение;

— разработана backend часть приложения;

— была создана связь между mobile и backend частями приложения;

— разработана база данных, развернутая на удаленном сервере;

— разработана frontend часть сайта модерации;

— подключена appmetrika, позволяющая фиксировать активность пользователей.

Приложение отвечает всем заявленным в техническом задании требованиям.

Список использованных источников

1. Интернет-торговля в России 2021 [Электронный ресурс]. – Режим доступа: URL: https://datainsight.ru/eCommerce_2021 – Заглавие с экрана (Дата обращения 27.05.2024)

2. Документация SpringBoot [Электронный ресурс]. – Режим доступа: URL: https://docs.spring.io/spring-boot/docs/current/reference/html/ – Заглавие с экрана. – (Дата обращения 23.05.2024).

3. Документация AppMetrica [Электронный ресурс]. – Режим доступа: URL: https://appmetrica.yandex.ru/docs/ru/ – Заглавие с экрана. – (Дата обращения 25.05.2024).

ПРИЛОЖЕНИЕ А
Экраны мобильного приложения

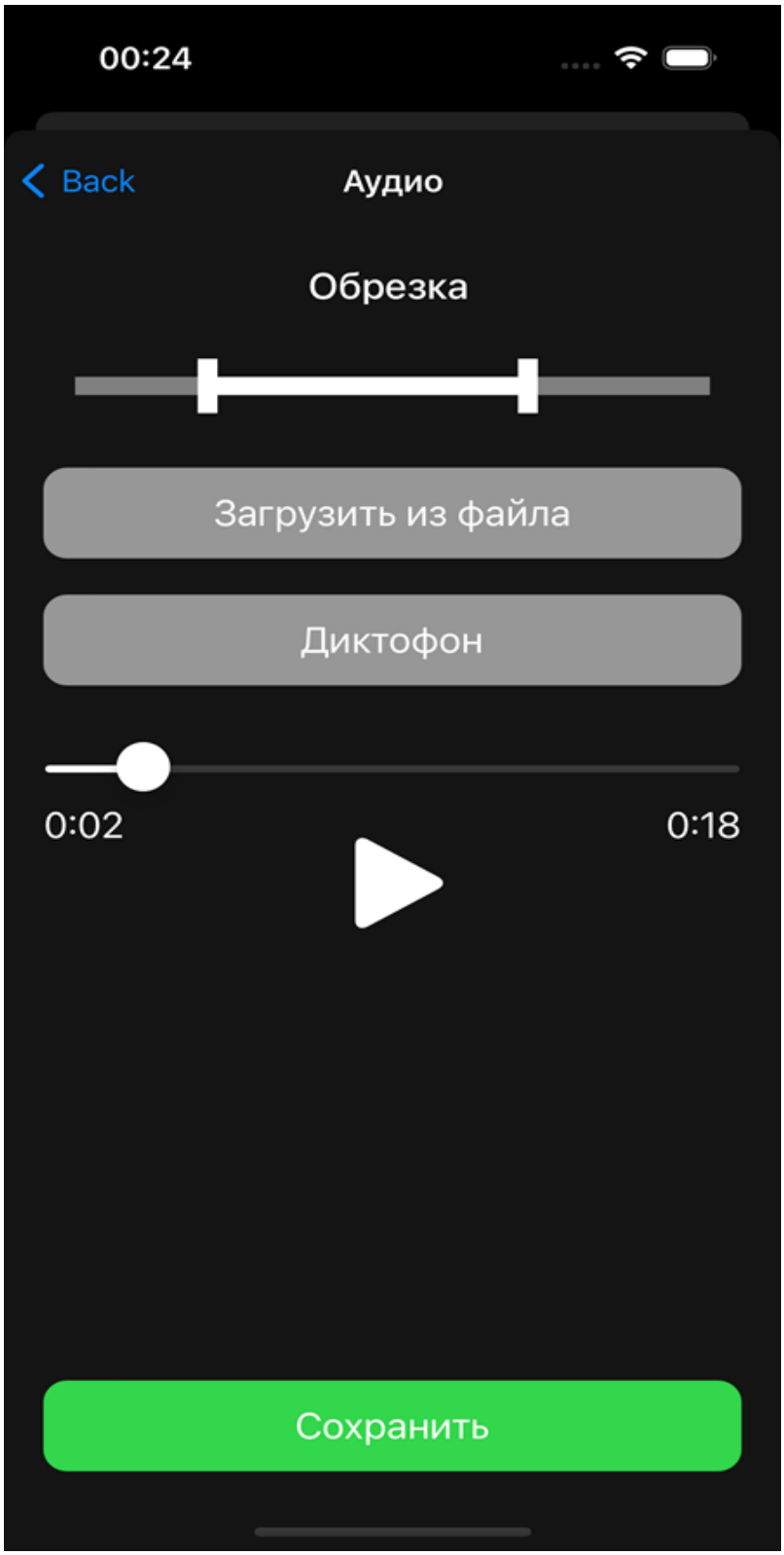

Рисунок 17 — Экран создания аудио

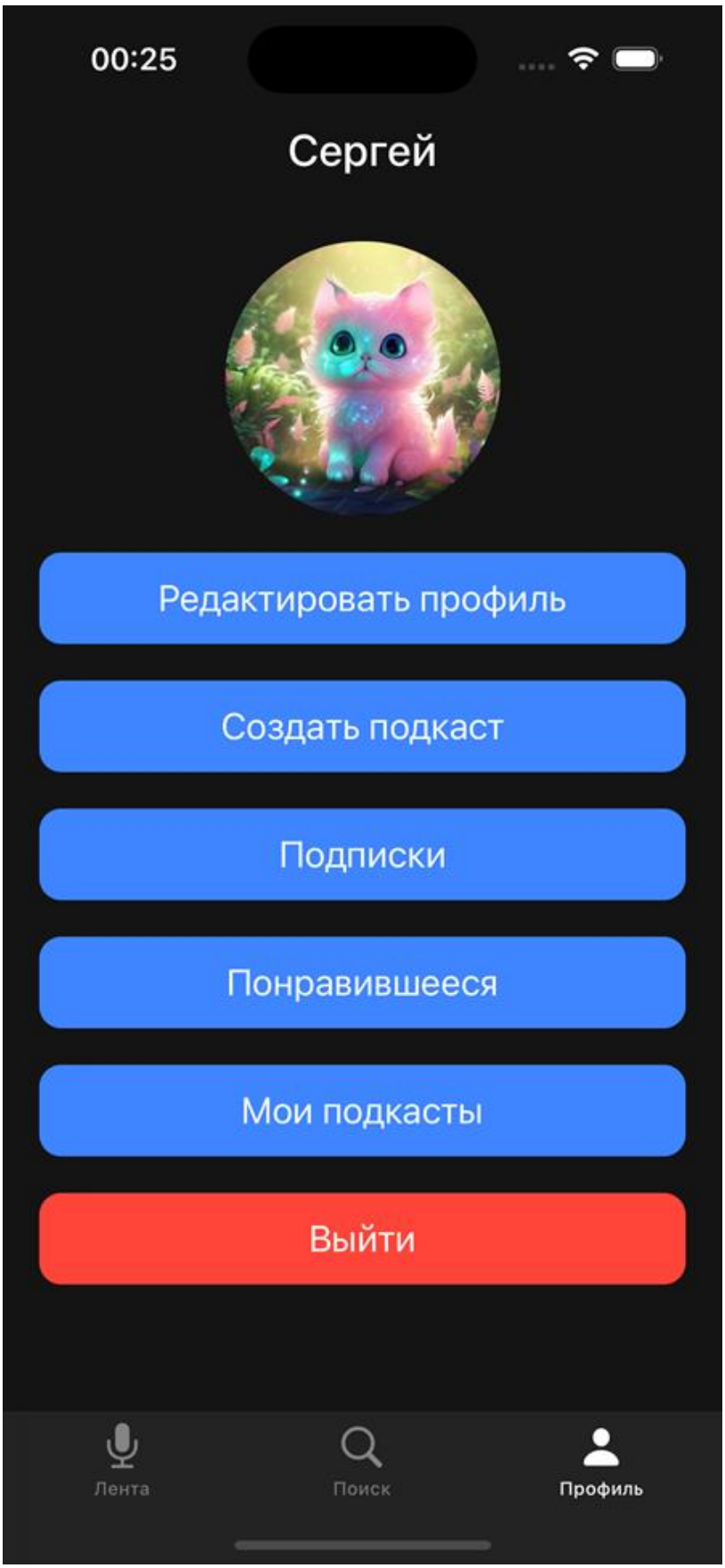

Рисунок 18 — Экран профиля

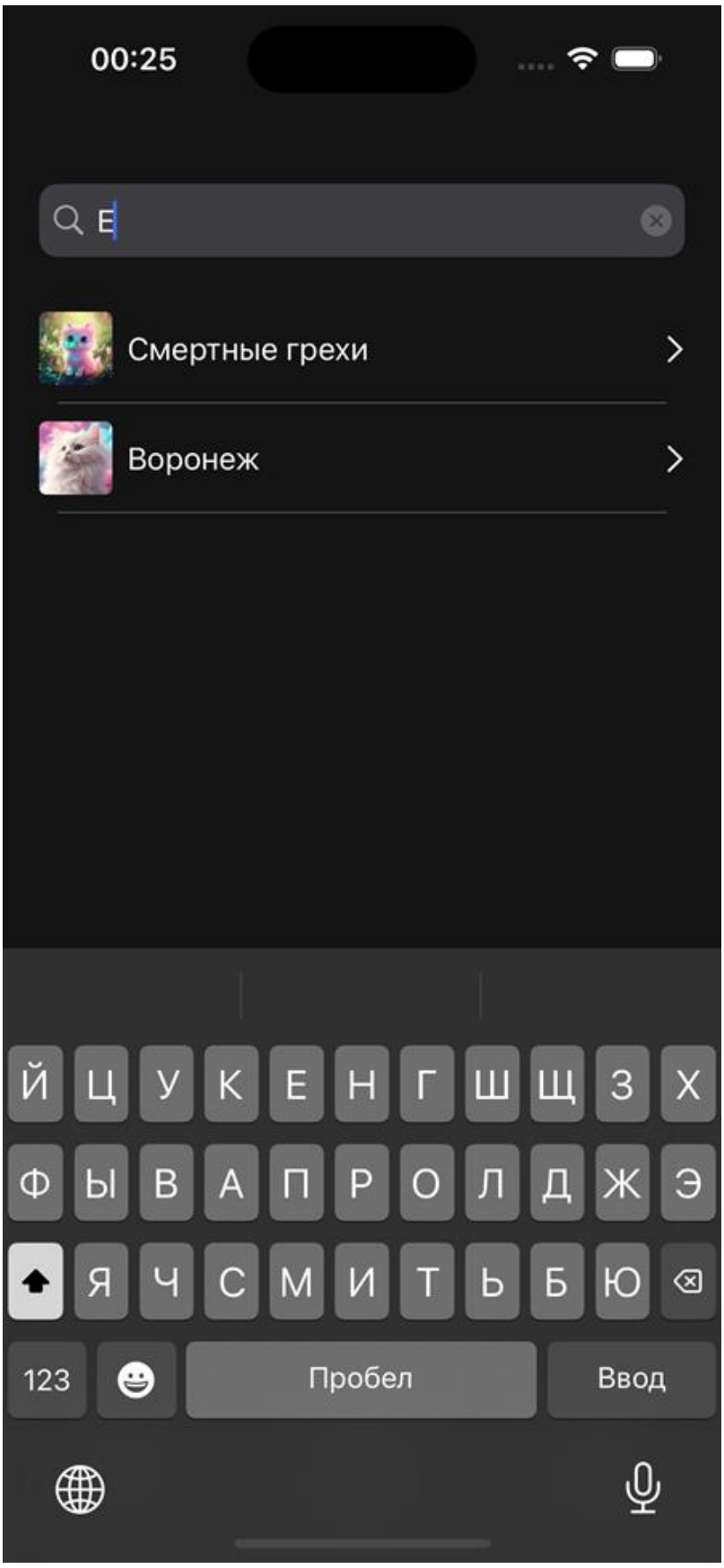

Рисунок 19 — Экран поиска

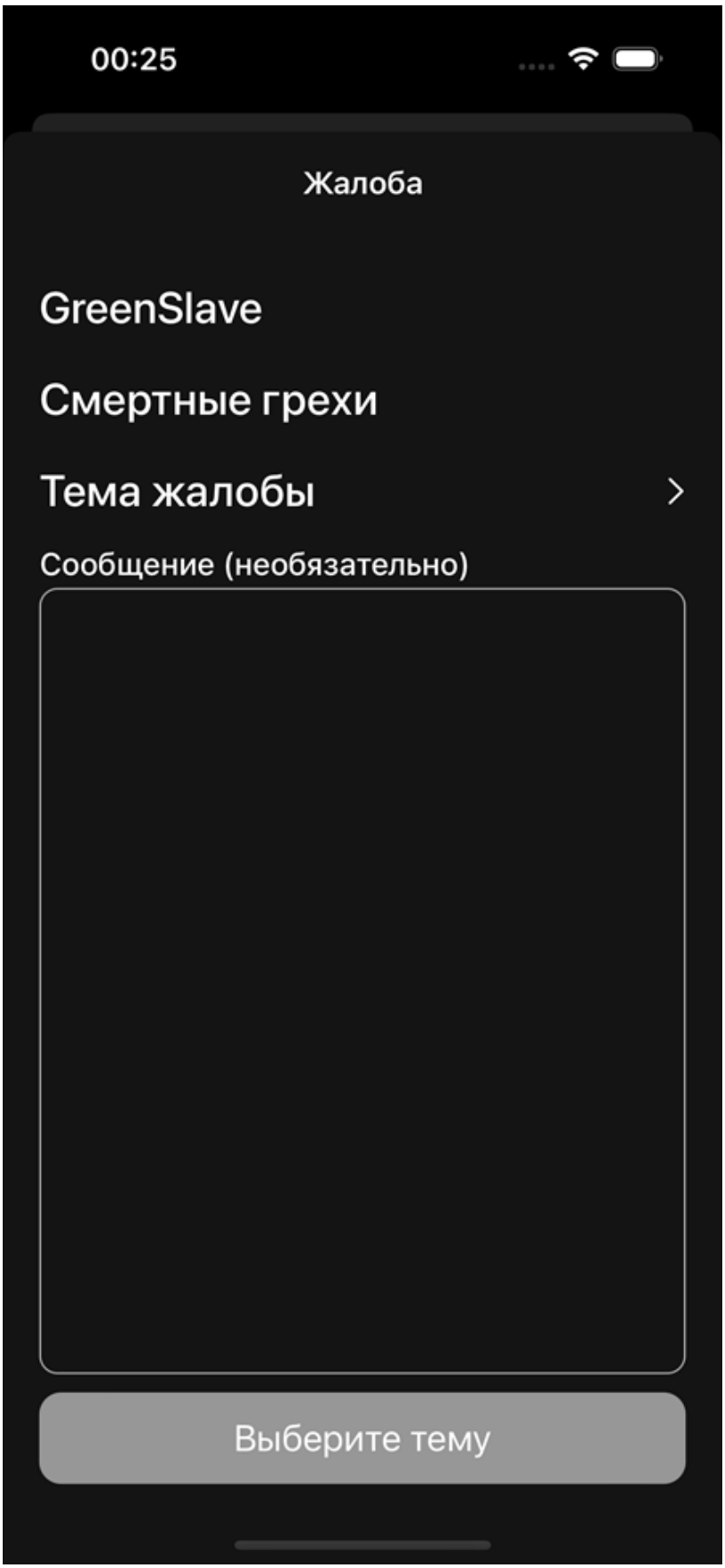

Рисунок 20 — Страница жалобы

ПРИЛОЖЕНИЕ Б
Экраны веб-приложения

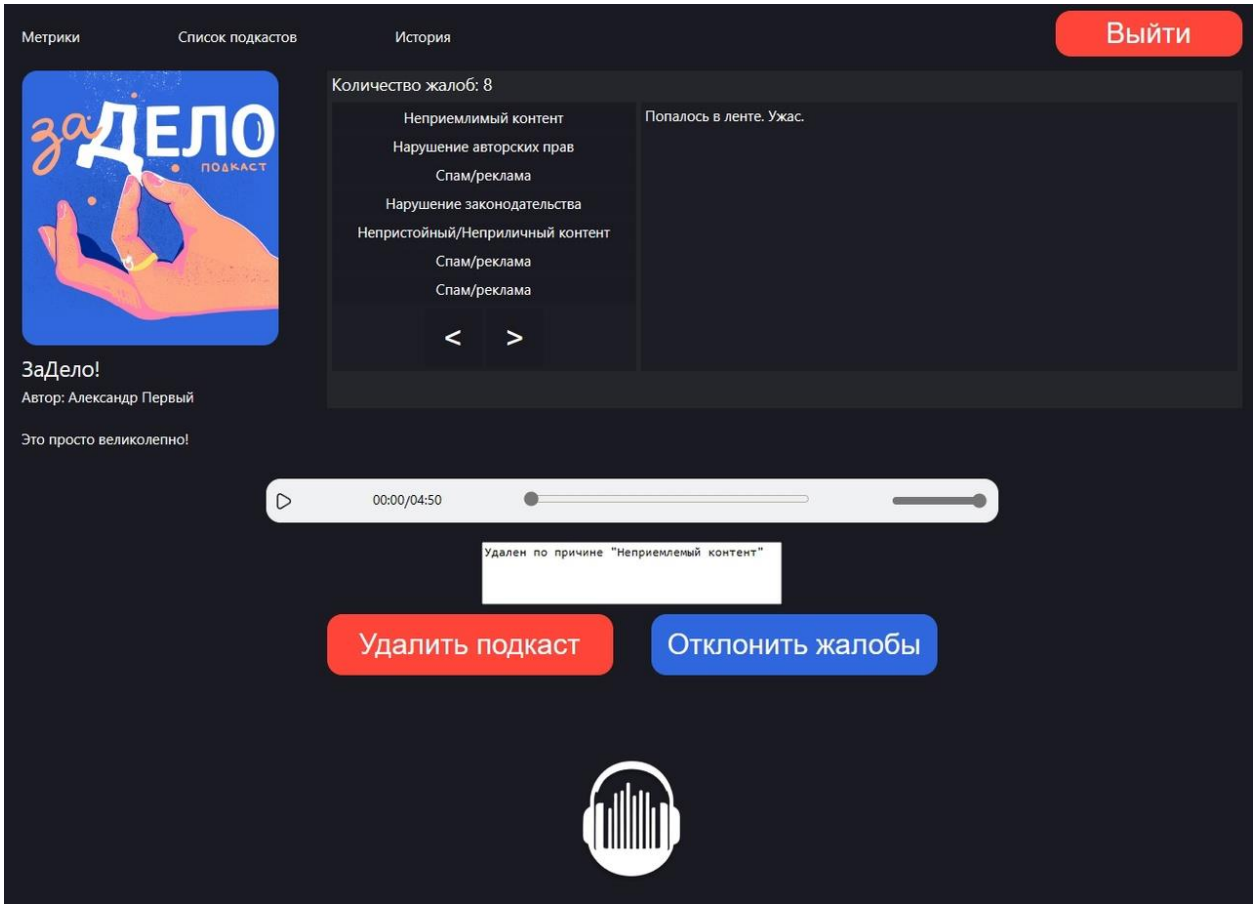

Рисунок 21 — Страница подкаста

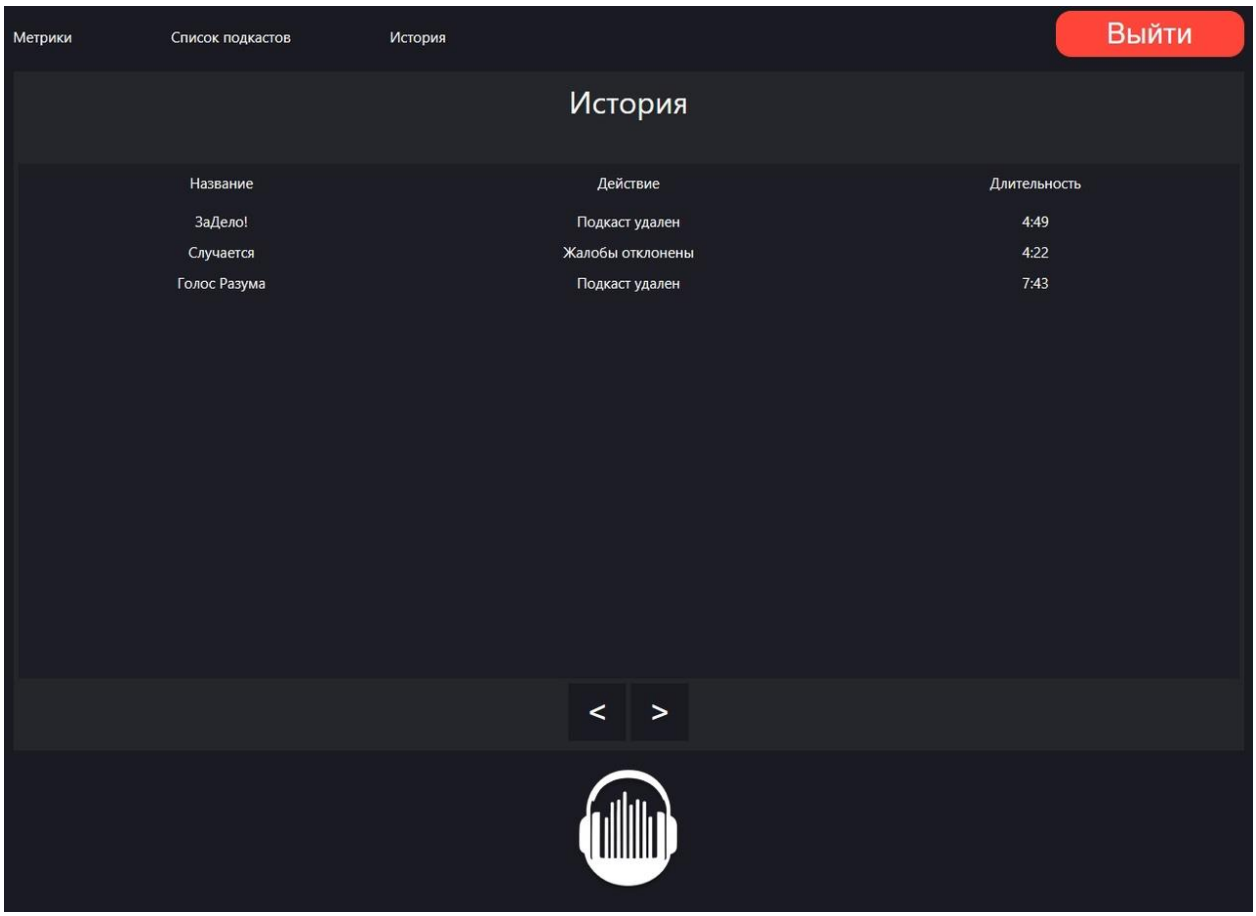

Рисунок 22 — Страница истории

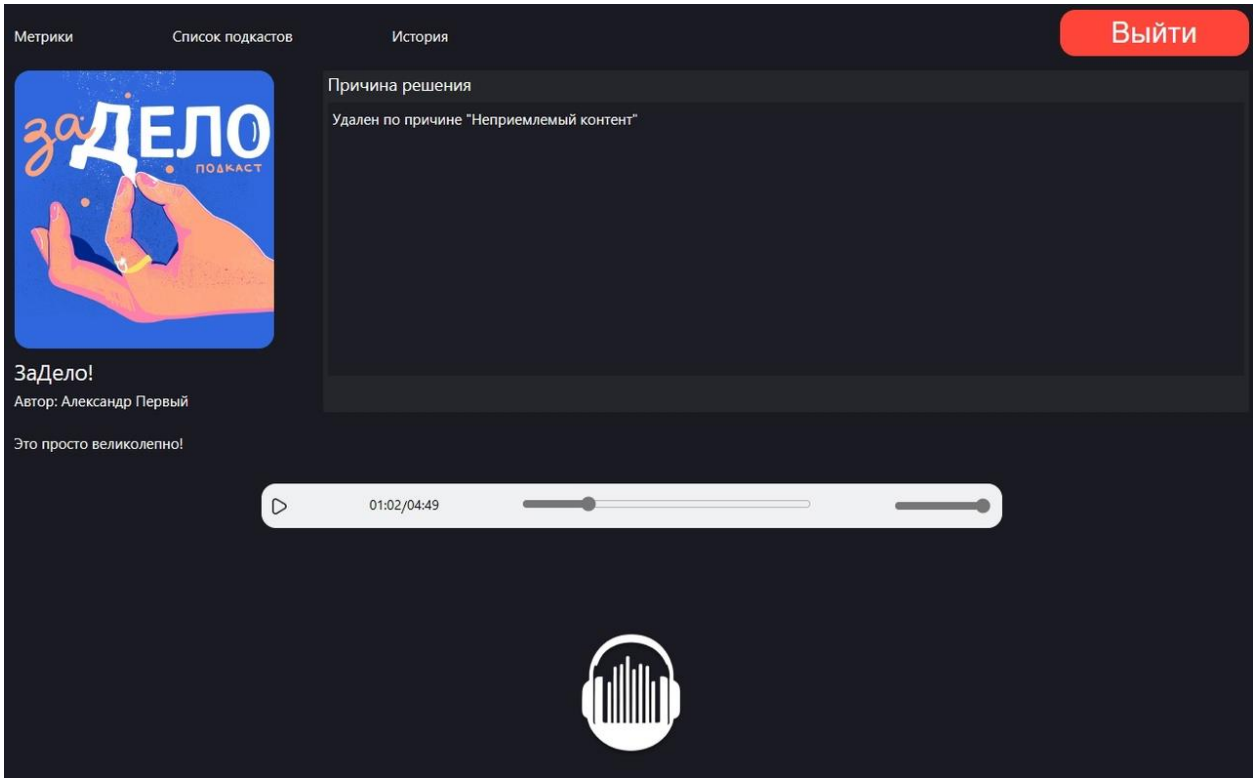

Рисунок 23 — Страница подкаста в истории

ПРИЛОЖЕНИЕ В
Диаграммы

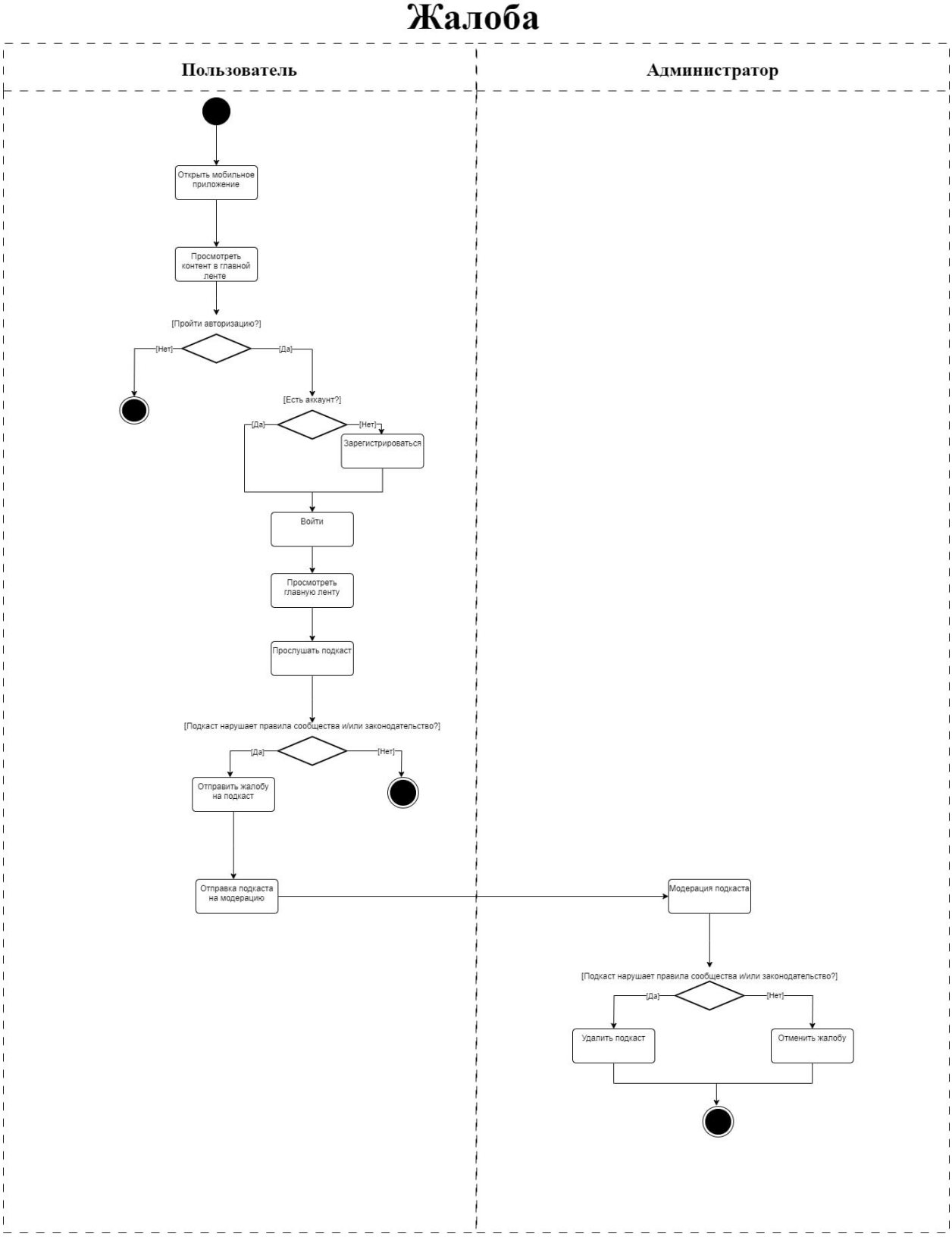

Рисунок 24 — Диаграмма активности Жалоба

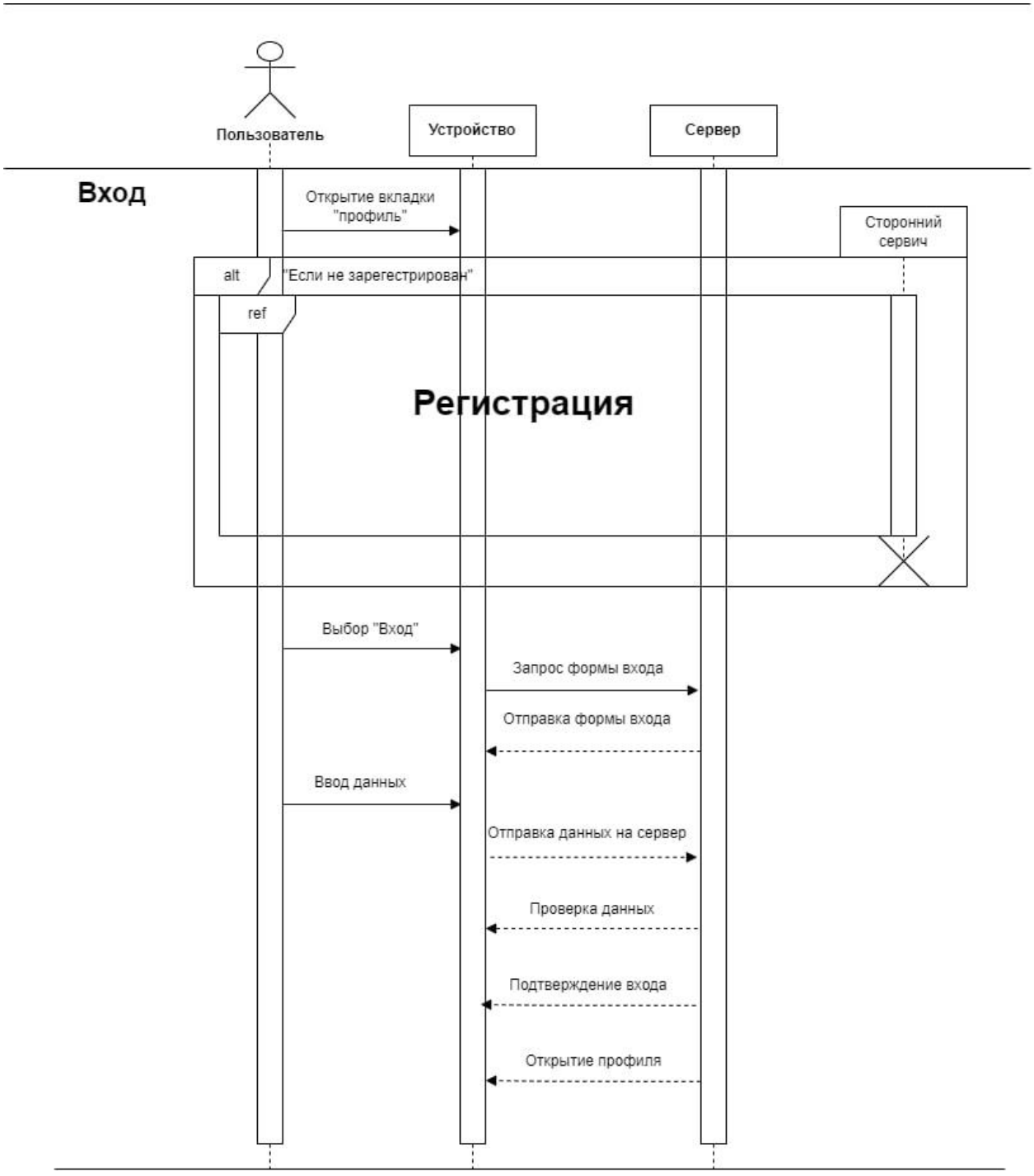

Рисунок 25 — Диаграмма последовательностей Вход

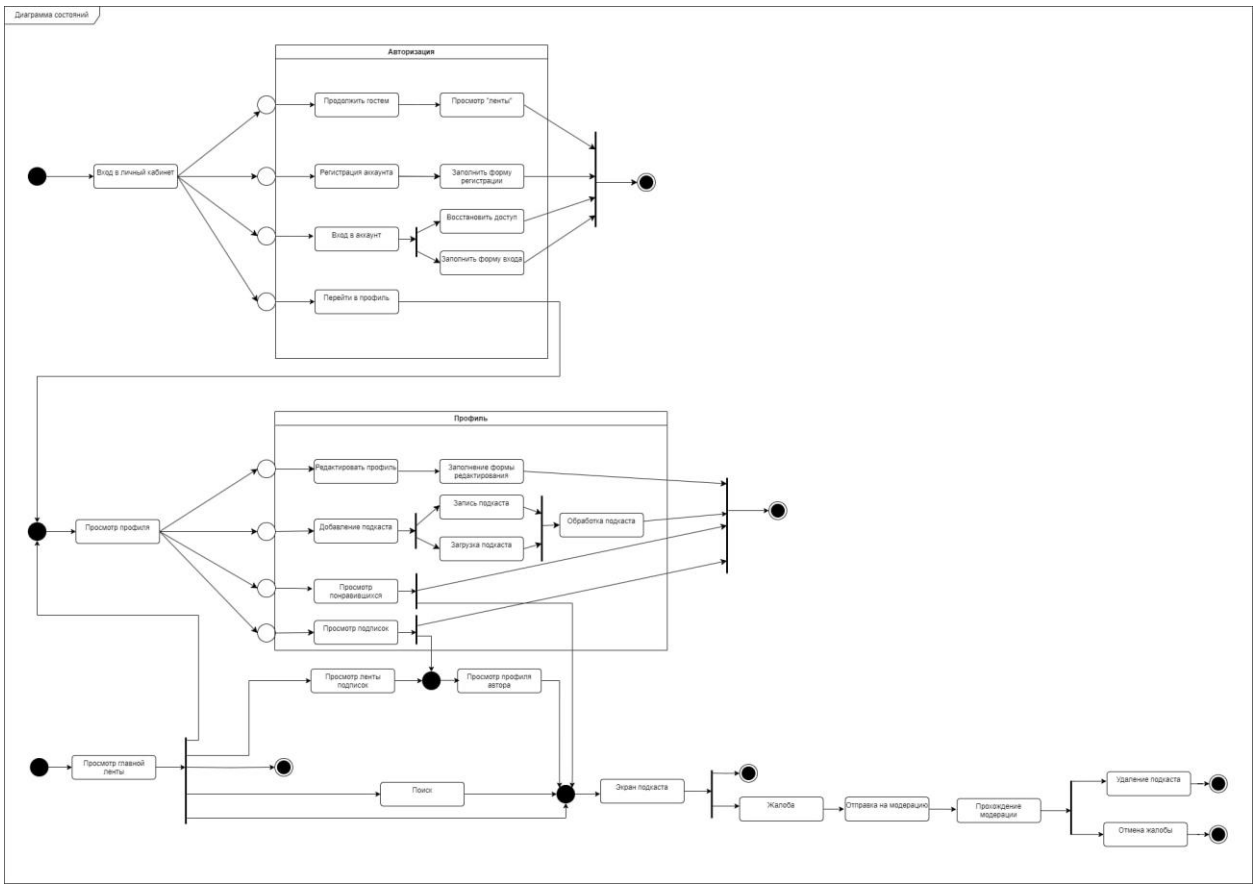

Рисунок 26 — Диаграмма состояний

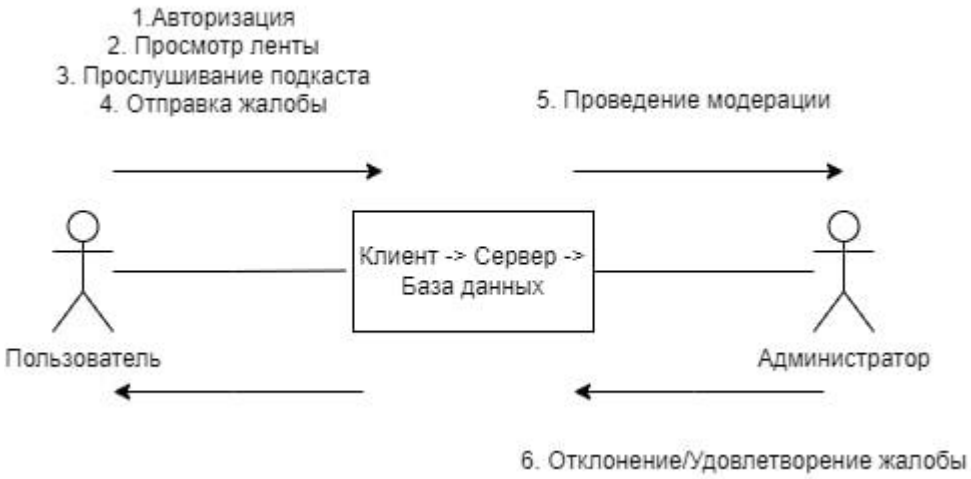

Рисунок 27 — Диаграмма сотрудничества