Министерство науки и высшего образования Российской Федерации

Федеральное государственное автономное образовательное учреждение высшего образования

ТОМСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ СИСТЕМ УПРАВЛЕНИЯ И РАДИОЭЛЕКТРОНИКИ (ТУСУР)

Кафедра автоматизированных систем управления (АСУ)

СИСТЕМА ПОДДЕРЖКИ ДЕЯТЕЛЬНОСТИ МУЗЫКАЛЬНЫХ КОЛЛЕКТИВОВ

Пояснительная записка к курсовому проекту

по дисциплине «Основы разработки программного обеспечения»

Студент гр.431-3

_____Е.П. Бекиш

«___» _____2024г.

Руководитель

Доцент кафедры АСУ, к.т.н.

_____ _____ А.К. Лукьянов

(оценка)

«___» _____2024г.

Томск 2024

Министерство образования и науки Российской Федерации

Федеральное государственное автономное образовательное учреждение высшего образования

ТОМСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

СИСТЕМ УПРАВЛЕНИЯ И РАДИОЭЛЕКТРОНИКИ (ТУСУР)

Кафедра автоматизированных систем управления (АСУ)

ЗАДАНИЕ К КУРСОВОМУ ПРОЕКТУ

по дисциплине «Основы разработки программного обеспечения»

Студенту группы 431-3 Бекиш Егору Павловичу

1. Тема работы: Система поддержки деятельности музыкальных коллективов.
2. Срок сдачи студентом законченной работы «__» _____ 20__ г.
3. Исходные данные: Информация о музыкальных коллективах, их участниках и репетициях.
4. Перечень подлежащих разработке вопросов:

4.1. Описание среды разработки;

4.2. Описание языка программирования;

4.3. Описание проекта;

5. Требования:

5.1. Программа должна работать на большинстве веб-браузерах.

5.2. Интерфейс пользователя должен быть адаптивным.

6. Дата выдачи задания: «__» _____ 2024 г.

Руководитель доцент каф. АСУ, к.т.н. _____ Лукьянов А.К.

Задание принял к использованию _____ Бекиш Е.П.

Оглавление

Введение

В наше технологическое время, когда цифровые решения становятся неотъемлемой частью организационных процессов, важно разрабатывать инновационные системы, способные оптимизировать и упростить управление различными видами деятельности. В этом контексте наш курсовой проект фокусируется на создании системы поддержки деятельности музыкальных коллективов – инструмента, предназначенного для автоматизации управления и координации внутренних процессов музыкальных групп.

В настоящее время процессы репетиций, планирование и взаимодействие внутри музыкальных коллективов часто ограничиваются ручными методами управления и неэффективным взаимодействием между участниками. Наша система призвана устранить эти проблемы, предоставляя возможность создавать внутренние музыкальные группы, эффективно управлять репетиционным процессом и облегчать вступление новых участников в коллектив.

Основные функциональности системы включают в себя создание внутренних групп, планирование репетиций с учетом удобства участников, а также механизм приглашений, обеспечивающий прозрачный процесс принятия новых участников в коллектив. Это не только повышает эффективность управления музыкальными группами, но и способствует улучшению общего опыта участников.

Цель нашего проекта заключается в создании интегрированной системы, которая поможет музыкальным коллективам эффективнее использовать свое время и ресурсы, снижая уровень ручной работы и повышая уровень координации. В дальнейшем мы рассмотрим детали реализации системы, ее ключевые функциональности и потенциальные выгоды для музыкальных групп и их участников.

Объектом исследования являются музыкальные коллективы, предметом исследования выступает процесс взаимодействия между участниками и руководителями этих коллективов. Акцент делается на систематическом анализе динамики внутриколлективных взаимодействий, включая процессы формирования внутренних групп, координации репетиций, управления приглашениями новых участников, а также эффективного обмена информацией между участниками и руководителями.

Цель проекта – реализовать полноценную систему поддержки деятельности музыкальных коллективов, обеспечивающую эффективное управление группами, планирование репетиций и удобное вступление новых участников.

Основные задачи:

- изучить предметную область;
- сформировать требования к системе;
- сформировать спецификации в системе;
- спроектировать систему;
- реализовать серверную часть системы;
- реализовать клиентскую часть системы.

1 АНАЛИЗ ТРЕБОВАНИЙ

1.1 Особенности предметной области

Поскольку объектом исследования были указаны музыкальные коллективы, то необходимо учесть ряд моментов, упоминание которых чрезвычайно важно для проектирования системы.

Начнем с участников бизнес-процесса и их участия в системе. Всего в процессе задействовано 3 роли, а именно:

- участники;
- заместители музыкальных групп;
- руководители музыкальных коллективов.

Участники представляют собой действующих членов коллектива, взаимодействующих с системой для согласования репетиций. Заместители руководят участниками музыкальных групп, и их роль в системе направлена на обеспечение непрерывности репетиционного процесса. Руководители музыкальных коллективов имеют доступ к функциям управления группами, включая планирование репетиций и управление приглашениями.

Одной из значимых особенностей предметной области является нестандартность ситуаций, с которыми могут столкнуться участники музыкальных коллективов. Например, необходимость планирования репетиций при участии музыкантов с разными графиками или изменения в составе группы. Эти факторы требуют гибкости и адаптивности системы поддержки деятельности, чтобы учесть различные сценарии взаимодействия и обеспечить эффективное управление коллективом.

Дополнительно, важным аспектом является обеспечение коммуникации между участниками системы. Учитывая специфику музыкальной деятельности, где четкое взаимопонимание между участниками и заместителями играет ключевую роль, система должна предоставлять

средства для оперативного обмена информацией, планирования событий и принятия решений в реальном времени.

1.1 Модель бизнес-процесса

Для более наглядного понимания процесса была реализована модель TO BE с использованием нотации bpmn 2.0. Для начала краткое пояснение, что подразумевается под моделью TO BE. TO BE – модель позволяющая систематизировать протекающие в системе процессы, а также используемы информационные объекты [1]. Теперь все также кратко о нотации bpmn 2.0. Bpmn 2.0 - система условных обозначений, которая отображает бизнес-процессы с помощью блок-схем. BPMN диаграмма показывает в какой последовательности совершаются рабочие действия и перемещаются потоки информации [2].

Отображение действий руководителя в процессе принятия/отклонения приглашений в музыкальный коллектив на TO BE bpmn модели представлено на рисунке 1.1.

По существу, старт процесса начинается с момента формирования нового музыкального коллектива, однако же в действительности старт деятельности происходит после формирования основного состава коллектива, решившего расширять свой состав дополнительными группами.

Музыканты, желающие вступить в музыкальный коллектив, просматривают группы через социальные сети и отправляют заявку на вступление в личные сообщения группы. Далее, если руководитель коллектива приглашает на прослушивание и одобряет вступление в коллектив, то дальнейшее их взаимодействие осуществляется внутри коллектива. В противном случае музыкант продолжает свои поиски по другим коллективам или меняет свои приоритеты.

Перейдем к разбору действий руководителя и менеджеров. Отображение действий на TO BE bpmn модели представлено на рисунке 1.2.

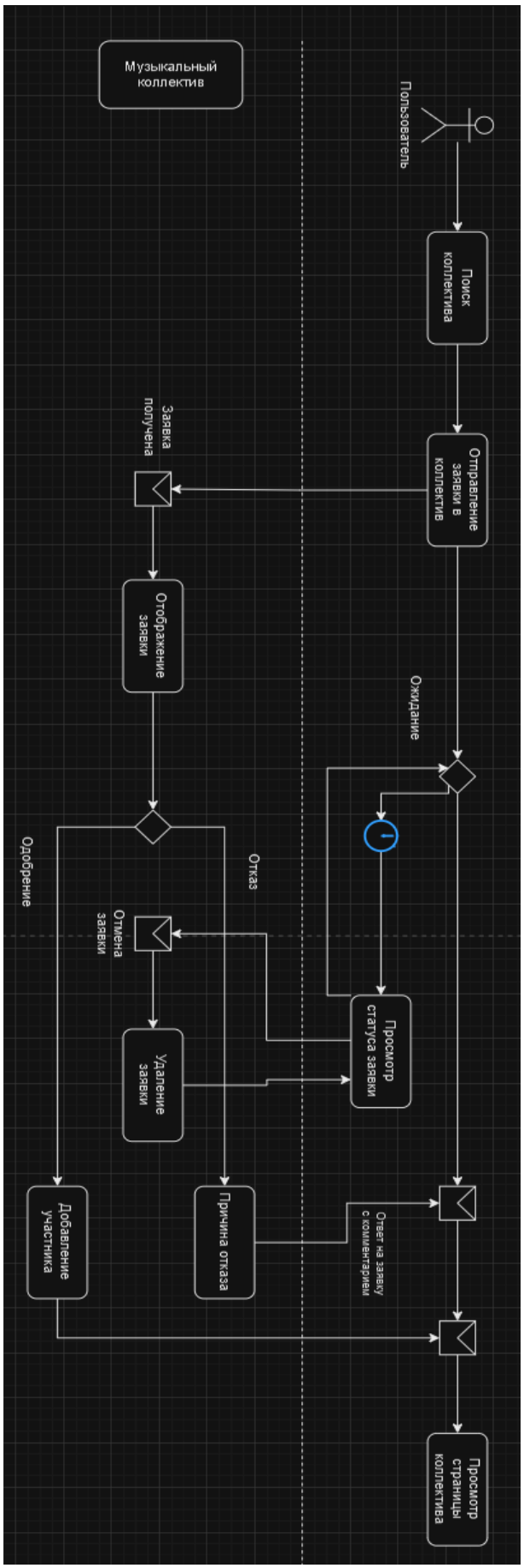

Рисунок 1.1 – Действия пользователя и руководителя

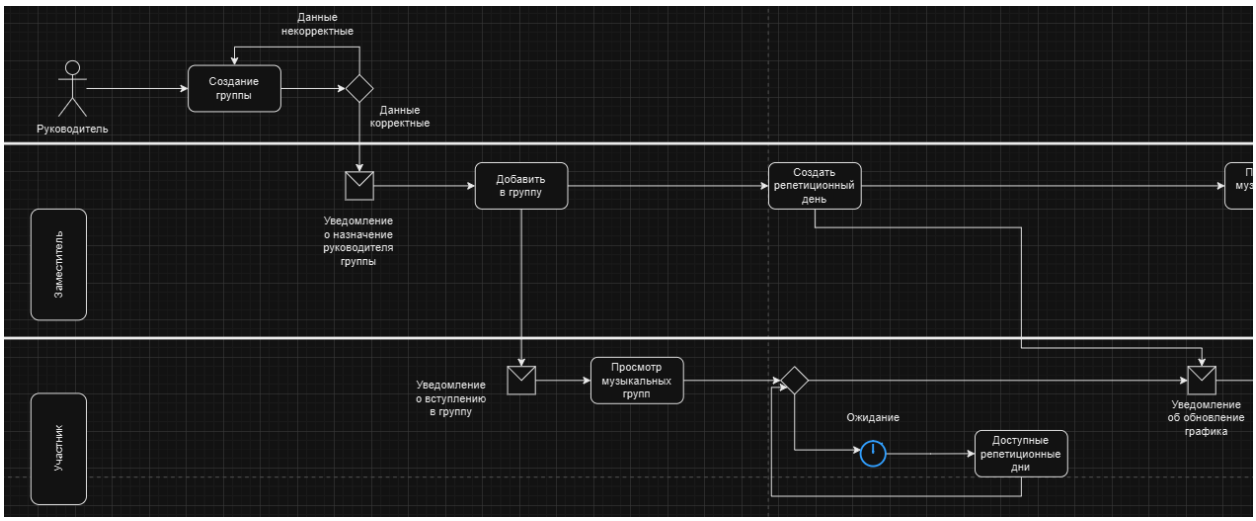

Рисунок 1.2. – Действия руководителя и заместителя

Деятельность музыкальной группы начинается после её формирования руководителем коллектива. Для этого руководитель назначает заместителя группы, координирующего репетиционный процесс участников. Далее заместитель наполняет группу участниками и формирует график репетиционных дней. В свою очередь участники этой группы ожидают анонсирование предварительного расписания, чтобы в дальнейшем предложить удобное для себя время.

Далее рассмотрим действия заместителя и участников в контексте согласования времени для репетиционного дня, как представлено на рисунке 1.3.

После обновления графика репетиционных дней, участники группы отправляют релевантное время репетиции, тогда как заместитель ожидает список предложений. После чего за группой закрепляется репетиционное время и процесс повторяется с момента согласования следующего репетиционного дня.

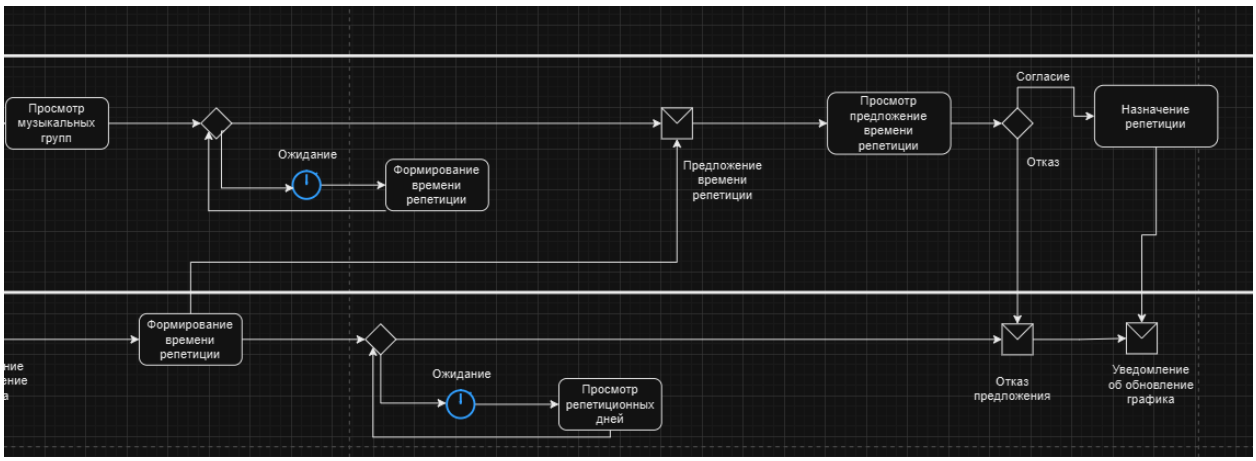

Рисунок 1.3 – Действия заместителя и участников

1.3 Формулировка требований к системе

1.3.1 Требования к составу выполняемых функций

Разрабатываемая система должна обеспечивать:

1. функционал администрирования системы для взаимодействия с базой данных;

2. функционал, позволяющий добавлять и изменять информацию о пользователе;

3. функционал, позволяющий удалить аккаунт пользователя;

4. функционал, позволяющий добавлять и удалять заявку;

5. функционал, позволяющий осуществлять поиск музыкальных коллективов;

6. функционал, позволяющий получать информацию о музыкальных группах, в которых состоит пользователь.

7. функциональные возможности, реализуемые компонентами, входящими в его состав: аутентификация пользователя, которая будет реализовывать:

7.1 авторизацию пользователя;

7.2 регистрацию пользователя.

1.3.2 Требования к организации входных данных

Входными данными в разрабатываемой системе должны являться такие модели как: музыкальный коллектив, музыкальная группа, заявка, участник коллектива и репетиция.

Основными данными, вводимыми в систему, должны являться:

- участник;
- музыкальный коллектив;
- музыкальная группа;
- репетиционный день.

Формат данных для участника представлен в таблице 1.1.

Таблица 1.1 – Формат данных модели участника

ФИО	email	Номер телефона	Музыкальный коллектив	Должность
Текстовый	Текстовый	Текстовый	Внешняя ссылка	Текстовый

Формат данных для репетиционного дня представлен в таблице 1.2.

Таблица 1.2 – Формат данных модели репетиционного дня

Группа	Дата	Начало окна предложений	Конец окна предложений	Макс. число участников
Внешняя ссылка	Текстовый	Числовой	Числовой	Числовой

Формат данных для коллектива представлен в таблице 1.3.

Таблица 1.3 – Формат данных модели музыкального коллектива

Название	Список закреп. групп	Список закреп. участников
Текстовый	Внешняя ссылка	Внешняя ссылка

Формат данных для музыкальной группы представлен в таблице 1.4.

Таблица 1.4 – Формат данных модели музыкальной группы

Название	Список закреп. участников	Заместитель группы
Текстовый	Внешняя ссылка	Внешняя ссылка

1.3.3 Требования к организации выходных данных

Выходными данными разрабатываемой системы должны являться результаты выполнения запросов, вывод экранных форм.

Результат поиска по критериям запроса должен выдавать следующую информацию:

- Участник:
  - ФИО;
  - email;
  - номер телефона;
  - музыкальная группа;
  - должность.
- Музыкальный коллектив:
  - название;
  - руководитель;
  - количество участников;
  - дата создания.
- Музыкальная группа:
  - название;
  - музыкальный коллектив;
  - заместитель;
  - количество участников;
  - дата создания.
- Репетиционный день:
  - музыкальная группа;
  - дата;

- время начало окна;
- время окончания окна;
- максимальное количество участников;
- статистика предложений.

1.3.4 Требования к временным характеристикам

Разрабатываемая система должна, в зависимости от размерности базы данных,обеспечивать время выполнения заданий в интервале от 0.5 секунд до 2-ой минуты при плохом интернете и большом объеме данных.

Требования к надёжности.

Разрабатываемая система должна удовлетворять следующим требованиям по времени восстановления после отказа, например:

- среднее время восстановления работоспособного состояния после отказа, вызванного неисправностью (сбоем) самого разрабатываемого Web-приложения, должно составлятьне более 2 часов;
- время восстановления после отказа, вызванного сбоем электропитания технических средств (и/или иными внешними факторами), не фатальным сбоем (не крахом) операционной системы, при условии соблюдения условий эксплуатации технических и программных средств не более 24 часов;
- время восстановления после отказа, вызванного неисправностью технических средств, фатальным сбоем (крахом) операционной системы, не должно превышать времени, требуемого на устранение неисправностей технических средстви переустановки программных средств.

Критериями отказа и предельного состояния разрабатываемого Web-приложение являются:

- Критерии отказа:
  - аварийное прекращение работы программного обеспечения;
  - невозможность системы в полном объеме выполнять заданные функции;

▪ превышение времени выполнения запросов на представление данных (более 3 минут).

Требования к составу и параметрам технических средств.

Разрабатываемые технологии Web-приложения должны функционировать на следующих технических средствах: на сервере со следующими характеристиками: процессор Intel x86 (2 ядра) частота от 2 ГГц и выше, ОЗУ 4 Гб и выше.

1.4 Сравнение системы с аналогами

Проанализировав аналоги системы, были выявлены две платформы: BandLab и ProCollabs.

BandLab предоставляет возможности для онлайн-сотрудничества музыкантов. Однако, в отличие от BandLab, система поддержки деятельности музыкальных коллективов фокусируется на управлении и координации крупных музыкальных групп. Там, где BandLab ориентирован на создание и запись музыки в цифровой среде, Система включает в себя более сложные механизмы управления расписанием репетиций, формирования внутренних групп и эффективного взаимодействия участников в крупных коллективах.

ProCollabs — это онлайн-сервис для совместной работы над музыкой, который предоставляет доступ к авторам песен, музыкантам, звукорежиссерам и музыкальным продюсерам. Это место для объединения, общения и совместной работы над написанием и продюсированием оригинальной музыки. В отличите от ProCollabs, система поддержки деятельности музыкальных коллективов ориентирована на более крупные проекты, где важна координация между большим количеством музыкантов.

2 ОПРЕДЕЛЕНИЕ СПЕЦИФИКАЦИЙ

2.1 Модель проектирования и разработки системы

Каскадная модель — модель процесса разработки программного обеспечения, в которой процесс разработки выглядит как поток, последовательно проходящий фазы анализа требований, проектирования, реализации, тестирования, интеграции и поддержки.

Следуя каскадной модели, разработчик переходит от одной стадии к другой строго последовательно. Сначала полностью завершается этап «определение требований», в результате чего получается список требований к ПО. После того как требования полностью определены, происходит переход к проектированию, в ходе которого создаются документы, подробно описывающие для программистов способ и план реализации указанных требований. После того как проектирование полностью выполнено, программистами выполняется реализация полученного проекта. На следующей стадии процесса происходит интеграция отдельных компонентов, разрабатываемых различными командами программистов. После того как реализация и интеграция завершены, производится тестирование и отладка продукта; на этой стадии устраняются все недочёты, появившиеся на предыдущих стадиях разработки. После этого программный продукт внедряется и обеспечивается его поддержка — внесение новой функциональности и устранение ошибок.

Каскадная модель была выбрана по причине подходящего алгоритма для разработки учебного проекта. Т.к. в рамках учебной разработки все этапы идут последовательно и после утверждения не требуют каких-либо дополнений [3].

2.2 Инструменты разработки

Для разработки десктоп версии был выбран ряд следующих инструментов.

HyperText Markup Language – это стандартизированный язык разметки документов в интернет-пространстве. Он определяет содержание и структуру веб–контента. HTML использует разметку для отображения различных заголовков, текстовых абзацев, изображений и прочего контента в веб-браузере [6].

Cascading Style Sheets (далее – CSS) – это язык, который позволяет изменить оформление внешнего вида документа отдельно от его содержания [7].

Помимо этого, также будут использоваться вспомогательные библиотеки такие как:

Axios — это библиотека с открытым исходным кодом, позволяющая делать HTTP-запросы. Она предоставляет методы .get(), .post(), put() и .delete() [8];

Flask — это веб-фреймворк на Python высокого уровня, который поощряет быструю разработку и чистый, прагматичный дизайн. Он берет на себя большую часть хлопот по веб-разработке, поэтому разработчик может сосредоточиться на написании своего приложения без необходимости изобретать велосипед. Он бесплатный с открытым исходным кодом. [10];

2.3 Входные и выходные данные

Входные данные:

- контактные данные пользователей;
- заявления пользователей;
- контактные данные музыкальных коллективов;
- должности в коллективе;

- репетиционные окна.

Выходные данные:

- база музыкальных коллективов;
- статусы принятия заявки на вступление в коллектив;
- статусы репетиционных предложений;
- информация об участниках и музыкальных коллективах, в которых они состоят;
- информация об участниках и музыкальных группах, к которым они принадлежат;
- график репетиционных дней с информацией о предложениях времени репетиции с соответствующим статусом;
- таблица заявок пользователя на вступление в музыкальные коллективы;
- таблица музыкальных групп, к которым принадлежит участник.

3 ПРОЕКТИРОВАНИЕ

3.1 Проектирование базы данных

Проанализировав предметную область и определив спецификации, можно перейти к проектированию базы данных. В результате выполнения данного этапа получилась схема базы данных, представленная на рисунке 3.1.

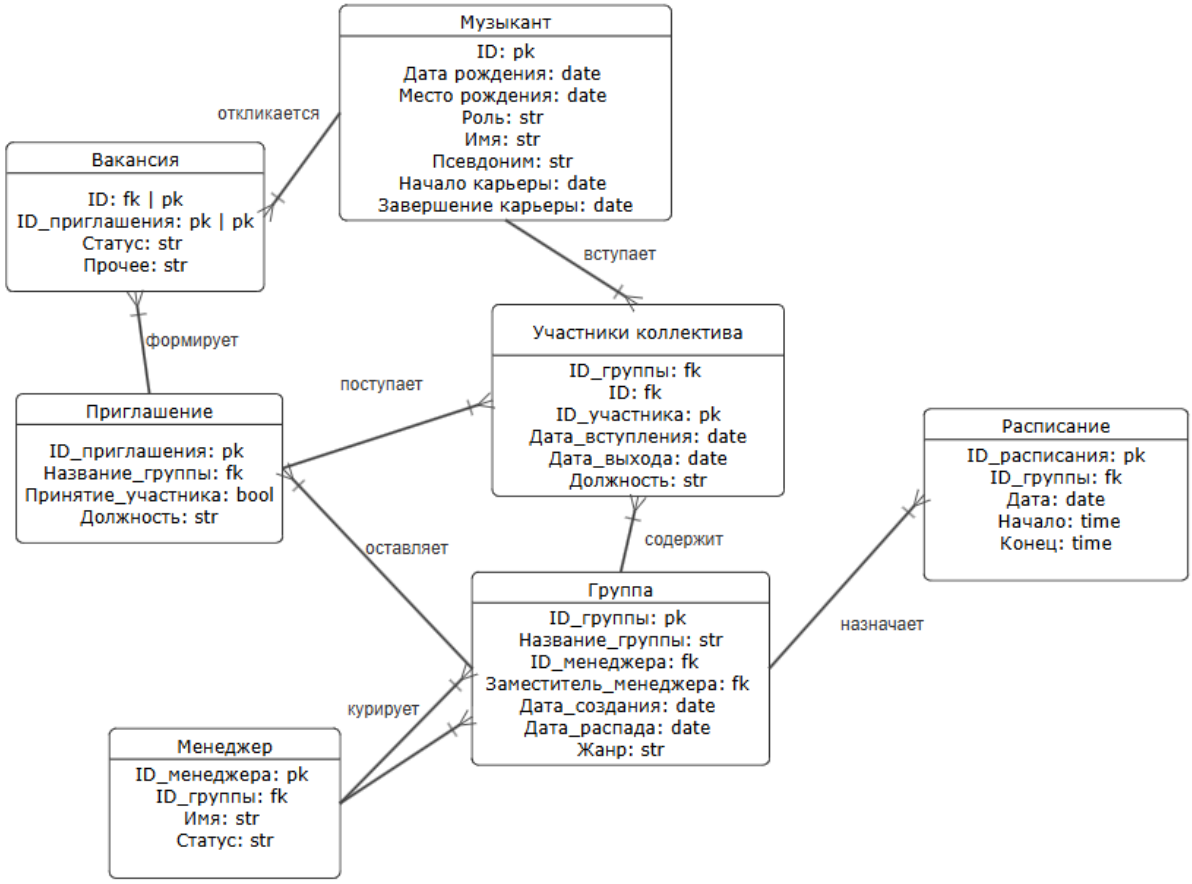

Рисунок 3.1 – Физическая модель базы данных

3.2 Проектирование программной структуры Системы

На данном этапе необходимо определить состояния сущностей, с которыми идет работа системы. Для этого с помощью диаграммы состояний отобразим всевозможные состояния пользователей. Диаграмма состояний представлена на рисунке 3.2.

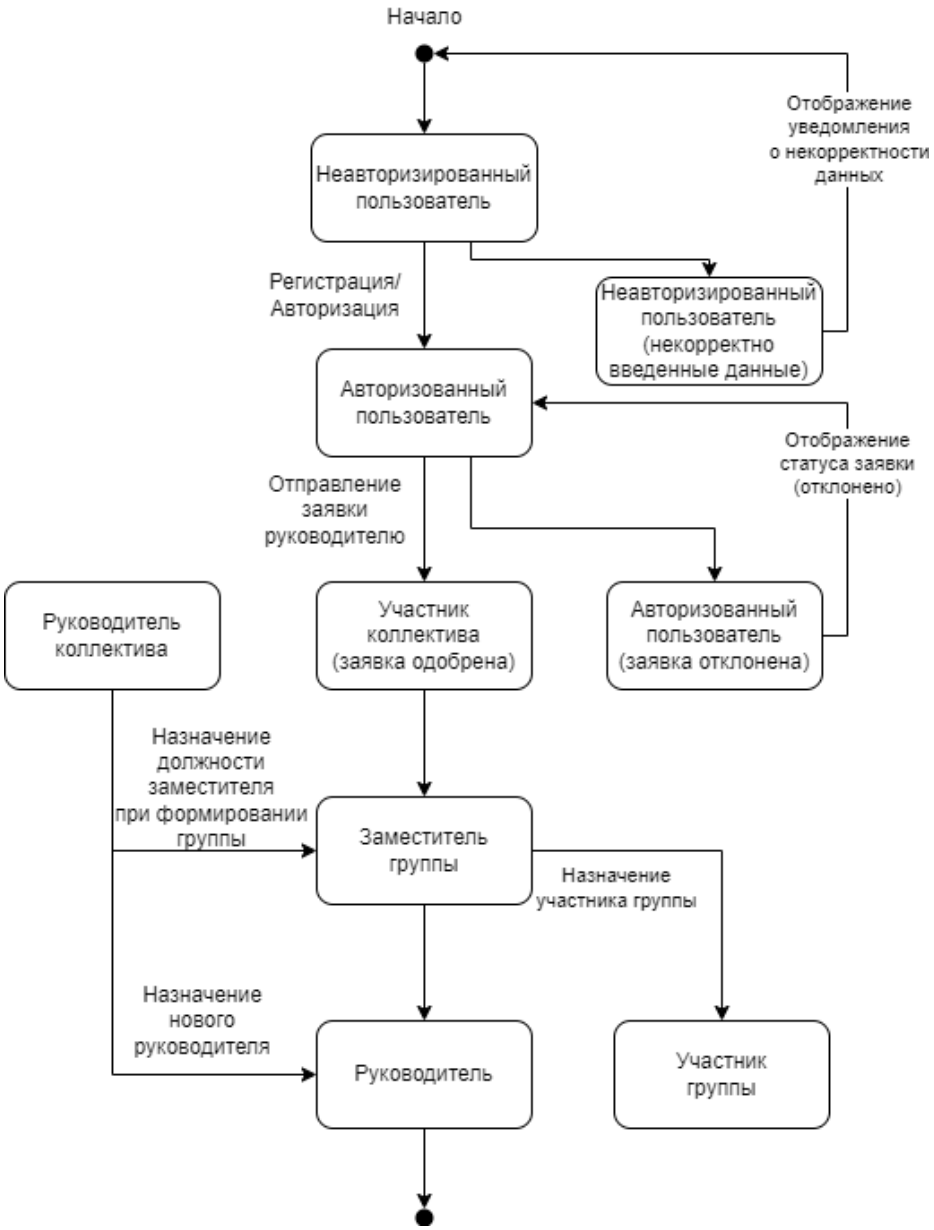

Рисунок 3.2. – Диаграмма состояний пользователя системы

На диаграмме мы можем заметить, что пользователь может находится в 6 состояниях. Все эти состояния пользователь должен проходить, взаимодействуя с существующим музыкальным коллективом, где его могут повысить до заместитель музыкальной группы, и далее – до руководителя. Таким образом в системе обеспечивается разграничение ролей.

Теперь отобразим способы использования с помощью диаграммы вариантов использования для каждой из ролей.

Диаграмма вариантов использования представлена на рисунке 3.3.

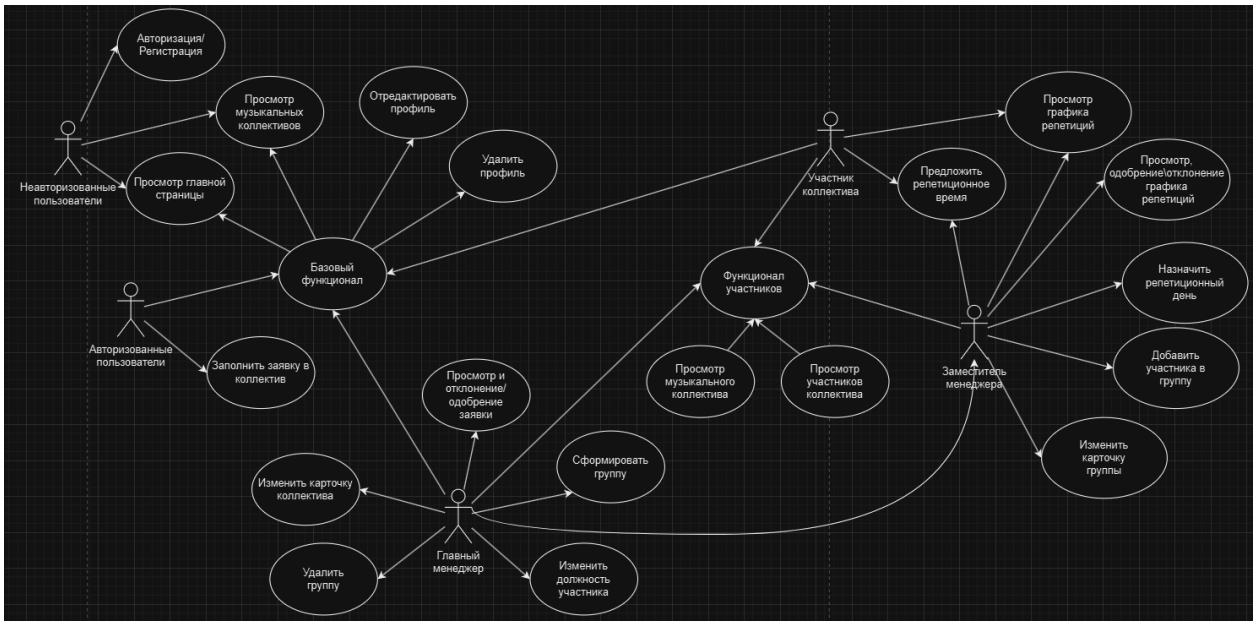

Рисунок 3.3 – Диаграмма вариантов использования

Просмотрев диаграмму, можно определить ряд вариантов использования для каждой роли.

Варианты использования неавторизованного пользователя:

- регистрация в системе;
- авторизация в системе;
- просмотр главной страницы;
- просмотр музыкальных коллективов.

Варианты использования авторизованного пользователя:

- заполнение заявки в коллектив;
- просмотр главной страницы и музыкальных коллективов;
- редактирование и удаление профиля.

Варианты использования участника коллектива:

- всё то же, что и у авторизованного пользователя;
- просмотреть карточку коллектива;
- просмотреть участников коллектива.

Варианты использования участника группы:

- всё то же и у участника коллектива;
- предложение репетиционного времени;
- просмотр графика репетиций;
- просмотр карточки группы/её участников;

Варианты использования заместителя группы:

- всё то же и у участника коллектива;
- изменение/просмотр карточки группы и её участников;
- назначение репетиционного дня;
- просмотр/отклонение/подтверждение репетиций;
- просмотр графика репетиций;

Варианты использования руководителя коллектива:

- всё то же и у заместителя группы;
- удаление коллектива/группы;
- изменение карточки коллектива;
- формирование музыкальной группы.

4 КОДИРОВАНИЕ

4.1 Инструменты разработки

В качестве среды разработки серверной части использовался PyCharm. Для взаимодействия с базой данных использовалась система управления базами данных PostgreSQL посредством программы кроссплатформенного типа PgAdmin.

В свою очередь для разработки клиентской части использовалась среда разработки Visual Studio Code. С целью создания макетов страниц применялся онлайн редактор Figma.

PgAdmin — это программа кроссплатформенного типа для работы с PostgreSQL-серверами. Подключив ПО, пользователи смогут создавать SQL-скрипты, отслеживать процессы и оперировать несколькими БД. Программа ориентирована на работу с PostgreSQL — от создания таблиц до запуска SQL-команд разного уровня сложности [13].

Visual Studio Code (VS Code) — текстовый редактор, разработанный Microsoft для Windows, Linux и macOS. Он позиционируется как «лёгкий» редактор кода для кроссплатформенной разработки веб- и облачных приложений. Включает в себя отладчик, инструменты для работы с Git, подсветку синтаксиса, IntelliSense и средства для рефакторинга. Однако он имеет широкие возможности для кастомизации: пользовательские темы, сочетания клавиш и файлы конфигурации. Распространяется бесплатно, разрабатывается как программное обеспечение с открытым исходным кодом, но готовые сборки распространяются под проприетарной лицензией.

Для серверной части нам также понадобится virtualenv. Виртуальное окружение virtualenv — это инструмент, позволяющий создавать изолированные среды для отдельных проектов Python, решая тем самым проблему зависимостей и совместимости приложений разных версий.

Далее с помощью менеджера пакетов нам потребуется установить все необходимые библиотеки, которые были описаны во втором разделе для серверной и клиентской частей соответственно.

После успешной настройки мы сможем проверить работоспособность нашего проекта, путём хостинга на локальный сервер.

4.2 Описание программной реализации серверной части

Программная реализация серверной части проекта велась с использованием Flask. В начале процесса разработки были определены и сформированы модели данных, отражающие структуру информации, которая будет храниться в базе данных. С учетом особенностей музыкальных коллективов были выделены соответствующие сущности, такие как “Коллектив”, "Группа", "Участник", "Репетиция" и другие, описывающие основные аспекты внутренней деятельности коллективов.

Для работы с базой данных была использована система управления базами данных PostgreSQL, что обеспечило эффективное взаимодействие с данными и поддержку высокого уровня абстракции для запросов и манипуляций с информацией. Определение моделей данных и связей между ними позволило создать структуру базы данных, отражающую логику взаимодействия участников музыкальных коллективов в системе.

Далее использовался Flask для построения API серверной части проекта. С использованием фреймворка были созданы эндпоинты, обеспечивающие CRUD-операции для каждой модели данных. Сериализаторы были использованы для преобразования объектов модели в формат JSON, обеспечивая удобный обмен данными между клиентской и серверной частями приложения. При этом применялись принципы DRY (Don't Repeat Yourself), что способствовало уменьшению дублирования кода и обеспечивало легкость поддержки и расширения.

В ходе разработки серверной части проекта также использовались миксины, которые предоставили возможность использовать и переиспользовать некоторые части функционала в различных частях системы, улучшая структуру кода и обеспечивая его модульность.

Наконец, для управления и администрирования данными в системе была реализована административная панель Flask. Это позволило администраторам и управляющим группами эффективно взаимодействовать с данными, проводить мониторинг и вносить необходимые изменения без необходимости прямого взаимодействия с базой данных. Реализация административной панели обеспечила удобство управления системой, что является важным элементом для обеспечения функциональности проекта.

4.3 Описание программной реализации клиентской части

Как было описано ранее реализация будет представлять из себя интерфейс из 4 основных страниц, контент которых будет зависеть от роли пользователя. Разберём реализацию на примере двух пользователей, опишем ситуацию, когда у нас будет первый пользователь, что создает музыкальный коллектив и в дальнейшим будет являться его руководителем, а второй пользователь будет проходить этап вступления в музыкальную группу первого пользователя.

Начнём со стартовой страницы, она является ознакомительной и в целом представляет из себя переход на 2 страницы: 1. Авторизация .2 Регистрация.

Реализация стартовой страницы представлена на рисунке 4.1.

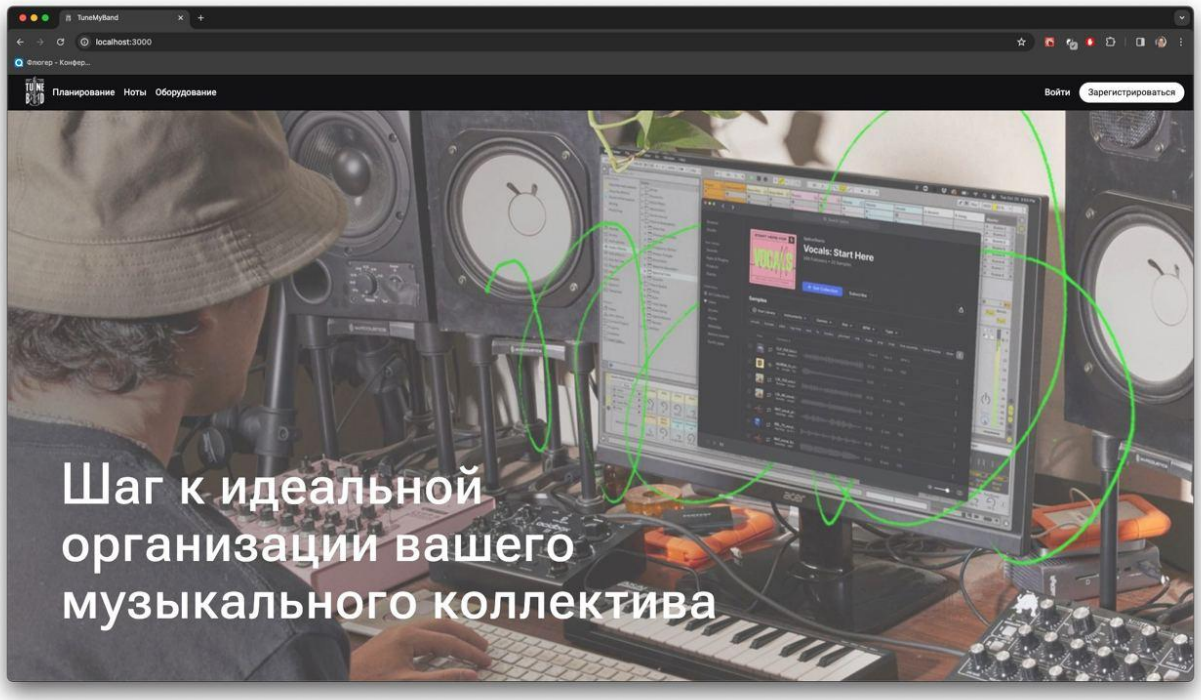

Рисунок 4.1 – Стартовая страница

Страница представляет из себя 2 основных блока:

- Header (является основным компонентом переходов);
- Главный слайд (ознакомительная информация);

Следующей страницей является страница либо же авторизация, либо же регистрация. Главное их отличие, собственно, в том, что если пользователь уже имеет аккаунт в нашей системе, то с помощью почтового ящика и пароля, что он указывал при регистрации, может спокойно перейти на следующий этап, просто введя эти данные и нажать кнопку “Войти”, пример данного этапа представлен на рисунке 4.2. Страница же регистрации позволяет создать аккаунт пользователю нашей системы, указал лишь адрес электронного почтового ящика и пароля, то есть секретной строки, который будет знать только он, эта страница представлена на рисунке 4.3.

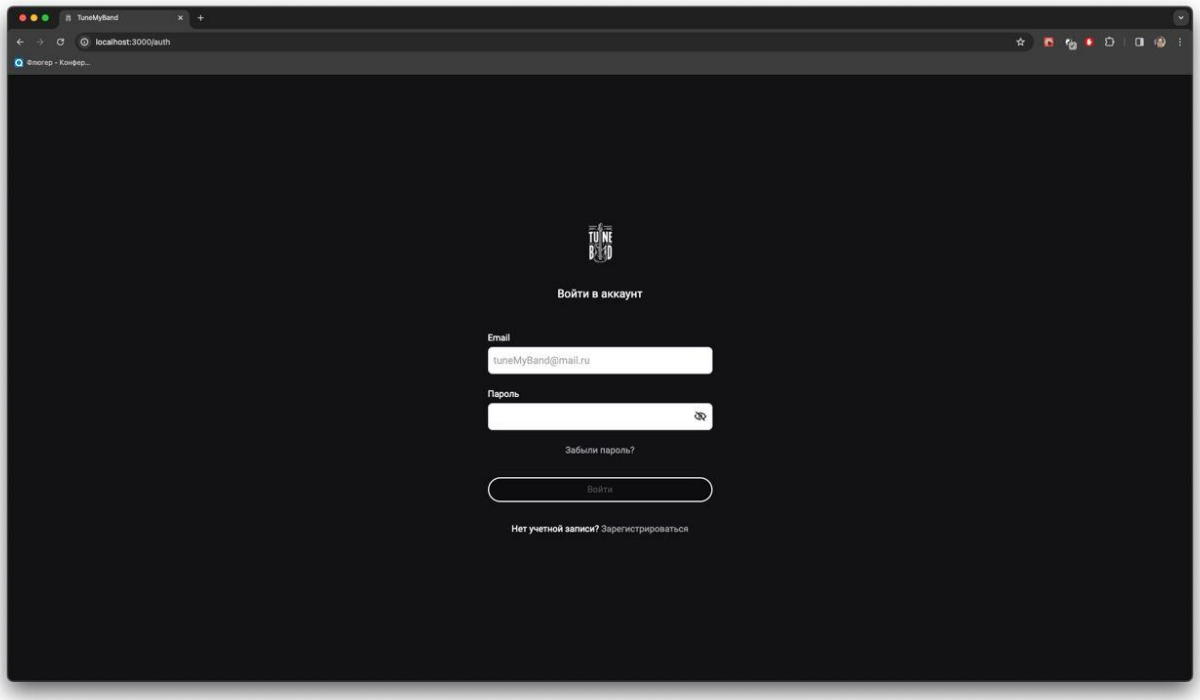

Рисунок 4.2 –Страница авторизации

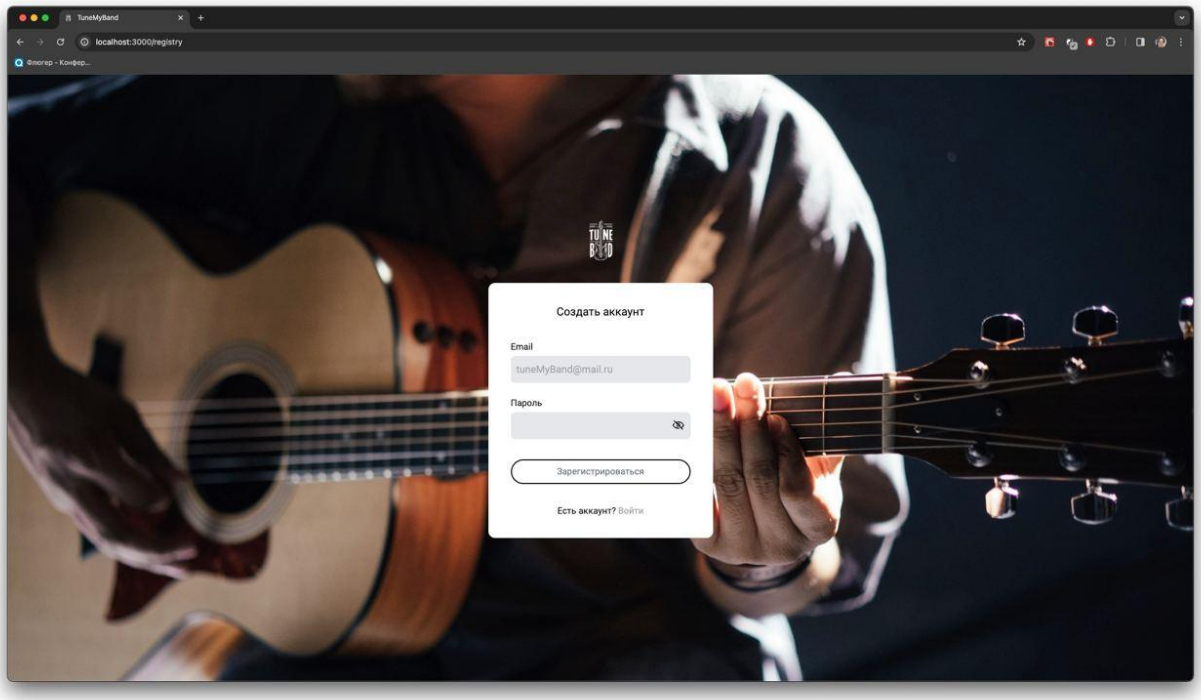

Рисунок 4.3 –Страница регистрации

После того, как пользователь прошел регистрацию/авторизацию, он попадет на страницу информации о профиле, в котором может изменить имя, фамилию и номер телефона, также она содержит информацию о всех музыкальных коллективах, в которых этот пользователь участвует, и

информацию о заявках на вступление в коллективы. Страница информации о профиле представлена на рисунке 4.4.

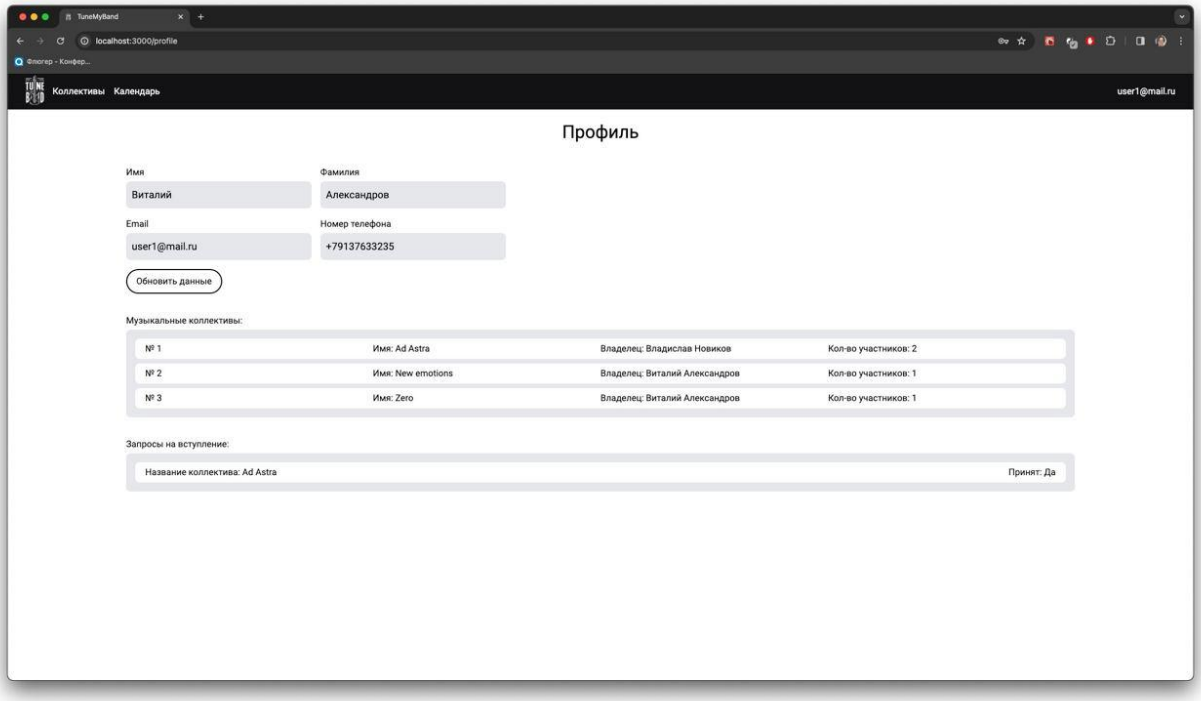

Рисунок 4.4 –Страница информации о профиле

Если же мы, допустим хотим выйти из аккаунта, то при нажатии на свой электронный почтовый адрес, будет открыто выпадающее меню, в котором присутствуют 2 кнопки: Профиль – собственно, перейти на страницу профиля аккаунта, и Выйти – выход из аккаунта.

После этого пользователь может перейти на страницу музыкальных коллективов, по навигации в header компоненте. Где нас встречает страница с 3 вариантами действий, которые при нажатии на нужное открывает отдельную функциональность на странице: Все – увидеть список всех музыкальных коллективов, и отправить запрос на присоединение, Участие – увидеть список всех музыкальных коллективов, в которых ты состоишь независимо от роли, Создание – создать музыкальный коллектив. Страница создания музыкальной группы представлена на рисунке 4.5. Страница со списком всех музыкальных коллективов, то есть вкладка “Все” представлена на рисунке 4.6. Также на рисунке 4.7 представлена ситуация, когда пользователь отправил запрос на присоединение к музыкальному коллективу, у него откроется окно с

описанием о статусе операции и в дальнейшем на странице профиля он сможет увидеть список своих заявок.

Когда пользователь создал свой музыкальный коллектив, то на вкладке Участие он с помощью кнопки может перейти на страницу музыкального коллектива, где в зависимости от роли в ней ему будет доступен конкретный функционал. А полный функционал – это изменение имени коллектива, исключение участников, а также ответ на запросы пользователей о вступлении в коллектив. Весь этот функционал представлен на рисунках 4.8-4.10.

Далее, после того как музыкальный коллектив набран и в нём присутствуют участники, мы можем создать группы, почти тот же функционал что и у музыкального коллектива, но внутри него. В зависимости далее от этих групп мы будем выставлять репетиционные дни. Страница календаря, где заместитель руководителя назначает репетиционные дни представлена на странице 4.11. А также на рисунке 4.12 представлена страница календаря обычного пользователя, что является участником 2-х музыкальных коллективов, и он видит у себя репетиционные дни, которые назначили эти самые коллективы.

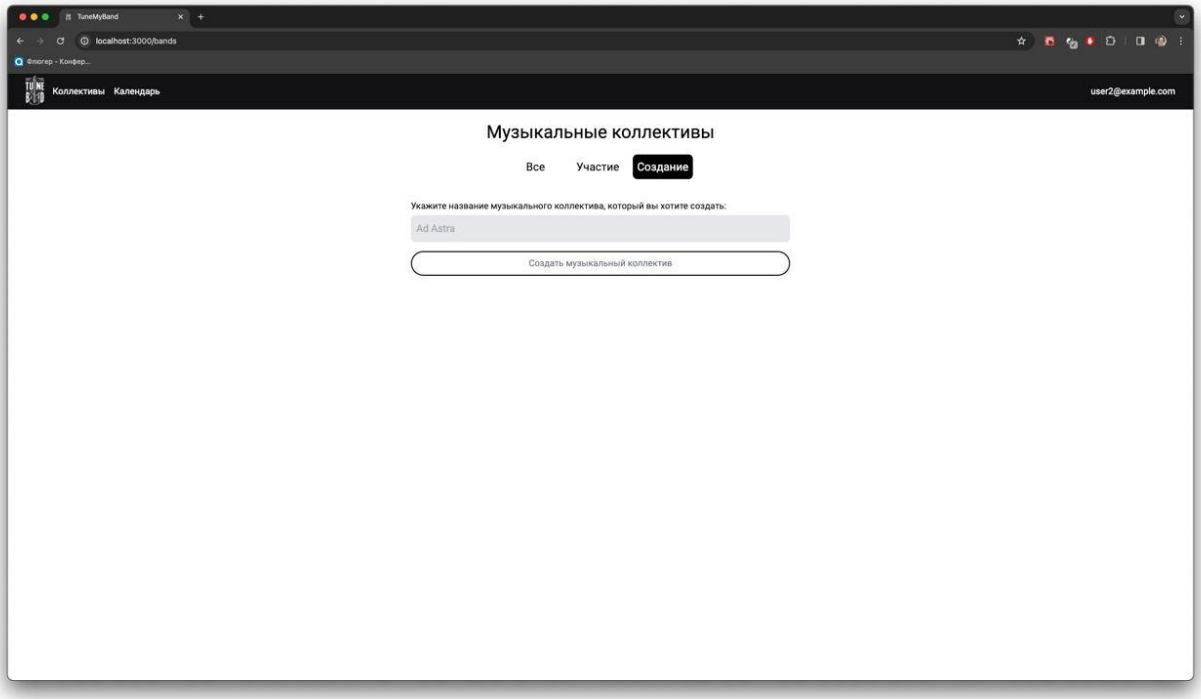

Рисунок 4.5 –Страница создания музыкального коллектива

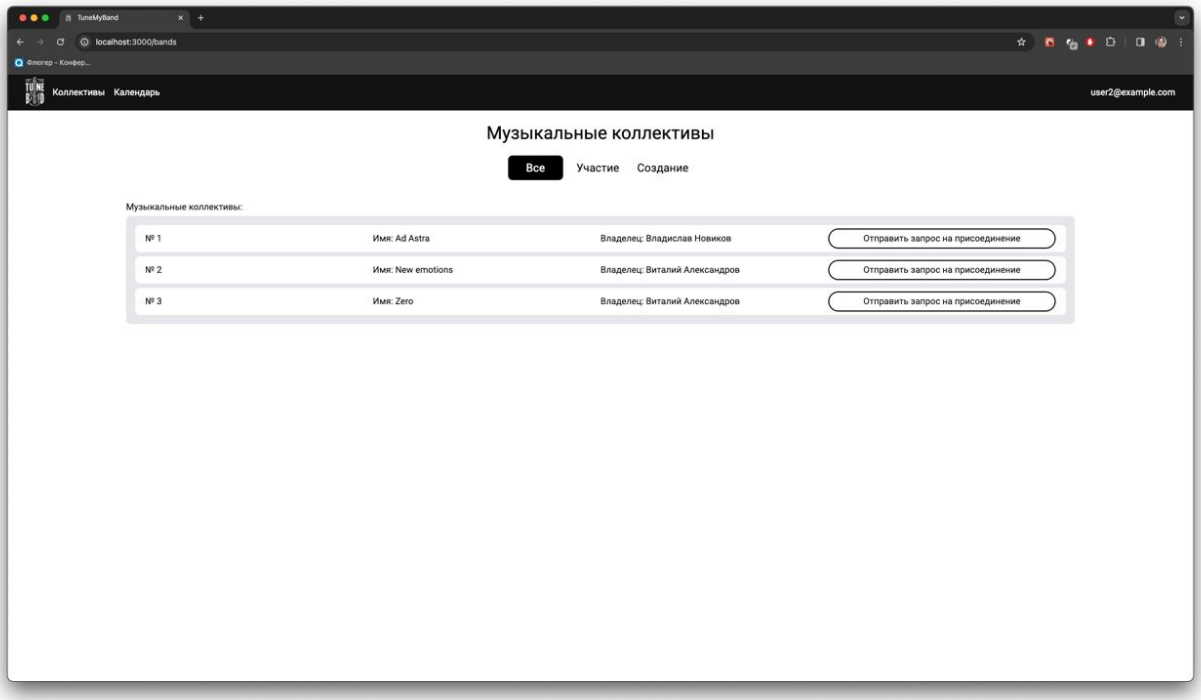

Рисунок 4.6 –Страница с записью всех музыкальных коллективов

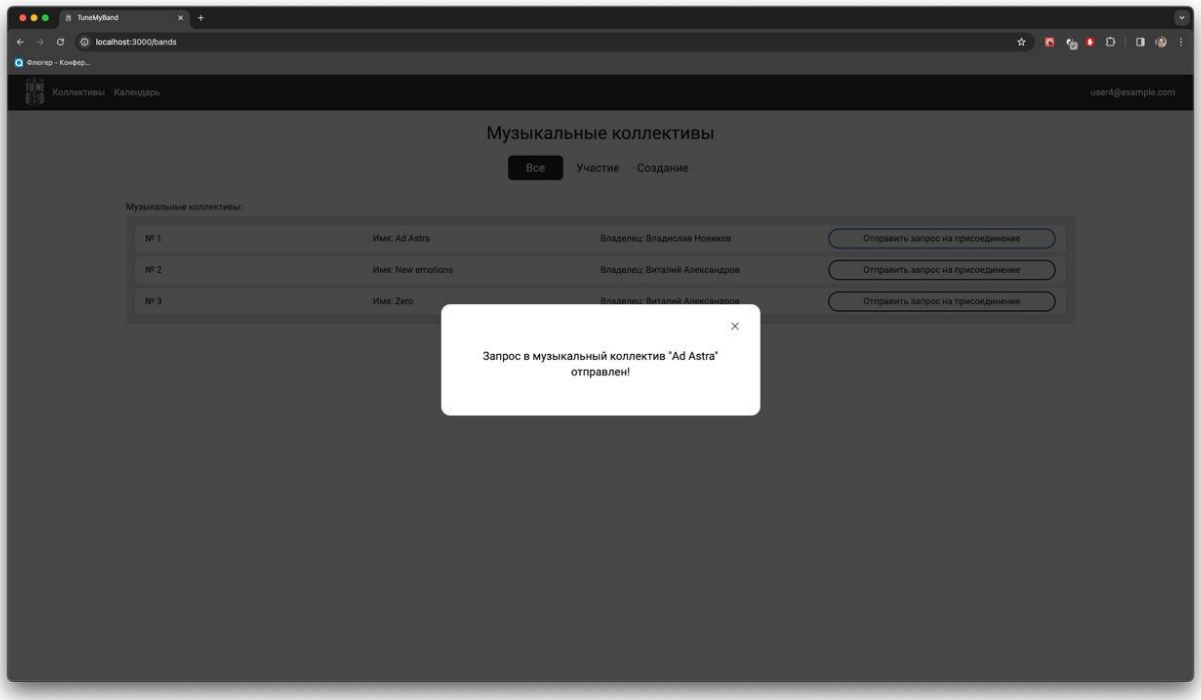

Рисунок 4.7 –Страница отправки запроса в музыкальный коллектив

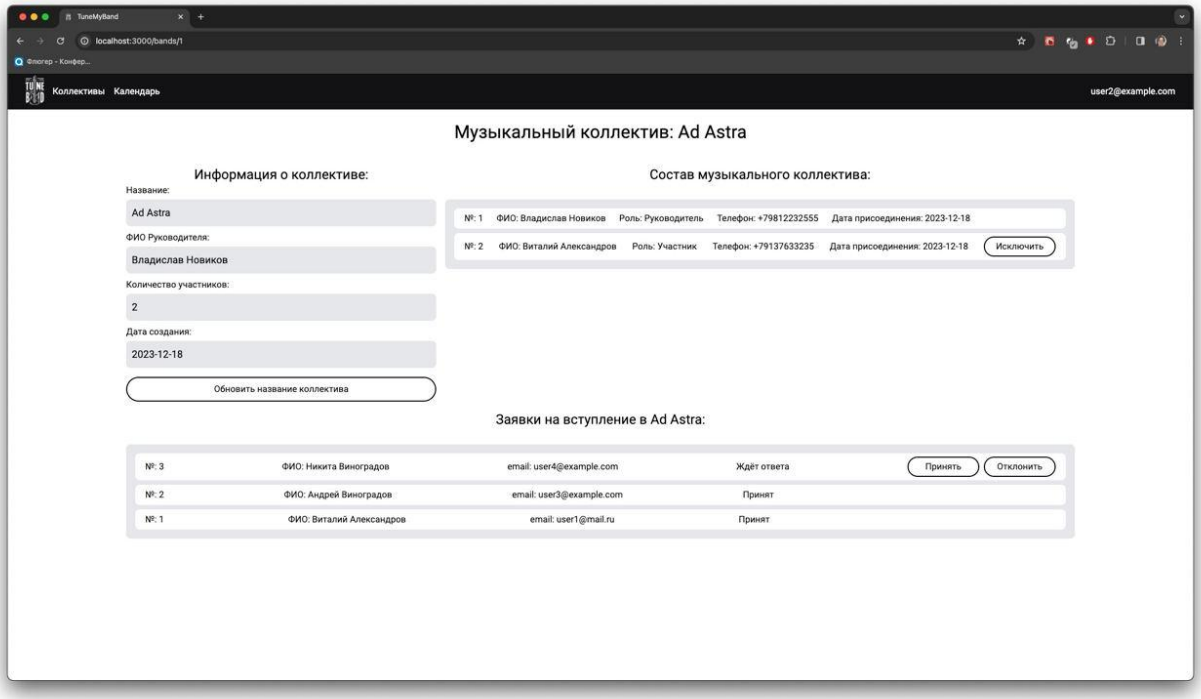

Рисунок 4.8 –Страница музыкального коллектива от лица её руководителя

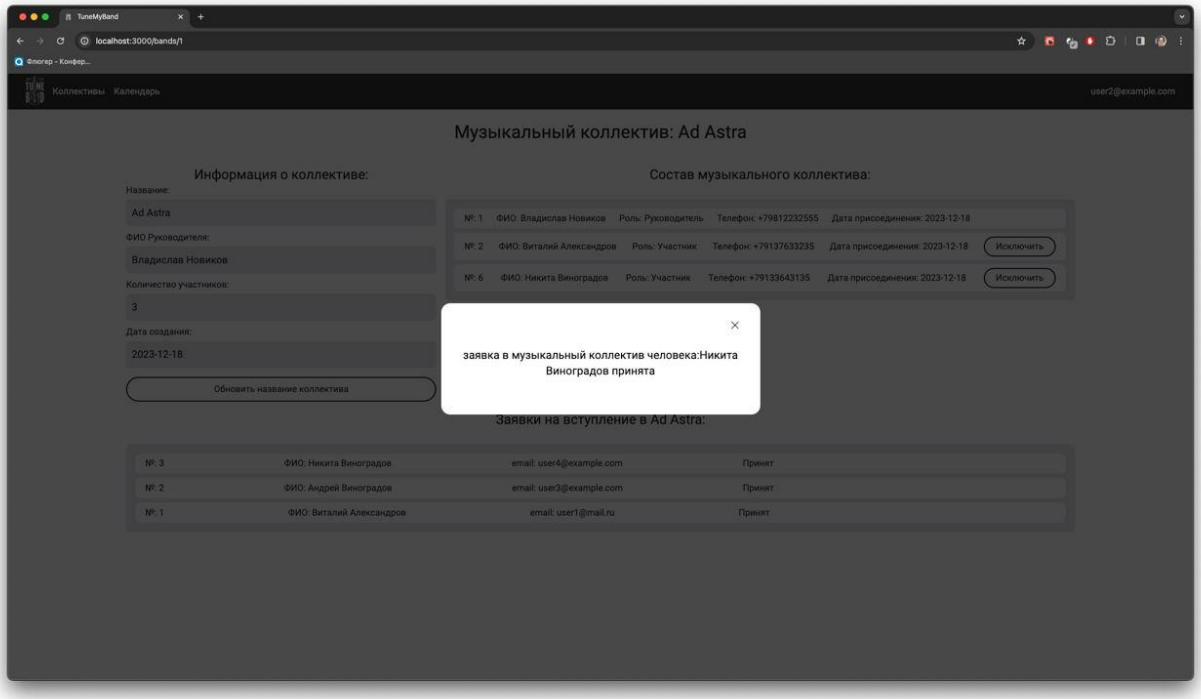

Рисунок 4.9 –Статус принятия заявки на вступление в коллектив

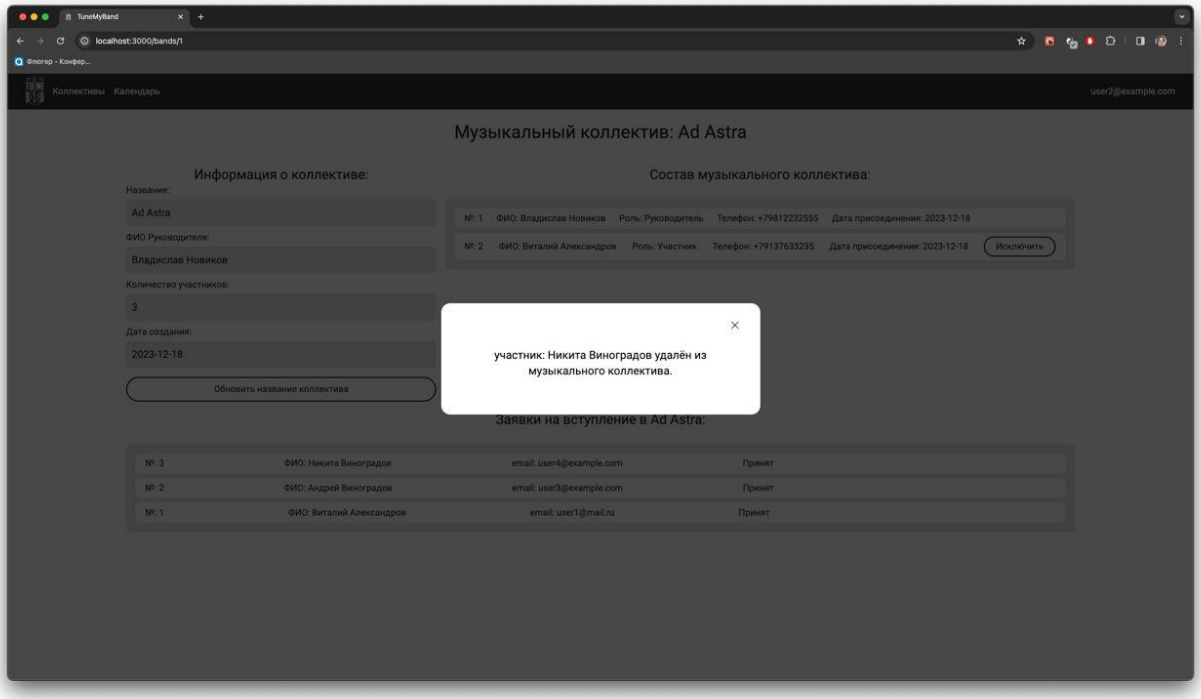

Рисунок 4.10 –Статус исключения из музыкального коллектива

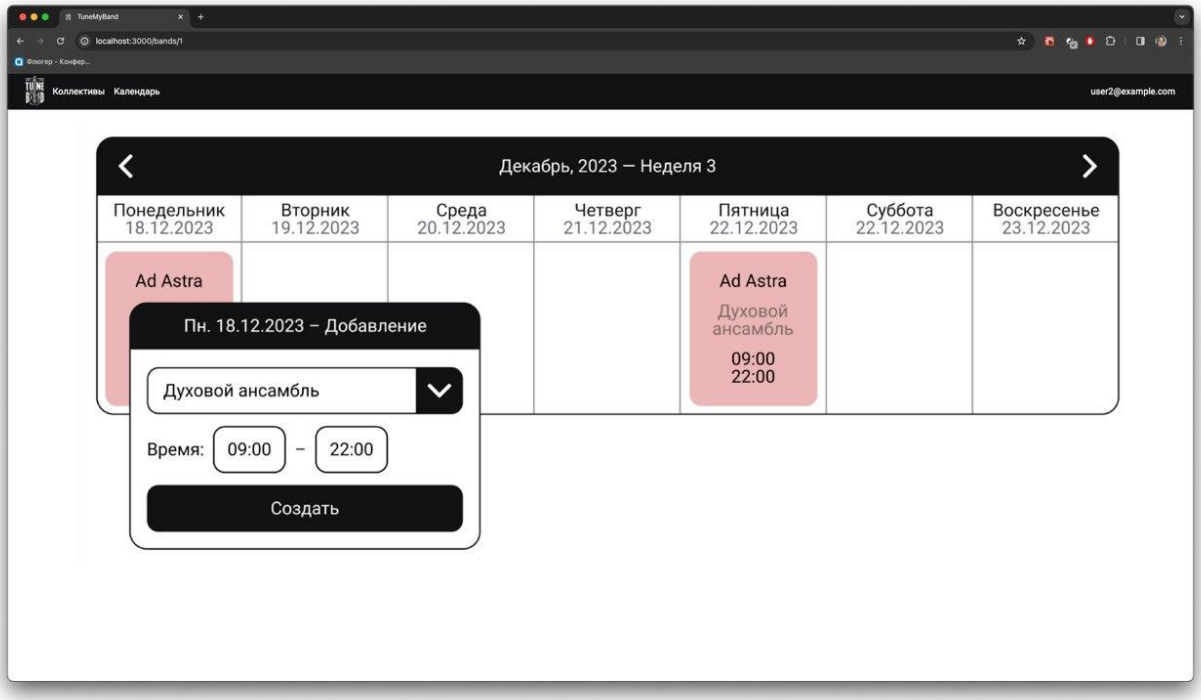

Рисунок 4.11 –Страница календаря от лица заместителя руководителя коллектива

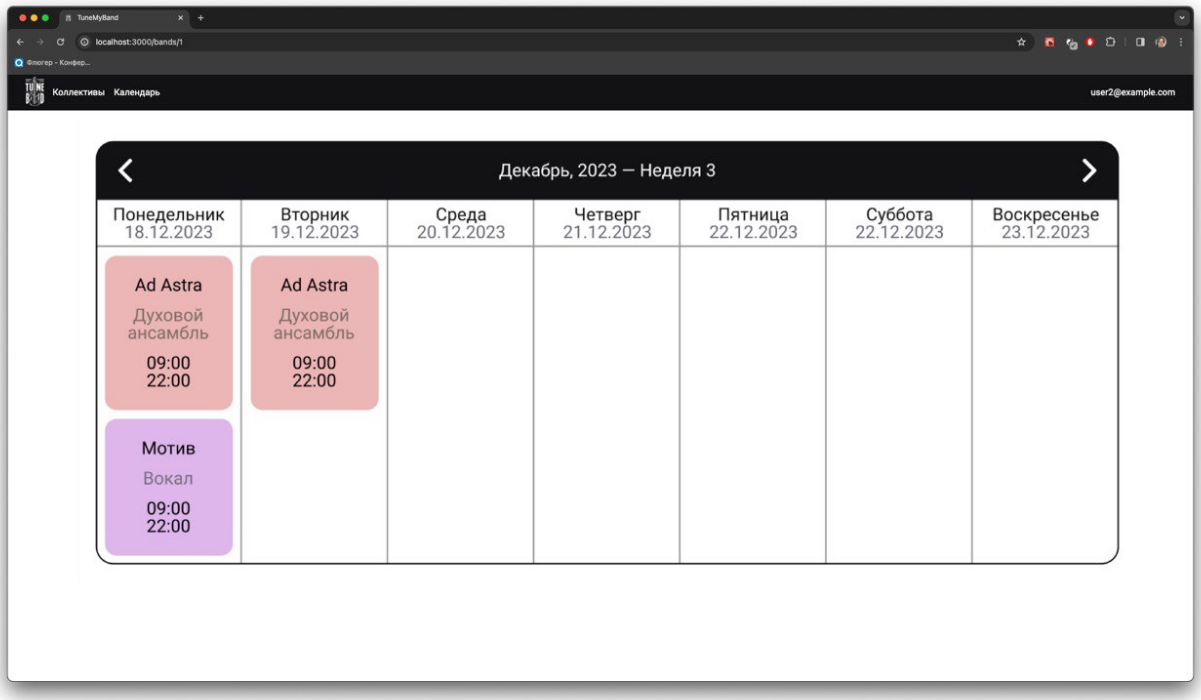

Рисунок 4.12 –Страница календаря от лица обычного пользователя

Заключение

В результате выполненной работы по созданию системы управления деятельностью музыкальных коллективов были успешно достигнуты поставленные задачи. Инициально определены роли участников системы, выраженные в bpmn диаграммах, где каждый этап бизнес-процесса был визуализирован. Обобщенная bpmn диаграмма объединила все этапы, четко демонстрируя взаимодействие всех участников.

Сформулированный перечень требований к системе стал основой для проектирования Use-cases и диаграммы вариантов использования, обеспечивая четкое понимание функциональности для каждой роли в системе. Этот этап способствовал упрощению задач для команды и оптимизации времени на следующих этапах разработки.

В целом, проект демонстрирует достаточный функциональности и эффективности, а также соответствует современным технологиям разработки, таким как TypeScript для клиентской части. Важно отметить также использование Flask, PostgreSQL в серверной части, что обеспечивает надежность и согласованность всей системы.

Список используемой литературы

1. Описание бизнес-процессов Как есть (AS IS) и Как должно быть (TO BE) [Электронный ресурс]. — Режим доступа: https://www.trinion.org/blog/opisanie-biznes-processov-kak-est-as-is-i-kak-dolzhno-byt-to-be (дата обращения: 16.11.2024).
2. Что такое BPMN [Электронный ресурс]. — Режим доступа: https://www.lucidchart.com/pages/ru/bpmn (дата обращения: 16.11.2024).
3. Как устроена каскадная модель управления проектами [Электронный ресурс]. — Режим доступа: https://kachestvo.pro/kachestvo-upravleniya/proektnoe-upravlenie/kak-ustroena-kaskadnaya-model-upravleniya-proektami/ (дата обращения: 16.11.2024).
4. TypeScript [Электронный ресурс]. — Режим доступа: https://www.typescriptlang.org/ (дата обращения: 16.11.2024).
5. Язык гипертекстовой разметки [Электронный ресурс]. — Режим доступа: http://htmlbook.ru/html (дата обращения: 12.11.2024).
6. CSS [Электронный ресурс]. — Режим доступа: https://developer.mozilla.org/ru/docs/Learn/Getting_started_with_the_web/CSS_basics (дата обращения: 14.11.2024).
7. Axios [Электронный ресурс]. — Режим доступа: https://rabota-naiti.ru/karera/trudoustroystvo/poryadok.html (дата обращения: 14.11.2024).
8. Flask [Электронный ресурс]. – Режим доступа: https://flask.palletsprojects.com/en/stable/ (дата обращения: 16.11.2024).
9. DRF [Электронный ресурс]. – Режим доступа: https://www.django-rest-framework.org/ (дата обращения: 09.11.2024).
10. PgAdmin [Электронный ресурс]. – Режим доступа: https://pgadmin.ru/ (дата обращения: 18.11.2024).
11. VSCode [Электронный ресурс]. – Режим доступа: https://code.visualstudio.com/ (дата обращения: 18.11.2024).