Федеральное государственное автономное

образовательное учреждение

Высшего образования

«СИБИРСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Институт космических и информационных технологий

институт

Кафедра вычислительной техники

кафедра

УТВЕРЖДАЮ

Заведующий кафедрой

_____ О.В. Непомнящий

«____» _____2022 г.

БАКАЛАВРСКАЯ РАБОТА

09.03.01 – Информатика и вычислительная техника

код – наименование направления

Приложение с аккордами песен на платформе VK MiniApps

тема

Руководитель _____ В. С. Васильев

подпись, дата

Выпускник _____ А. А. Свистунов

подпись, дата

Нормоконтролер _____ В. С. Васильев

подпись, дата

Красноярск 2022

Федеральное государственное автономное

образовательное учреждение

Высшего образования

«СИБИРСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Институт космических и информационных технологий

институт

Кафедра вычислительной техники

кафедра

УТВЕРЖДАЮ

Заведующий кафедрой

_____ О.В. Непомнящий

«____» _____2022 г.

ЗАДАНИЕ
НА ВЫПУСКНУЮ КВАЛИФИКАЦИОННУЮ РАБОТУ
в форме бакалаврской работы

Студенту Свистунову Александру Андреевичу

Группа КИ18-08Б

Направление 09.03.01 Информатика и вычислительная техника

Тема выпускной работы: Приложение с аккордами песен на платформе VK MiniApps

Утверждена приказом по университету №7914/С от 26.05.22

Руководитель ВКР: В.С. Васильев, магистр техники и технологий

Исходные данные для ВКР: Информация о компьютерной периферии.

Перечень разделов ВКР:

1. Разработка спецификации требований.
2. Проектирование.
3. Реализация и документация.

Перечень графического материала: нет

Руководитель _____ В. С. Васильев
подпись, дата

Задание принял к исполнению _____ А. А. Свистунов
подпись, дата

«____» _____2021 г.

РЕФЕРАТ

Выпускная квалификационная работа по теме “Приложение с аккордами для песен на платформе VK MiniApps” содержит 44 страницы текстового документа, 29 рисунков, 16 использованных источников, 1 таблицу.

АККОРДЫ, КЛИЕНТ-СЕРВЕР, VK MINIAPPS, API, HTTP.

Цель работы: создание приложения для платформы VK MiniApps, представляющего собой каталог текстов песен с аккордами, с возможностью сохранять песни в песенник, транспонировать аккорды, делиться песнями с друзьями, а также с функцией поиска песен по всему каталогу.

В результате выполнения ВКР было разработано приложение «Guitarly».

В выпускную квалификационную работу входит введение, 3 главы и заключение.

Во введении ставится цель и выполняется ее декомпозиция на задачи.

В первой главе рассматриваются аналоги, формулируются основные требования к разрабатываемому приложению.

Во второй главе проектируются архитектура системы, структура базы данных, алгоритм выделения аккорда на тексте песни.

В третьей главе описана реализация основных элементов приложения, написана инструкция для разработчика по развертыванию окружения для разработки, приведены результаты тестирования.

В заключении подводятся итоги по выполненной работе.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

У гитаристов часто возникает такая проблема, что они не знают подходящих аккордов к какой-либо песне. Традиционно она решается песенниками, в которые записываются текст и аккорды. Раньше песенники были бумажными, а сейчас с лёгкостью можно найти аккорды к любой песне с помощью Интернета, будь то веб-сайт или мобильное приложение. Платформа VK MiniApps позволяет создать приложение, обладающее как достоинствами веб-сайта, так и сильными сторонами мобильного приложения.

Целью работы является создание приложения для платформы VK MiniApps, представляющего собой каталог текстов песен с аккордами, с возможностью сохранять песни в песенник, транспонировать аккорды, делиться песнями с друзьями, а также с функцией поиска песен по всему каталогу.

Структура работы отражает решаемые **задачи**:

- в первой главе приведены:
  а) анализ сильных и слабых сторон аналогов;
  б) спецификация требований к разрабатываемой системе;
- во второй главе работы разрабатываются: архитектура системы, структура базы данных, алгоритм выделения аккордов из текста;
- в третьей главе описаны особенности реализации серверной и клиентской частей, приведены инструкции по сборке и развертыванию системы.

1 Анализ задания

Необходимо разработать приложение для гитаристов, представляющее собой каталог текстов песен с аккордами, на платформе VK MiniApps. Для обеспечения конкурентоспособности приложения, выполнен анализ сильных и слабых сторон аналогичных приложений. С их учетом составлены спецификации требований к серверной и клиентской частям на разработку.

1.1 Анализ существующих аналогов

По запросу в Google «тексты песен с аккордами» выдается несколько сайтов, которые и являются аналогами разрабатываемому приложению. В таблице 1 приведено сравнение трех самых популярных сайтов с аккордами: amdm.ru [1], hm6.ru [2] и akkordam.ru [3].

Таблица 1 — Сравнение аналогов

	Есть каталог артистов	Есть возможность менять тональность	Есть песенник	Современный дизайн	Есть поиск
amdm.ru	Да	Да	Да	Нет	Да
hm6.ru	Да	Да	Да	Да	Да
akkordam.ru	Нет	Нет	Нет	Нет	Нет

Из таблицы 1 можно сделать вывод, что сайт akkordam.ru проигрывает своим аналогам по всем позициям, но в то же время он занимает одну из топовых строчек при поиске в Google, а значит люди им пользуются. Сайт amdm.ru также является одним из самых популярных и имеет все необходимые гитаристам функции, но у него устаревший и неудобный дизайн. Ещё у этого сайта есть проблема с дубликатами песен, что создает проблемы при поиске. Сайт hm6.ru является самым удобным и функциональным из всех рассмотренных аналогов, но он заблокирован на территории РФ за нарушение авторских прав, поэтому

обычные пользователи не могут им пользоваться без специальных средств обхода блокировок.

Исходя из анализа сильных и слабых сторон аналогов определены основные функции, которые должно иметь приложение:

- каталог артистов;
- каталог песен;
- поиск по базе данных;
- возможность менять тональность;
- добавление и удаление песен из песенника.

Главным отличием от аналогов будет то, что приложение разрабатывается под платформу VK MiniApps и не имеет аналогов на ней.

1.2 Платформа VK MiniApps

Разрабатываемое приложение должно состоять из клиентской и серверной частей. Нативное приложение ВКонтакте отображает в WebView или в iframe клиентское приложение, которое делает запросы к серверу (API). На рисунке 1.1 изображена схема работы с платформой VK MiniApps. При этом, ВКонтакте никак не участвует в бизнес-логике нашего приложения, а лишь является точкой входа для пользователей.

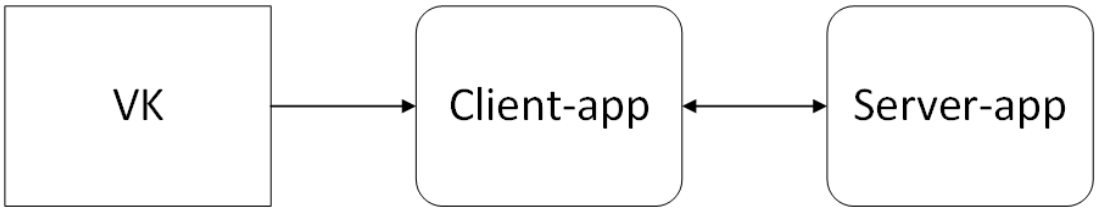

Рисунок 1.1 — Схема работы с платформой VK MiniApps

1.3 Спецификация требований на клиентское приложение

1.3.1 Функциональные требования

На рисунке 1.2 приведена диаграмма прецедентов создаваемого приложения. Наполнение приложения данными не будет доступно обычным пользователям, поэтому нужно выделить отдельную роль для администратора.

Рисунок 1.2 — Диаграмма вариантов использования приложения

Прецедент 1. Просмотр песни.

Цель сценария: Увидеть текст песни с аккордами.

Предусловия: Пользователь открыл песню по ссылке или перешел на неё из каталога.

Основной сценарий:

Отправляется запрос на сервер, возвращается объект с информацией о песне и об артисте.

Условия ввода в действие альтернативных сценариев

Условие 1. Сервер изначально недоступен.

Выводится сообщение об ошибке.

Условие 2. Песня не найдена.

Перенаправление на главную страницу.

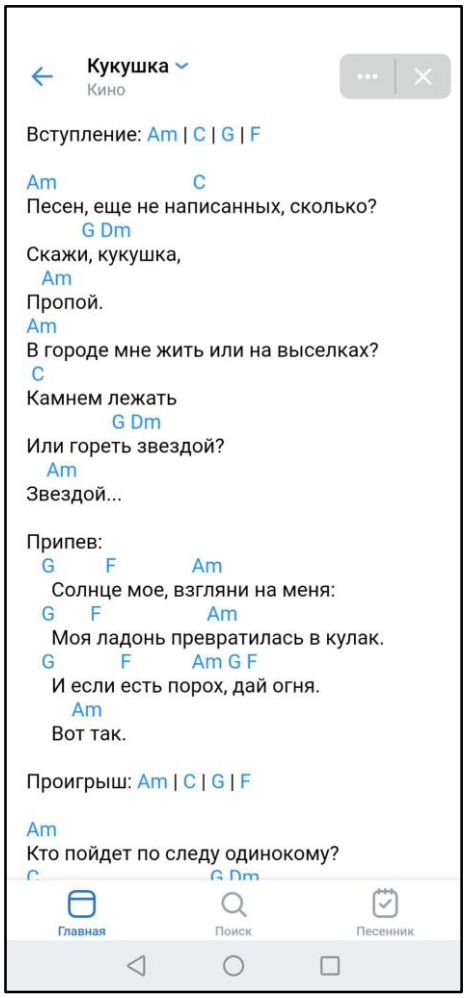

Рисунок 1.3 — Страница просмотра песни

Прецедент 2. Добавить в песенник.

Цель сценария: Добавить песню в песенник для быстрого доступа к ней.

Предусловия: Пользователь находится в окне “Просмотр песни”, открыл выпадающее меню в шапке, нажал кнопку “Добавить в песенник”.

Основной сценарий:

Отправляется запрос на сервер на изменение статуса песни. Изменяется текст кнопки на “Удалить из песенника” и меняется иконка.

Условия ввода в действие альтернативных сценариев

Условие 1. Сервер изначально недоступен.

Выводится сообщение об ошибке.

Прецедент 3. Удалить песню из песенника.

Цель сценария: Удаление песни из песенника.

Предусловия: Пользователь находится в окне “Просмотр песни”, открыл выпадающее меню в шапке, нажал кнопку “Удалить из песенника”.

Основной сценарий:

Отправляется запрос на сервер на изменение статуса песни. Изменяется текст кнопки на “Добавить в песенник” и меняется иконка.

Условия ввода в действие альтернативных сценариев

Условие 1. Сервер изначально недоступен.

Выводится сообщение об ошибке.

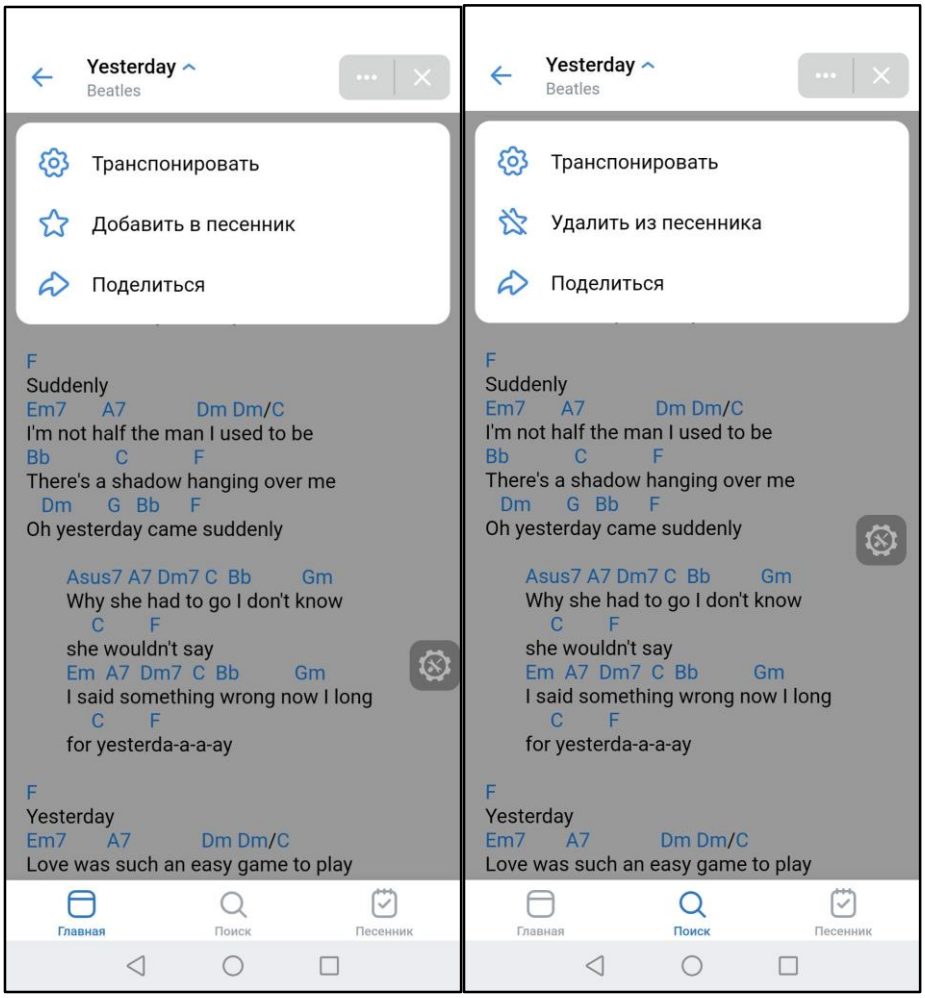

Рисунок 1.4 — Добавить в песенник / удалить из песенника

Прецедент 4. Поделиться песней.

Цель сценария: Поделиться песней с друзьями.

Предусловия: Пользователь находится в окне “Просмотр песни”, открыл выпадающее меню в шапке, нажал кнопку “Поделиться”.

Основной сценарий:

Открывается модальное окно с двумя вариантами: “В личном сообщении” и “На своей стене”. Используются функции, которые предоставляет платформа VK Mini Apps: *VKWebAppShowWallPostBox* и *VKWebAppShare*.

A. При выборе “В личном сообщении” открывается список всех диалогов с возможностью выбрать кому отправить ссылку.

B. При выборе “На своей стене” открывается окно подтверждения публикации.

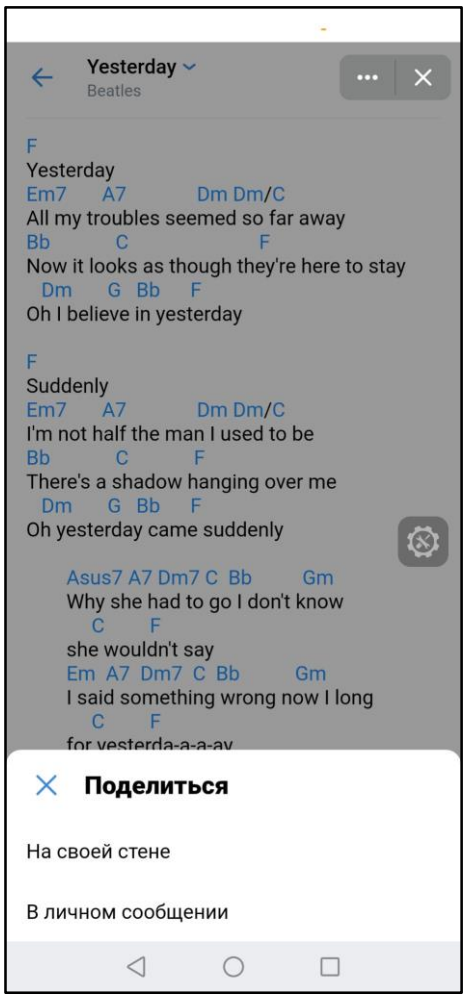

Рисунок 1.5 — Модальное окно “Поделиться”

Прецедент 5. Изменить тональность.

Цель сценария: Транспонировать аккорды (поменять тональность), чтобы было удобнее играть и петь.

Предусловия: Пользователь находится в окне “Просмотр песни”, открыл выпадающее меню в шапке, нажал кнопку “Транспонировать”.

Основной сценарий:

Открывается модальное окно с двумя кнопками: “-” и “+” (понижение и повышение тональности соответственно). При нажатии на них, на сервер отправляется запрос с новым тоном и в ответ приходит текст песни с аккордами в новой тональности. Пользователю показывается новый текст.

Условия ввода в действие альтернативных сценариев

Условие 1. Сервер изначально недоступен.

Выводится сообщение об ошибке.

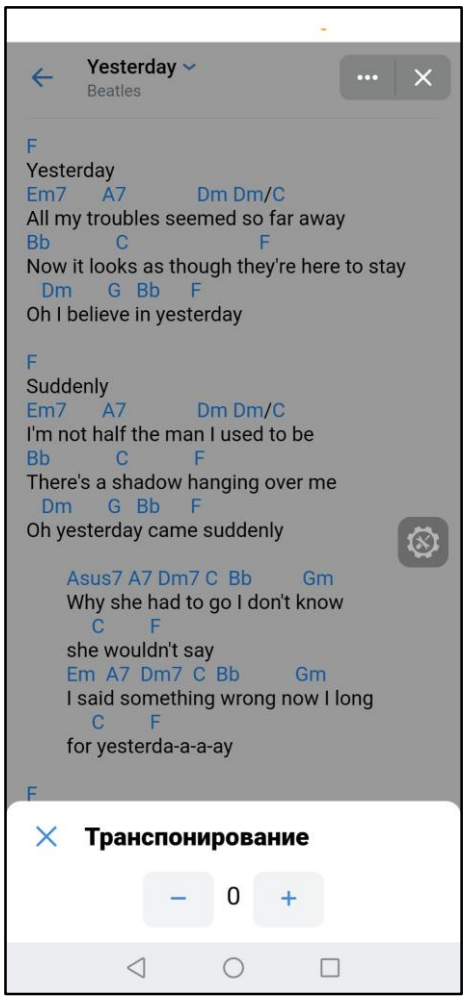

Рисунок 1.6 — Модальное окно “Транспонирование”

Прецедент 6. Просмотр артиста.

Цель сценария: Посмотреть все песни конкретного артиста.

Предусловия: Пользователь находится в одном из окон: “Главная страница”, “Просмотр песни”, “Поиск”. Нажата кнопка с фотографией артиста.

Основной сценарий:

Открывается страница с фотографией артиста и списком всех песен.

Условия ввода в действие альтернативных сценариев

Условие 1. Сервер изначально недоступен.

Выводится сообщение об ошибке.

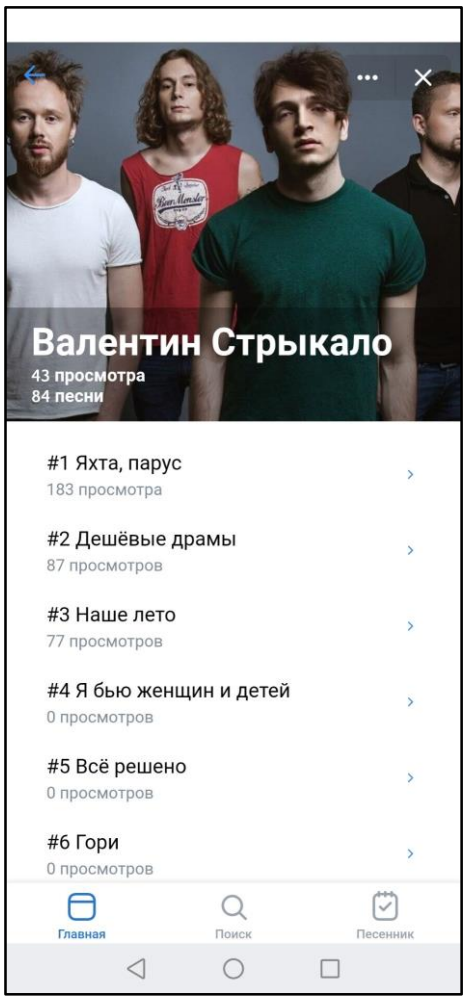

Рисунок 1.7 — Страница артиста

Прецедент 7. Поиск.

Цель сценария: Найти нужную песню или артиста.

Предусловия: Пользователь нажал кнопку “Поиск” в таббаре.

Основной сценарий:

Пользователь вводит какой-то текст, на сервер отправляется запрос со строкой поиска. Сервер возвращает те песни и тех артистов, в названиях которых есть введенный пользователем текст.

Условия ввода в действие альтернативных сценариев

Условие 1. Сервер изначально недоступен.

Выводится сообщение об ошибке.

Условие 2. Ничего не найдено.

Пользователю выводится сообщение о том, что ничего не найдено, и предлагается изменить запрос.

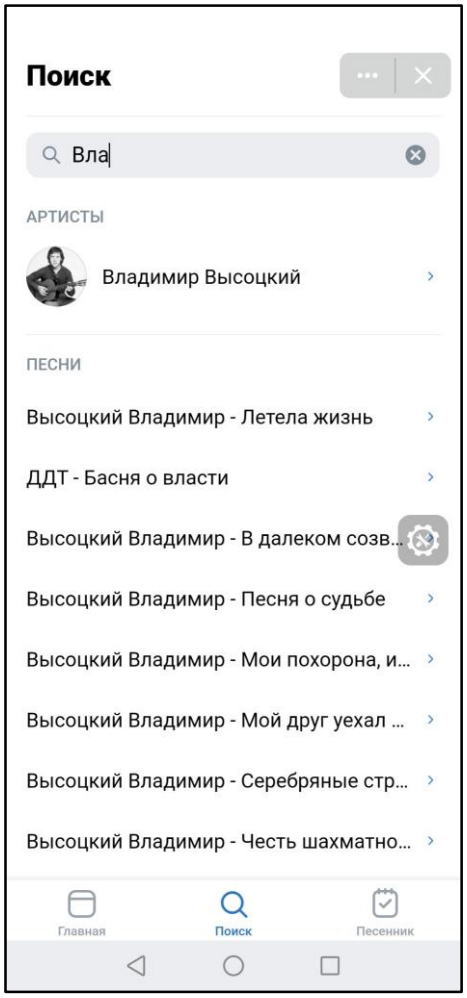

Рисунок 1.8 — Поиск

Прецедент 8. Просмотр песенника.

Цель сценария: Посмотреть все песни, которые пользователь добавил в песенник.

Предусловия: Пользователь нажал кнопку “Песенник” в таббаре.

Основной сценарий:

Открывается страница со списком всех песен, которые пользователь добавил в песенник.

Условия ввода в действие альтернативных сценариев

Условие 1. Сервер изначально недоступен.

Выводится сообщение об ошибке.

Условие 2. Песенник пуст.

Выводится сообщение о том, что песенник пуст.

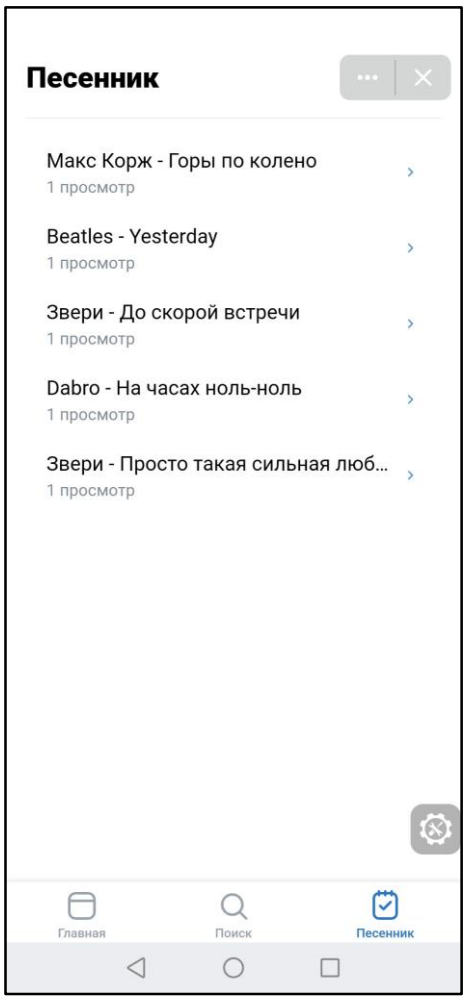

Рисунок 1.9 — Песенник

Прецедент 9. Редактировать песню.

Цель сценария: Отредактировать текст или название песни.

Предусловия: Администратор находится в окне “Просмотр песни”, открыл выпадающее меню в шапке, нажал кнопку “Редактировать”.

Основной сценарий:

Открывается страница с формой с заполненными тремя полями: Название, Полное название и Текст песни. После завершения редактирования администратор нажимает кнопку “Сохранить”, отправляется запрос на сервер, администратор перенаправляется на страницу “Просмотр песни”.

Условия ввода в действие альтернативных сценариев

Условие 1. Сервер изначально недоступен.

Выводится сообщение об ошибке.

Условие 2. Ошибки валидации.

Поле с ошибкой подсвечивается красным цветом, показывается сообщение об ошибке.

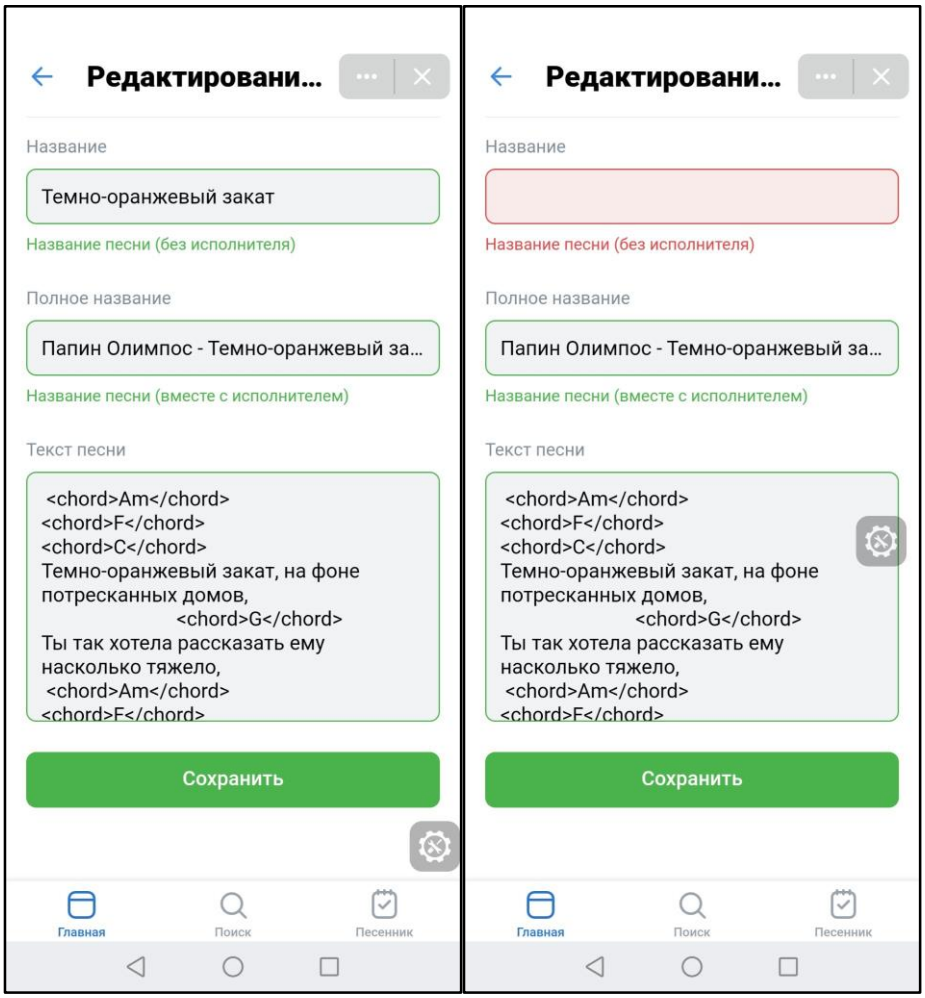

Рисунок 1.10 — Страница редактирования песни

Прецедент 10. Удалить песню.

Цель сценария: Удалить песню из каталога.

Предусловия: Администратор находится в окне “Просмотр песни”, открыл выпадающее меню в шапке, нажал кнопку “Удалить”.

Основной сценарий:

Открывается модальное окно с подтверждением. Если администратор подтверждает удаление песни, то отправляется запрос на сервер и, после успешного ответа, песня удаляется из каталога, а администратор перенаправляется на страницу “Главная”.

Условия ввода в действие альтернативных сценариев

Условие 1. Сервер изначально недоступен.

Выводится сообщение об ошибке.

1.3.2 Нефункциональные требования

Клиентское приложение должно выполнять следующие требования:

- дизайн приложения должен соответствовать дизайн-системе ВКонтакте [4];
- использовать библиотеку адаптивных React-компонентов [5];
- добавить возможность изменять тему (светлая/тёмная).

1.4 Спецификация требований на серверное приложение

1.4.1 Функциональные требования

Доступ к API осуществляется по ссылке https://api.guitarly.ru. Сервер и клиент взаимодействуют с помощью REST API и JSON.

Все методы, кроме *users/auth,* требуют наличие в HTTP-заголовка Authorization, в теле которого находится JWT-токен. В противном случае возвращается ошибка 401 Unauthorized. JWT-токен выдает метод *users/auth.*

Все методы, принимающие параметры *page* и *id*, могут вернуть ошибку 404 Not Found.

Метод users/auth

Производит вход пользователя в систему.

Входные параметры передаются с помощью HTTP POST в теле запроса на адрес https://api.guitarly.ru/users/auth. Параметры запроса передаются переданы в кодировке UTF-8.

Описание параметров:

- queryString - строка инициализации с параметрами запуска. При запуске приложения ВКонтакте в него передаются данные об источнике запуска, пользователе, его правах доступа и другая полезная информация [6].

Возвращаемый результат:

В случае успеха возвращается HTTP статус-код 200 OK и JSON следующего формата:

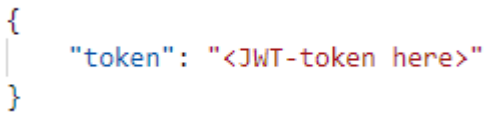

Рисунок 1.11 — пример ответа сервера для метода users/auth

Метод home/feed

Возвращает данные для главной страницы.

Никаких входных параметров не требуется. Метод вызвается с помощью HTTP GET запроса на адрес https://api.guitarly.ru/home/feed.

В случае успеха возвращается HTTP статус-код 200 OK и JSON следующего формата:

```
{
    "latestSongs": [
        {
            "id": 8214,
            "title": "Song Title",
            "fullTitle": "Singer - Song Title",
            "createdAt": "2021-11-19T10:49:10.550044",
            "publishedAt": "2021-11-19T10:49:10.550044",
            "isDeleted": false,
            "viewsNumber": 1,
            "artistId": 51
        }
    ],
    "topArtists": [
        {
            "id": 8214,
            "title": "Artist Title",
            "picture30": "/images/artists/grechka/grechka30.png",
            "picture100": "/images/artists/grechka/grechka100.png",
            "pictureOriginal": "/images/artists/grechka/grechka.png",
            "totalViews": 10
        }
    ],
    "topSongs": [
        {
            "id": 8214,
            "title": "Song Title",
            "fullTitle": "Singer - Song Title",
            "createdAt": "2021-11-19T10:49:10.550044",
            "publishedAt": "2021-11-19T10:49:10.550044",
            "isDeleted": false,
            "viewsNumber": 1,
            "artistId": 51
        }
    ]
}
```


Рисунок 1.12 — пример ответа сервера для метода home/feed

Метод home/search

Осуществляет поиск по базе данных.

Входные параметры передаются с помощью HTTP GET в строке запроса на адрес https://api.guitarly.ru/home/search. Параметры запроса должны быть переданы в кодировке UTF-8.

Описание параметров:

- q - строка поиска. Вхождение этой строки будет искаться в названиях артистов и песен.

В случае успеха возвращается HTTP статус-код 200 OK и JSON следующего формата:


```
{
    "artists": [
        {
            "id": 2,
            "title": "Кино",
            "picture30": "/images/artists/kino/kino30.jpeg",
            "picture100": "/images/artists/kino/kino100.jpeg",
            "pictureOriginal": "/images/artists/kino/kino.jpeg",
            "viewsNumber": 1
        }
    ],
    "songs": [
        {
            "id": 2,
            "title": "Пачка сигарет",
            "text": null,
            "fullTitle": "Виктор Цой - Пачка сигарет",
            "createdAt": "2021-11-17T20:37:36.197796",
            "publishedAt": "2021-11-17T20:37:36.197796",
            "isDeleted": false,
            "viewsNumber": 1,
            "artistId": 2
        }
    ]
}
```


Рисунок 1.13 — пример ответа сервера для метода home/search

Метод artists/top

Возвращает самых популярных артистов.

Входные параметры передаются с помощью HTTP GET в строке запроса на адрес https://api.guitarly.ru/artists/top. Параметры запроса должны быть переданы в кодировке UTF-8.

Описание параметров:

- page - номер страницы. Необязательный, по умолчанию - 1. На каждой странице по 10 элементов.

В случае успеха возвращается HTTP статус-код 200 OK и JSON следующего формата:


```
{
    "artists": [
        {
            "id": 2,
            "title": "Кино",
            "picture30": "/images/artists/kino/kino30.jpeg",
            "picture100": "/images/artists/kino/kino100.jpeg",
            "pictureOriginal": "/images/artists/kino/kino.jpeg",
            "viewsNumber": 1
        }
    ]
}
```


Рисунок 1.14 — пример ответа сервера для метода artists/top

Метод artists/{id}

Возвращает информацию об артисте по ID.

Входные параметры передаются с помощью HTTP GET в строке запроса на адрес https://api.guitarly.ru/artists/{id}. Параметры запроса должны быть переданы в кодировке UTF-8.

В качестве параметра передаётся ID артиста.

В случае успеха возвращается HTTP статус-код 200 OK и JSON следующего формата:

```
{
    "id": 25,
    "title": "Гражданская оборона",
    "picture30": "/images/artists/grob/grob30.jpg",
    "picture100": "/images/artists/grob/grob100.jpg",
    "pictureOriginal": "/images/artists/grob/grob.jpg",
    "viewsNumber": 1,
    "songs": [
        {
            "id": 420,
            "title": "Все идет по плану",
            "text": null,
            "fullTitle": "Гражданская Оборона - Все идет по плану",
            "createdAt": "2021-11-17T20:50:45.169245",
            "publishedAt": "2021-11-17T20:50:45.169245",
            "isDeleted": false,
            "viewsNumber": 0,
            "artistId": 25
        }
    ]
}
```


Рисунок 1.15 — пример ответа сервера для метода artists/{id}

Метод songs/{id}

Возвращает информацию о песне по ID.

Входные параметры передаются с помощью HTTP GET в строке запроса на адрес https://api.guitarly.ru/songs/{id}. Параметры запроса должны быть переданы в кодировке UTF-8.

В качестве параметра передаётся ID песни.

В случае успеха возвращается HTTP статус-код 200 OK и JSON следующего формата:

```
{
    "song": {
        "id": 3891,
        "title": "Я с тобой",
        "text": "<span className=\"chord\">C</span> <span className=\"chord\">F</span>\nМир такой красивый...",
        "fullTitle": "Звери - Я с тобой",
        "createdAt": "2021-11-17T22:26:57.944991",
        "publishedAt": "2021-11-17T22:26:57.944991",
        "isDeleted": false,
        "viewsNumber": 1,
        "artistId": 21,
        "artist": {
            "id": 21,
            "title": "Звери",
            "picture30": "/images/artists/zveri/zveri30.jpeg",
            "picture100": "/images/artists/zveri/zveri100.jpeg",
            "pictureOriginal": "/images/artists/zveri/zveri.jpeg",
            "viewsNumber": 2
        }
    },
    "recommendations": [
        {
            "id": 3885,
            "title": "Просто такая сильная любовь",
            "text": null,
            "fullTitle": "Звери - Просто такая сильная любовь",
            "createdAt": "2021-11-17T22:26:50.093259",
            "publishedAt": "2021-11-17T22:26:50.093259",
            "isDeleted": false,
            "viewsNumber": 2,
            "artistId": 21
        }
    ],
    "isFavorite": true,
    "isPublished": true
}
```


Рисунок 1.16 — пример ответа сервера для метода songs/{id}

Метод songs/latest

Возвращает список последних добавленных песен.

Входные параметры передаются с помощью HTTP GET в строке запроса на адрес https://api.guitarly.ru/songs/latest. Параметры запроса должны быть переданы в кодировке UTF-8.

Описание параметров:

- page - номер страницы. Необязательный, по умолчанию - 1. На каждой странице по 10 элементов.

В случае успеха возвращается HTTP статус-код 200 OK и JSON следующего формата:

```
[
    {
        "id": 8214,
        "title": "Крики",
        "text": null,
        "fullTitle": "Гречка - Крики",
        "createdAt": "2021-11-19T10:49:10.550044",
        "publishedAt": "2021-11-19T10:49:10.550044",
        "isDeleted": false,
        "viewsNumber": 1,
        "artistId": 51
    }
]
```


Рисунок 1.17 — пример ответа сервера для метода songs/latest

Метод songs/top

Возвращает список самых популярных песен.

Входные параметры передаются с помощью HTTP GET в строке запроса на адрес https://api.guitarly.ru/songs/top. Параметры запроса должны быть переданы в кодировке UTF-8.

Описание параметров:

- page - номер страницы. Необязательный, по умолчанию - 1. На каждой странице по 10 элементов.

В случае успеха возвращается HTTP статус-код 200 OK и JSON следующего формата:


```
[
    {
        "id": 8214,
        "title": "Крики",
        "text": null,
        "fullTitle": "Гречка - Крики",
        "createdAt": "2021-11-19T10:49:10.550044",
        "publishedAt": "2021-11-19T10:49:10.550044",
        "isDeleted": false,
        "viewsNumber": 1,
        "artistId": 51
    }
]
```

Рисунок 1.18 — пример ответа сервера для метода songs/top

Метод songs/unpublished

Возвращает список неопубликованных песен. Метод доступен только администратору.

Входные параметры передаются с помощью HTTP GET в строке запроса на адрес https://api.guitarly.ru/songs/unpublished. Параметры запроса должны быть переданы в кодировке UTF-8.

Описание параметров:

- page - номер страницы. Необязательный, по умолчанию - 1. На каждой странице по 10 элементов.

В случае успеха возвращается HTTP статус-код 200 OK и JSON следующего формата:


```
[
    {
        "id": 8214,
        "title": "Крики",
        "text": null,
        "fullTitle": "Гречка - Крики",
        "createdAt": "2021-11-19T10:49:10.550044",
        "publishedAt": "2021-11-19T10:49:10.550044",
        "isDeleted": false,
        "viewsNumber": 1,
        "artistId": 51
    }
]
```


Рисунок 1.19 — пример ответа сервера для метода songs/unpublished

Метод songs/new

Добавляет песню в базу данных. Метод доступен только администратору.

Входные параметры передаются с помощью HTTP POST в теле запроса на адрес https://api.guitarly.ru/songs/new. Параметры запроса должны быть переданы в кодировке UTF-8.

Пример тела запроса:

```
{
    "title": "<Title here>",
    "fullTitle": "<Full title here>",
    "text": "<Song text here>",
    "artistId": 51
}
```


Рисунок 1.20 — пример тела запроса для метода songs/new

В случае успеха возвращается HTTP статус-код 200 OK с ID новой песни.

Метод songs/edit/{id}

Обновляет данные песни в базе данных. Метод доступен только администратору.

Входные параметры передаются с помощью HTTP PUT в теле запроса на адрес https://api.guitarly.ru/songs/edit/{id}. Параметры запроса должны быть переданы в кодировке UTF-8.


```
{
    "title": "<Title here>",
    "fullTitle": "<Full title here>",
    "text": "<Song text here>",
    "artistId": 51
}
```


Рисунок 1.21 — пример тела запроса для метода songs/edit/{id}

В случае успеха возвращается HTTP статус-код 200 OK с ID новой песни.

Метод songs/publish/{id}

Опубликовать песню, чтобы она попала в каталог. Метод доступен только администратору.

Входные параметры передаются с помощью HTTP POST в URL запроса на адрес https://api.guitarly.ru/songs/publish/{id}. Параметры запроса должны быть переданы в кодировке UTF-8.

В качестве параметра передаётся ID песни, которую надо опубликовать. В случае успеха возвращается HTTP статус-код 200 OK.

Метод songs/favorite/{id}

Добавить или удалить песню в избранное.

Входные параметры передаются с помощью HTTP POST в URL запроса на адрес https://api.guitarly.ru/songs/favorite/{id}. Параметры запроса должны быть переданы в кодировке UTF-8.

В качестве параметра передаётся ID песни, которую надо опубликовать. В случае успеха возвращается HTTP статус-код 200 OK и булево-значение. Если вернулось true, то песня добавлена в песенник. Иначе - песня удалена из песенника.

Метод songs/favorites

Возвращает список песен из песенника.

Входные параметры передаются с помощью HTTP GET в строке запроса на адрес https://api.guitarly.ru/songs/favorites. Параметры запроса должны быть переданы в кодировке UTF-8.

Описание параметров:

- page - номер страницы. Необязательный, по умолчанию - 1. На каждой странице по 10 элементов.

В случае успеха возвращается HTTP статус-код 200 OK и JSON следующего формата:


```
[
    {
        "id": 8214,
        "title": "Крики",
        "text": null,
        "fullTitle": "Гречка - Крики",
        "createdAt": "2021-11-19T10:49:10.550044",
        "publishedAt": "2021-11-19T10:49:10.550044",
        "isDeleted": false,
        "viewsNumber": 1,
        "artistId": 51
    }
]
```


Рисунок 1.22 — пример ответа сервера для метода songs/favorites

Метод songs/delete/{id}

Удалить песню. Метод доступен только администраторам.

Входные параметры передаются с помощью HTTP DELETE в строке запроса на адрес https://api.guitarly.ru/songs/delete/{id}. Параметры запроса должны быть переданы в кодировке UTF-8.

В качестве параметра передаётся ID песни. В случае успеха возвращается HTTP статус-код 200 OK.

Метод songs/transpose

Изменить тональность песни.

Входные параметры передаются с помощью HTTP POST в теле запроса на адрес https://api.guitarly.ru/songs/transpost. Параметры запроса должны быть переданы в кодировке UTF-8.

Описание параметров:

- id - ID песни
- tone - требуемая тональность.

В случае успеха возвращается HTTP статус-код 200 OK и текст песни с аккордами в новой тональности.

1.4.2 Нефункциональные требования

Сервер должен выполнять следующие требования:

- работать на операционной системе Linux Ubuntu 18.04.6 LTS;
- при отсутствии нагрузки, работать безотказно;
- код программы должен быть написан в едином стиле;
- придерживаться архитектурного подхода REST API.

1.5 Выводы по главе

На основе анализа аналогов, сделан вывод, что на рынке нет приложений, полностью удовлетворяющих всем требованиям задания. Рассмотренные сайты имеют ряд критических недостатков: ограниченный каталог песен, устаревший дизайн, перегруженность рекламой.

Сформулирована спецификация требований для клиентского приложения в виде диаграмм прецедентов и их текстового описания. Подробно рассмотрены

все функции, необходимые для обеспечения конкурентоспособности приложения.

Сформулирована спецификация требований для серверного приложения в виде текстового описания поддерживаемых методов. Методы разработаны в соответствии с требованиями клиенткой части.

2 Проектирование

Прецеденты описывают поведение пользователя в системе. Для визуализации взаимодействия компонентов между собой, а также с пользователем используется диаграмма последовательностей, отражающая динамическую модель системы. Результатом проектирования является статическая модель системы – диаграмма классов, на основе которой может быть выполнена генерация кода.

На рисунке 2.1 приведена архитектура разработанной системы. Система состоит из двух подсистем: клиентское и серверное приложения. Клиентское приложение выполняет роль пользовательского интерфейса. Для получения данных оно сообщается с серверным приложением по протоколу HTTP. Серверное приложение, или же API, принимает запросы от клиента, обрабатывает их и возвращает ответ. Для хранения данных на сервере используется база данных, в ней хранится информация о пользователях, песнях и артистах. Также на сервере используется файловая система для хранения изображений артистов.

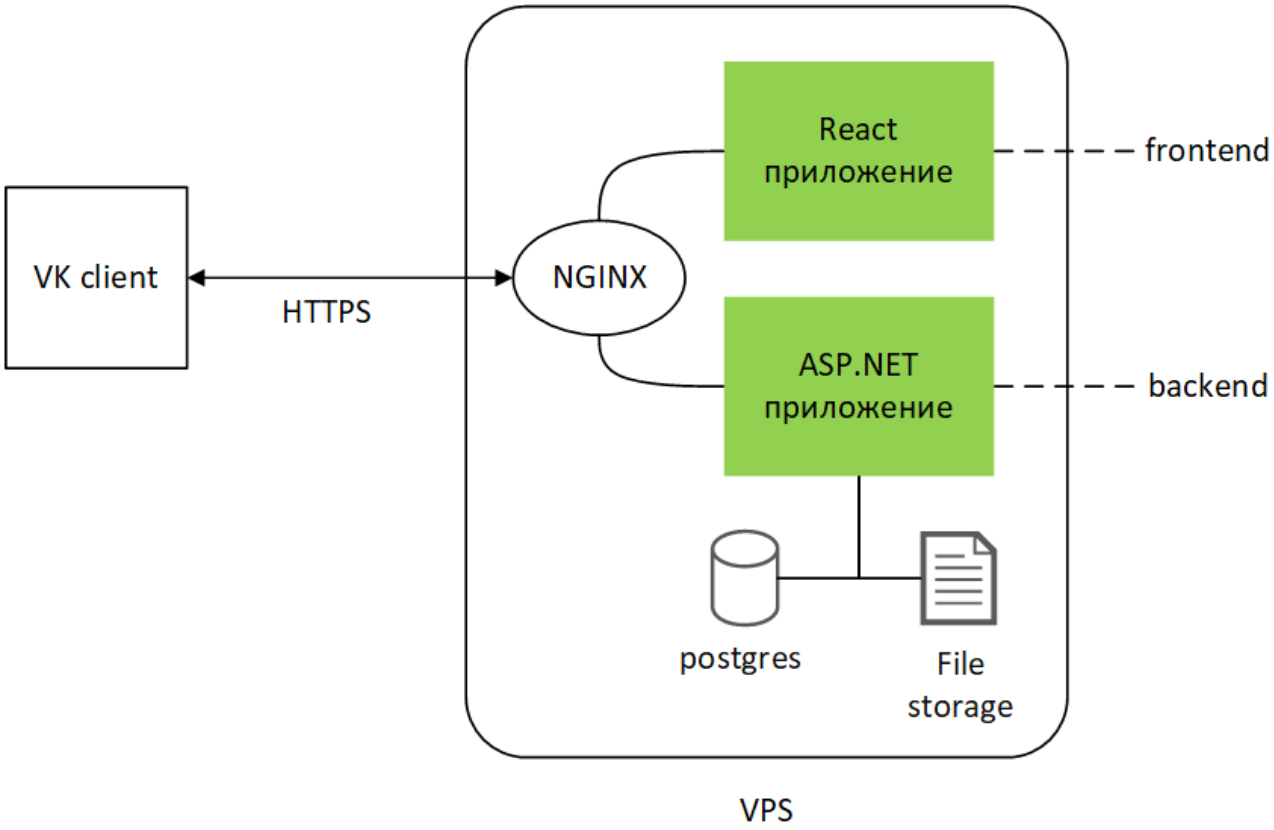

Рисунок 2.1 — Архитектура системы

2.1 Динамическая модель системы

2.1.1 Диаграммы последовательности

Значительная часть логики приложения связана с песнями и артистами. В разделе приведены диаграммы последовательностей, для наиболее значимых прецедентов, связанных со взаимодействием различных компонентов системы.

На рисунке 2.2 изображена диаграмма последовательности для варианта использования “Просмотр песни”, наглядно представляющая процесс взаимодействия клиента с сервером для получения текста песни и отображения ее на экране пользователя.

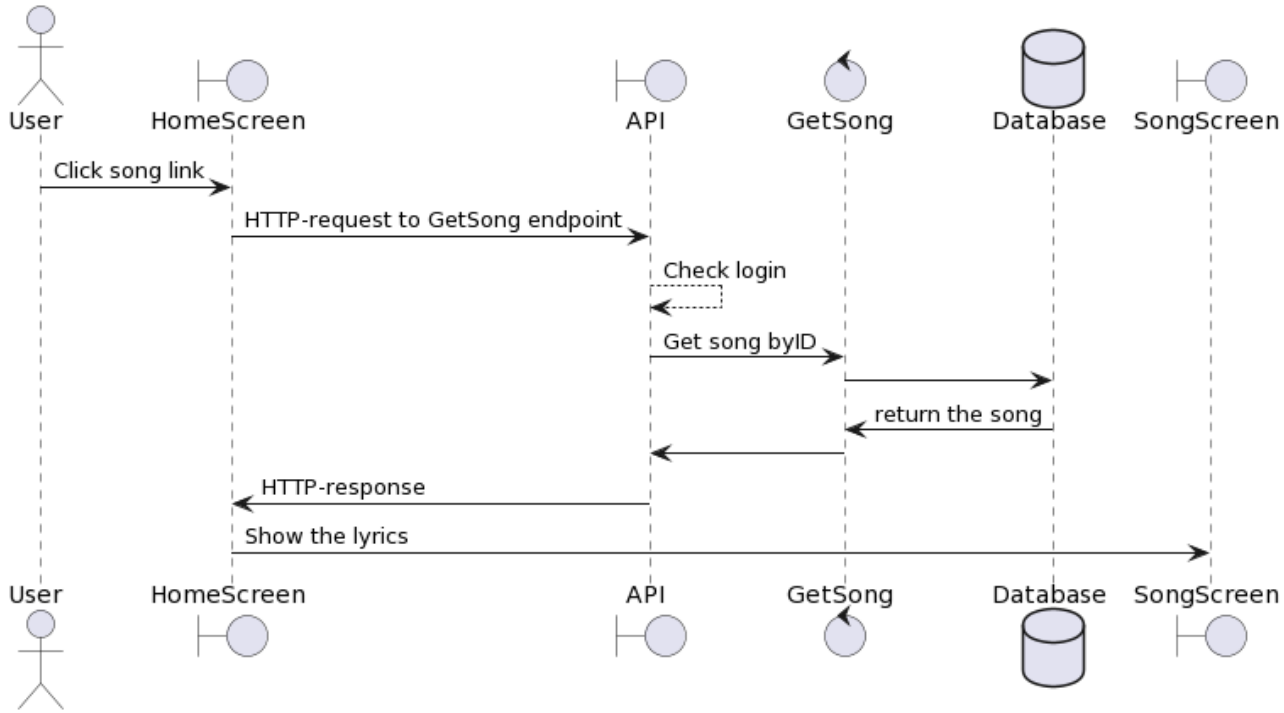

Рисунок 2.2 — Диаграмма последовательности “Просмотр песни”

Когда пользователь нажимает на ссылку на песню, с клиентского приложения делается HTTP-запрос к API. Этот запрос содержит в себе ID песни и данные для идентификации пользователя (JWT-токен). На сервере происходит проверка токена и делается запрос к базе данных. Если песня с запрашиваемым ID существует, то она возвращается сервером в виде json-объекта. На клиенте отрисовывается другая страница на которую выводится текст песни.

На рисунке 2.3 приведена диаграмма последовательности для варианта использования “Добавить песню”. Этот прецедент интересен тем, что он доступен только администратору приложения и включает в себя работу с базой данных как в режиме чтения, так и в режиме записи.

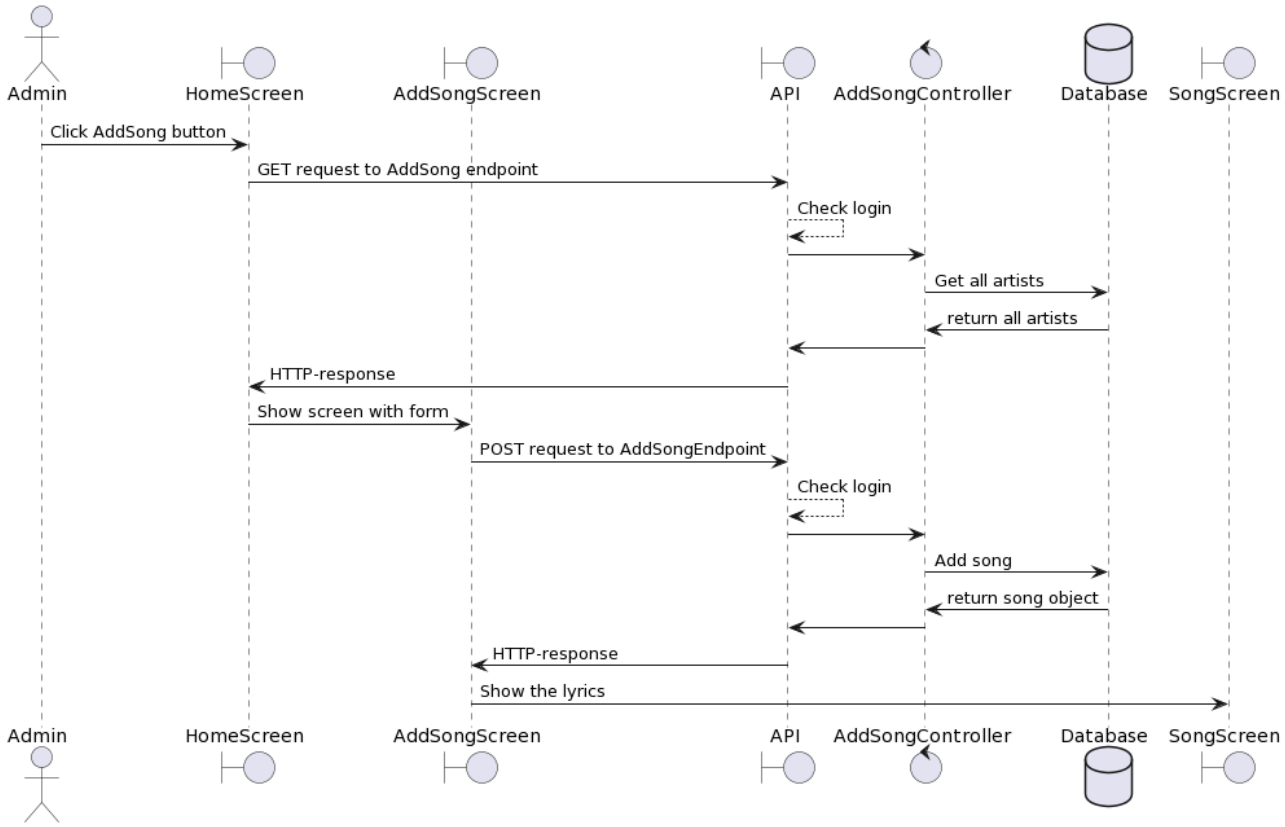

Рисунок 2.3 — Диаграмма последовательности “Добавить песню”

У администратора на главном экране отображается кнопка “Добавить песню”. При нажатии на нее, происходит запрос к API для того, чтобы получить список имеющихся артистов в базе данных. Список артистов возвращается обратно клиенту и клиент показывает администратору другую страницу, на которой находится форма для заполнения. После заполнения формы, она отправляется HTTP POST запросом на сервер, где происходит запись в базу данных. После успешного сохранения администратору показывается экран с песней.

2.1.2 Алгоритм выделения аккордов

Для удобства пользователя решено визуально отделять аккорды внутри текста песни от самих слов. На рисунке 2.4 показан пример текста песни и выделенных другим цветом аккордов.

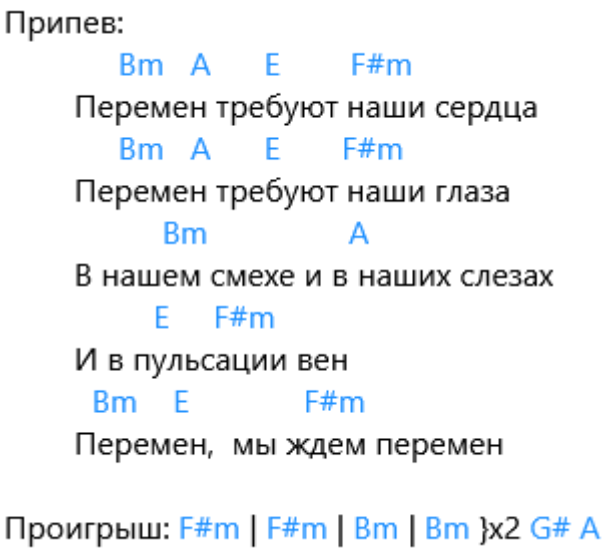

Рисунок 2.4 — Пример песни с аккордами

Аккорд - (в текстовом представлении) это комбинация нескольких символов. Первым символом идет нота, она может быть одним из следующих вариантов: A (ля), B (си), C (до), D (ре), E (ми), F (фа), G (соль). Нота - это обязательная часть аккорда. Рядом с нотой могут стоять дополнительные символы, обозначающие минорность или мажорность аккорда, его тонику, интервал, добавочный звук. [7]

Для того, чтобы выделять аккорды из всего текста, можно воспользоваться регулярными выражениями. Экспериментальным путем был выведен следующий шаблон для поиска: *([ABCDEFGH][#-+\d|mmaj|maj|dim|sus|b|o|aug|add|verm]*(\/\d.)?)($|\s|[.?!)(,:\/])*.При добавлении новой песни, на сервере происходит предобработка, а именно поиск аккордов с помощью вышеуказанного шаблона и добавление вокруг них тегов <chord> и </chord>, спереди и сзади соответственно.

На клиенте, перед тем как отобразить текст на экране, происходит замена всех тегов <chord>...</chord> на html-тег ... со специальным css-стилем.

2.1.3 База данных

Для хранения информации о песнях, артистах и пользователях на сервере используется база данных. На рисунке 2.5 представлена её ER-диаграмма.

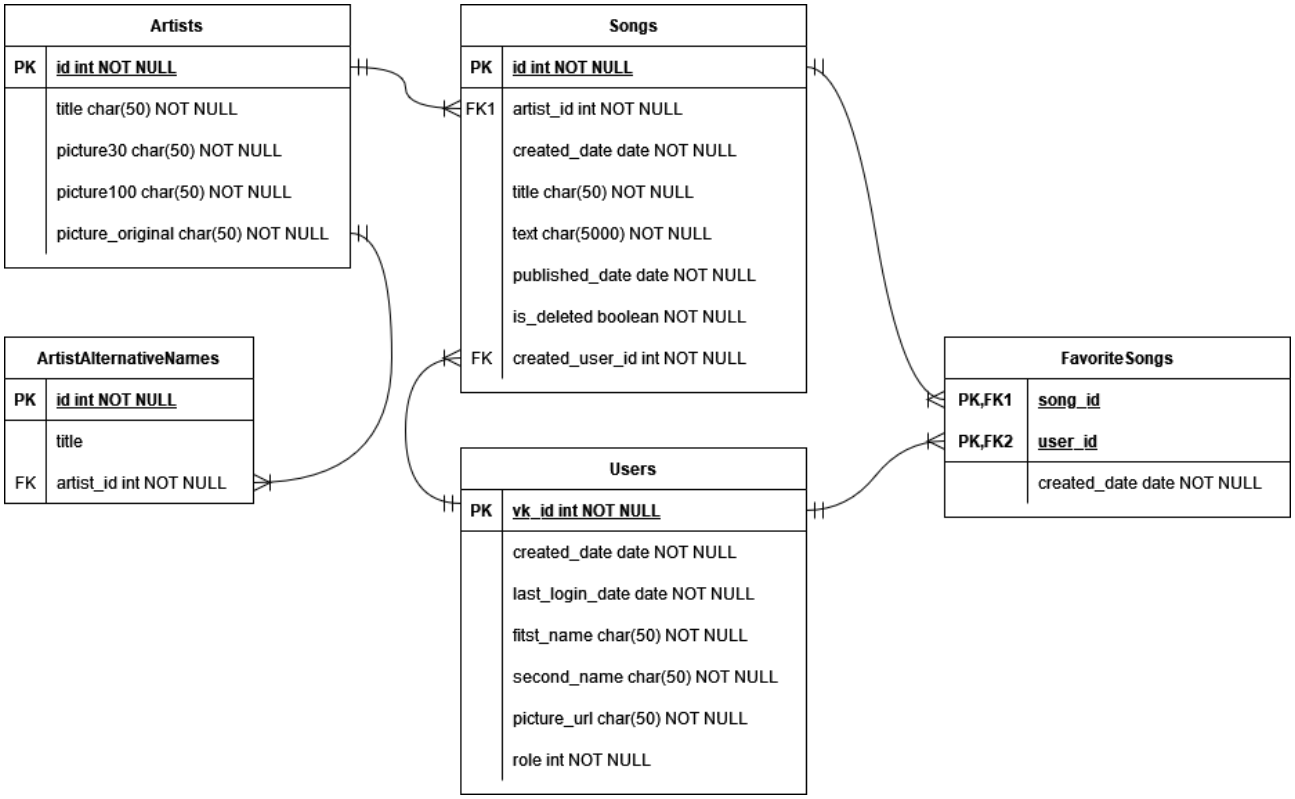

Рисунок 2.5 — ER-диаграмма базы данных

На данном этапе база данных состоит из трех основных таблиц (*Users*, *Songs*, *Artists*) и двух вспомогательных (*FavoriteSongs*, *ArtistAlternativeNames*). *FavoriteSongs* используется при добавлении/удалении песни из избранного. Таблица *ArtistAlternativeNames* хранит в себе дополнительные имена артистов, это помогает при поиске.

2.2 Выводы по главе

В соответствии с техническим заданием:

- предложена архитектура системы и структура базы данных;
- с помощью нотации диаграмм последовательностей задокументированы наиболее сложные отношения в системе, более детально проработано взаимодействие объектов;

- разработан алгоритм выделения аккордов в тексте песни.

3 Реализация и документация

3.1 Реализация

3.1.1 Выбор инструментов

Для реализации клиентской части было решено выбрать JavaScript-библиотеку Reactjs, так как она обладает большим количеством готовых компонентов и удобна в использовании. Также немаловажным фактором является то, что ВКонтакте рекомендует Reactjs для разработки приложений для VK MiniApps, так как у них уже приготовлено много библиотек и примеров кода для этой платформы.

Для реализации серверной части выбран язык C# и фреймворк ASP.NET Core.

3.1.2 Главный экран

Приложение содержит несколько экранов, поэтому необходимо осуществлять переход между ними. Логика, структура и стили главного экрана прописаны в компоненте *Home,* в котором указаны ссылки на другие компоненты - *Artist, Song, Search, FavoriteSongs*.

Игровой экран состоит из нескольких блоков:

- последние добавленные песни;
- самые популярные артисты;
- самые популярные песни.

Информация для этих блоков подгружается с сервера по GET-запросу на https://api.guitarly.ru/home/feed и затем рендерится на экран пользователя.

3.1.4 База данных

Данные приложения (информация о пользователях, тексты песен, артисты) хранятся в базе данных PostgreSQL. Чтобы не работать с SQL напрямую, в проекте используется Entity Framework Core - решение для работы с базами данных (ORM), которое используется в программировании на языках семейства .NET. Оно позволяет взаимодействовать с СУБД с помощью сущностей *(entity),* а не таблиц. Работа с Entity Framework Core осуществляется по принципу “Code First” - ORM строит базу данных на основе миграционных файлов, которые генерируются из переданных моделей. На рисунке 3.1 представлена схема работы с базой данных с помощью Entity Framework Core.

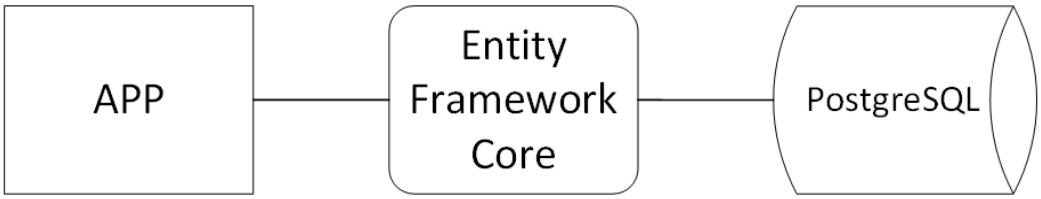

Рисунок 3.1 — Схема работы с базой данных

3.1.5 Интерфейс

Интерфейс приложения построен с помощью React-компонентов VKUI [5]. VKUI - набор готовых компонентов от разработчиков ВКонтакте, которые подходят по стилю под дизайн сайта ВКонтакте [8].

На каждый экран приложения в коде клиентской части выделен отдельный React-компонент, в котором описывается структура и стили страницы.

Изменение цветовой схемы интерфейса (светлая/темная темы) реализованы с помощью библиотеки vk-bridge [9]. Вызов метода *subscribe* этой библиотеки автоматически проверяет значение специальной переменной в файлах cookies и обновляет ее (переменную), если пользовательская тема на сайте vk.com поменялась.

3.2 Инструкция разработчика

Настройка окружения программиста для разработки приложений с использованием Node.js, .NET и PostgreSQL одновременно осуществляется нетривиально. Однако, доставку окружения можно обеспечить средствами виртуализации.

Для Node.js был выбран официальный Docker-образ от разработчиков языка: node:16.14.0-alpine [10].

Для .NET были выбраны официальные образы от Microsoft: mcr.microsoft.com/dotnet/aspnet:5.0 [11] и mcr.microsoft.com/dotnet/sdk:5.0 [12]. Первый служит для развертывания релиз-версии приложения, а второй для сборки проекта.

Для базы данных использовался официальный образ от разработчиков PostgreSQL: postgres [13].

Для развертывания трех контейнеров одновременно создан файл docker-compose.yml, таким образом проект можно запустить одной командой: *docker-compose up.*

3.3 Тестирование

Тестирование приложение осуществлялось с помощью специальной платформы ВКонтакте “VK Testers” [14]. В тестировании приняло участие около 20 человек и было найдено и исправлено больше 50 ошибок.

На рисунке 3.2 изображен пример отчета о найденной проблеме.

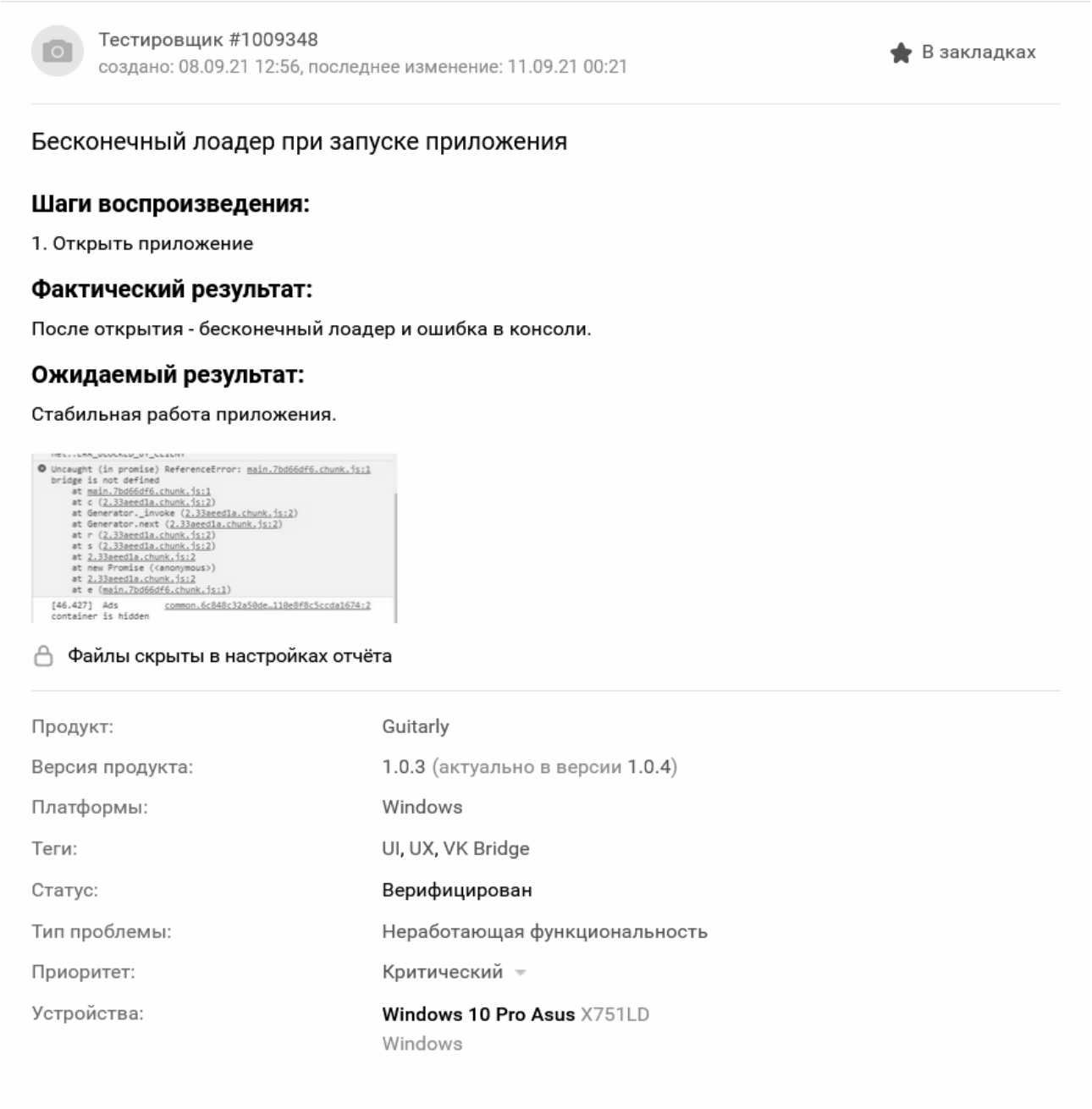

Рисунок 3.2 — Пример отчета о найденной проблеме

Тестировщик сообщил, что в браузере Google Chrome приложение не открывается, а вместо этого появляется бесконечная загрузка. Проблема была в том, что в браузерах Google Chrome и Apple Safari алгоритм установки переменной куки отличается от Mozilla Firefox. В дальнейшем эта ошибка была исправлена.

3.4 Выводы по главе

1. Реализовано приложение согласно спецификации требований.
2. Составлена инструкция по настройке окружения разработчика для использования трех Docker-контейнеров одновременно.
3. Проведено тестирование приложения с использованием платформы “VK Testers”.

ЗАКЛЮЧЕНИЕ

В результате проделанной работы:

1. Спроектировано, реализовано и протестировано приложение “Guitarly”.
2. Создано удобное окружение для разработки.
3. Приложение опубликовано в общий доступ и доступно по адресу https://guitarly.ru [15].

В разработанном приложении присутствуют недостатки, исправить их можно путем пополнения функциональных возможностей:

- добавить категории песен;
- добавить сортировку в каталоги песен (по названию, по популярности, по дате добавления);
- добавить возможность комментировать песни;
- добавить возможность добавлять песни самим пользователям;
- улучшить поиск по приложению.

Исходный код приложения доступен для скачивания с git-репозитория [16].

СПИСОК СОКРАЩЕНИЙ

СУБД – система управления базами данных

ORM (Object-Relational Mapping) – объектно-реляционное отображение

ER-диаграмма (Entity-Relationship diagram) – диаграмма «сущность-связь»

API (Application Programming Interface) – программный интерфейс приложения

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. AmDm.ru - портал для музыкантов [Электронный ресурс]: – Режим доступа: https://amdm.ru/
2. Аккорды песен под гитару [Электронный ресурс]: – Режим доступа: https://hm6.ru/
3. AkkordAm.ru - обучение игре на гитаре [Электронный ресурс]: – Режим доступа: https://akkordam.ru/
4. Figma [Электронный ресурс]: – Режим доступа: https://www.figma.com/@vk – Дизайн система vk.com
5. Github [Электронный ресурс]: – Режим доступа: https://vkcom.github.io/VKUI/ – библиотека адаптивных React-компонентов VKUI
6. VK для разработчиков [Электронный ресурс]: – Режим доступа: https://vk.com/dev/apps_init – Использование API | Параметры запуска приложения
7. Wikipedia [Электронный ресурс]: - Режим доступа: https://ru.wikipedia.org/wiki/Буквенно-цифровое_обозначение_аккорда - Буквенно-цифровое обозначение аккорда - Wikipedia
8. ВКонтакте [Электронный ресурс]: - Режим доступа: https://vk.com
9. Github [Электронный ресурс]: - Режим доступа: https://github.com/VKCOM/vk-bridge - VKCOM/vk-bridge: A package for integrating VK Mini Apps with official VK clients for iOS, Android and Web
10. Docker Hub [Электронный ресурс] - Режим доступа: https://hub.docker.com/_/node - Node - Official Image | Docker Hub
11. Docker Hub [Электронный ресурс] - Режим доступа: https://hub.docker.com/_/microsoft-dotnet-aspnet - ASP.NET Core Runtime by Microsoft | Docker Hub

12. Docker Hub [Электронный ресурс] - Режим доступа: https://hub.docker.com/_/microsoft-dotnet-sdk - .NET SDK by Microsoft | Docker Hub
13. Docker Hub [Электронный ресурс] - Режим доступа: https://hub.docker.com/_/postgres - Postgres - Official Image | Docker Hub
14. VK Testers [Электронный ресурс] - Режим доступа: https://vk.com/bugs
15. Guitarly [Электронный ресурс] - Режим доступа: https://guitarly.ru
16. Github [Электронный ресурс] - Режим доступа: https://github.com/zn/guitarly-code - zn/guitarly-code: Codebase for guitarly.ru

Федеральное государственное автономное
образовательное учреждение
Высшего образования
«СИБИРСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Институт космических и информационных технологий
институт
Кафедра вычислительной техники
кафедра

УТВЕРЖДАЮ
Заведующий кафедрой
_____ О.В. Непомнящий
«20» 06 2022 г.

БАКАЛАВРСКАЯ РАБОТА

09.03.01 – Информатика и вычислительная техника
код – наименование направления

Приложение с аккордами песен на платформе VK MiniApps
тема

Руководитель _____ В. С. Васильев
подпись, дата

Выпускник _____ А. А. Свистунов
подпись, дата

Нормоконтролер _____ В. С. Васильев
подпись, дата

Красноярск 2022