

Оглавление

Введение.....	2
Определения, обобщения и сокращения	3
Описание предметной области	4
Описание прикладного процесса.....	4
Формирование требований	4
Проектирование.....	6
Используемый стек технологий	6
Системная архитектура	6
Архитектура данных.....	7
Программная архитектура.....	8
Разработка	12
Реализация серверного API.....	12
Реализация пользовательского интерфейса	16
Заключение	24
Список использованной литературы	25

Введение

Объектом разработки является генератор музыкальных плейлистов на языке программирования Kotlin с использованием фреймворка Spring – для серверной части приложения и языка программирования JavaScript, без использования фреймворков – для клиентской части приложения. Выбранный фреймворк один из самых поддерживаемых и распространённых.

Целью работы является разработка веб-приложения и пользовательского интерфейса, анализ требования и моделирование процессов, средств автоматизации и архитектуры информационной системы.

В ходе работы были получены следующие результаты:

- Серверная часть системы, принимающая запросы.
- Клиентская часть системы, предоставляющая интерфейс пользователя.
- База данных для хранения информации о пользователях, файлах, а также служебной информации внутри системы.

Определения, обобщения и сокращения

Spotify – сервис для прослушивания музыки

Фреймворк – используемый при разработке Spring Framework

Плейлист – набор музыкальных треков. Может быть сохранен как в сервисе Spotify, так и в разрабатываемом веб-приложении

Музыкальный профиль – набор из 6 характеристик, описывающий музыкальные предпочтения пользователя: акустичность, танцевальность, инструментальность, энергичность, валентность и количество речи в треках

Описание предметной области

Описание прикладного процесса

Разрабатываемое веб-приложение представляет собой систему управления контентом в своем аккаунте Spotify. Такая система должна позволять управлять содержимым аккаунта, не обращаясь непосредственно к данному сервису

Можно выделить такие операции, как: создание новых плейлистов, просмотр уже созданных, а также просмотр статистики прослушивания своего аккаунта, используемую при генерации

Формирование требований

В ходе анализа прикладного процесса был получен следующий список функциональных требований:

Модуль аутентификации должен иметь:

- Возможность аутентификации через Spotify
- Возможность выйти из аккаунта

Модуль работы с плейлистами должен иметь:

- Возможность создания плейлиста с выгрузкой в Spotify
 - по самым прослушиваемым трекам в мире
 - по любимым исполнителям пользователя
 - по любимым трекам пользователя
- Возможность просматривать ранее созданные плейлисты
- Возможность восстановить ранее созданный плейлист

Модуль обращения к сервису Spotify должен позволять:

- Получить мировой рейтинг треков за текущий день

Модуль получения статистики пользователя должен позволять:

- Получить список самых прослушиваемых исполнителей

- Получить список самых прослушиваемых треков
- Получить статистику по любимым жанрам
- Получить общий музыкальный профиль пользователя

Нефункциональные требования

Разрабатываемая система не является публичной, поэтому основным требованием выступает её закрытость, путем обязательного прохождения процесса авторизации и аутентификации.

Так же должна быть возможность работы с приложением напрямую, через общие программные интерфейсы, описанные по спецификации OpenAPI версии не ниже 3.0

Проектирование

Используемый стек технологий

Решение создания именно веб-приложения обусловлено тем, что необходимо было обеспечить доступ к системе с любого устройства, в любое время. Веб-приложение решает этот вопрос, а также снимает вопрос обновлений на стороне клиента.

Проект использует стек стандартных технологий, характерный для большинства веб-приложений: HTML, CSS, Javascript, а также Kotlin

В качестве веб-сервера используется Tomcat.

В качестве базы данных используется PostgreSQL, ввиду того, что данная СУБД является самой стабильной в данной связке. Библиотекой для работы с данными была выбрана Spring Data JPA.

В проекте используется свободная распределённая система управления версиями Git, хранилищем исходных кодов является крупнейший веб-сервис GitHub, а в качестве хостинга используется облачный сервис Render.com.

Системная архитектура

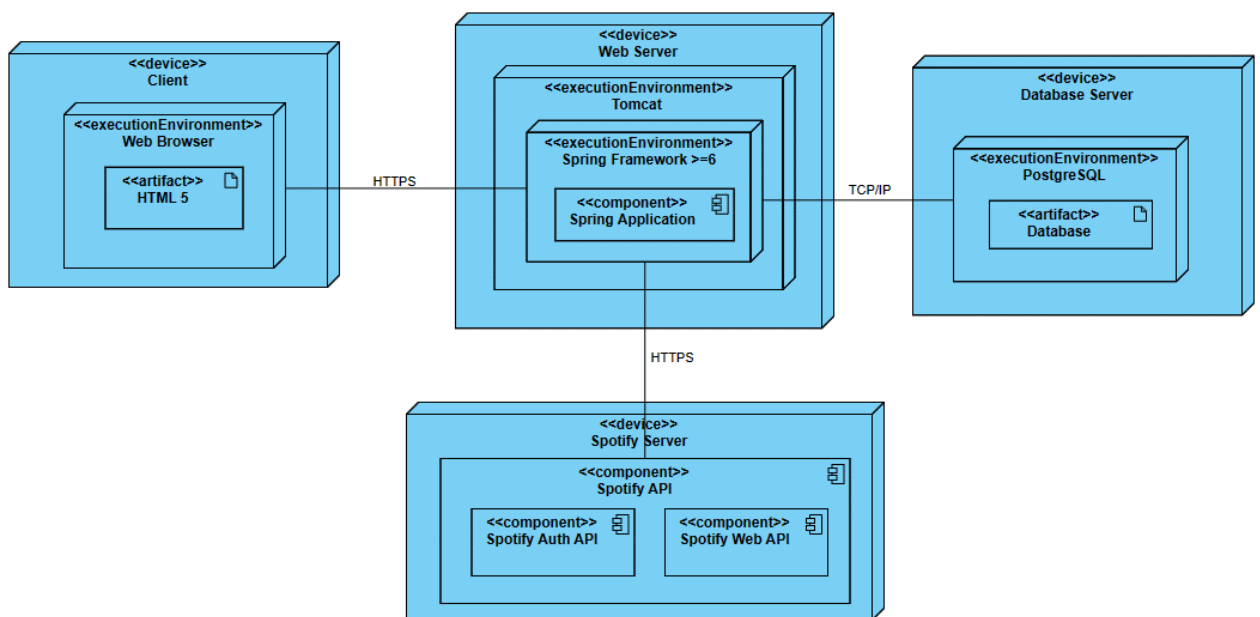


Рисунок 1. Системная архитектура приложения

На сервере приложения развернут контейнер Apache Tomcat, в котором запущена фреймворк Spring с приложением внутри. Веб-приложение подключается по протоколу TCP/IP к серверу базы данных с СУБД PostgreSQL. Для аутентификации и авторизации пользователя и получения данных об аккаунте приложение подключается по протоколу HTTPS к серверу Spotify. Подключение к браузеру на клиентском компьютере со страницами HTML производится также по протоколу HTTPS

Архитектура данных

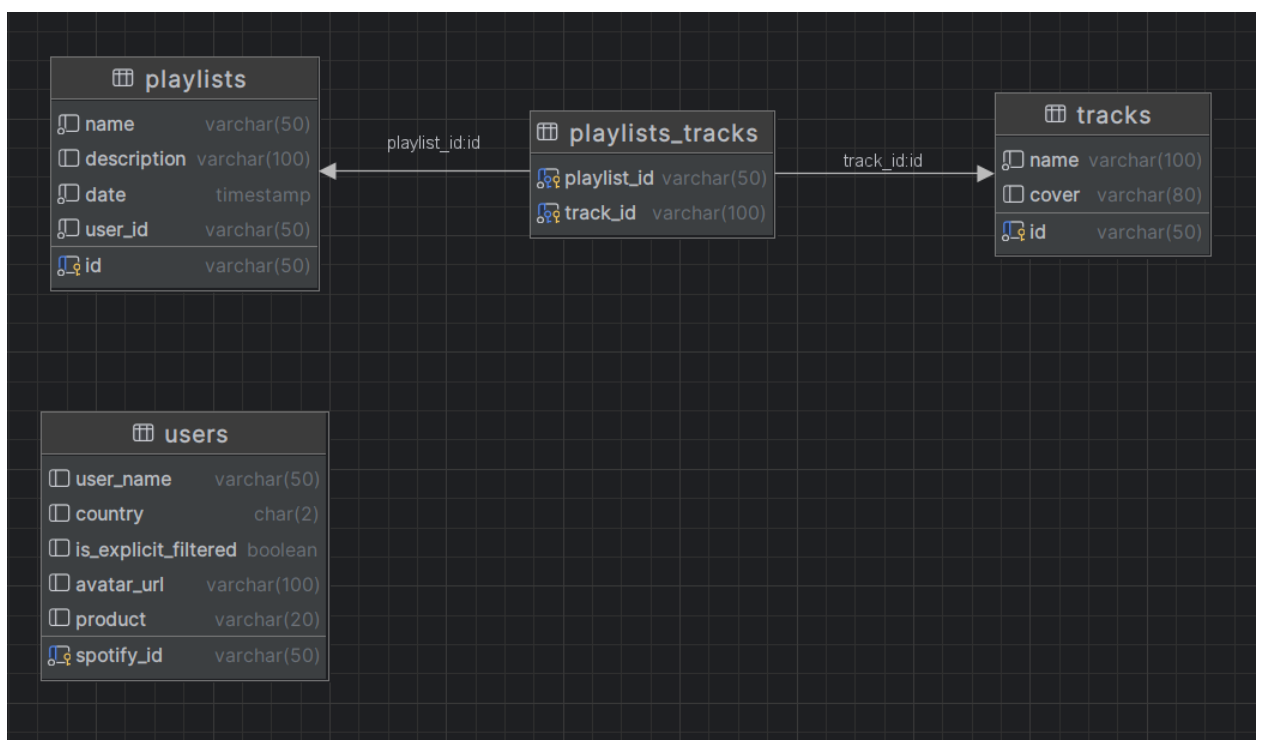


Рисунок 2. Диаграмма базы данных

Playlists – информация о плейлистах, созданных пользователями

Tracks – треки в созданных плейлистах

Playlists_tracks – обеспечивает связь много-ко-многом между треками и плейлистами

Users – информация о пользователях сервиса

Программная архитектура

Таблица 1. Отношения модулей и классов

Название модуля	Название класса	Назначение класса
Authorization	UsersAuthService	Сервис
	SpotifyUserDto	Модель передачи данных
	SpotifyAuthorizedUser	Сущность
	SpotifyTokenIntrospector	Анализатор токенов
Playlists	MessageResponse	Модель передачи данных
	Playlist	Сущность
	Track	Сущность
	PlaylistsController	Контроллер
	PlaylistsService	Сервис
	PlaylistsRepository	Репозиторий
	TracksRepository	Репозиторий
Spotify	AlbumDto	Модель передачи данных
	ArtistDto	Модель передачи данных
	ImageDto	Модель передачи данных
	PlaylistDto	Модель передачи данных
	PlaylistInfoDto	Модель передачи данных
	RecommendationsDto	Модель передачи данных
	TopArtistsDto	Модель передачи данных
	TopTracksDto	Модель передачи данных
	TrackDto	Модель передачи данных
	TrackFeaturesDto	Модель передачи данных
	SpotifyController	Контроллер
	SpotifyService	Сервис
Statistics	GenresDto	Модель передачи данных
	StatisticsService	Сервис

	StatisticsController	Контроллер
Users	User	Сущность
	UsersRepository	Репозиторий
	UsersService	Сервис

Таблица 2. Описание классов

Название класса	Описание класса
UsersAuthService	Класс, используемый при аутентификации пользователя по протоколу OAuth2. При наличии пользователя в системе обновляет его, при отсутствии – создает нового
SpotifyUserDto	Модель данных, хранящая имя и идентификатор пользователя, используемая при интроспекции токена
SpotifyAuthorizedUser	Сущность, отражающая аутентифицированного через OAuth2 пользователя
SpotifyTokenIntrospector	Класс, производящий проверку токена при аутентификации
MessageResponse	Класс, используемый при выполнении вызовов к системе. Хранит код ответа и сообщение
Playlist	Сущность, отражающая созданный плейлист вместе с треками
Track	Сущность, хранящая трек из одного из созданных плейлистов
PlaylistsController	Класс, позволяющий взаимодействовать с плейлистами
PlaylistsService	Класс, используемый для работы с плейлистами. Осуществляет генерацию плейлистов, а также их восстановление и отображение
PlaylistsRepository	Репозиторий, позволяющий осуществлять операции по получению, сохранению, обновлению и удалению плейлистов в базе данных

TracksRepository	Репозиторий, позволяющий осуществлять операции по получению, сохранению, обновлению и удалению треков в базе данных
AlbumDto	Класс, хранящий данные об альбоме, полученные от Spotify
ArtistDto	Класс, хранящий данные об исполнителе, полученные от Spotify
ImageDto	Класс, хранящий ссылку на иконку объекта, полученного от Spotify
PlaylistDto	Класс, хранящий данные о плейлисте, полученные от Spotify
PlaylistInfoDto	Класс, хранящий данные о плейлисте. Используется для создания плейлиста в Spotify
RecommendationsDto	Класс, хранящий список рекомендуемых треков, полученный от Spotify
TopArtistsDto	Класс, хранящий список наиболее прослушиваемых исполнителей пользователя, полученный от Spotify
TopTracksDto	Класс, хранящий список наиболее прослушиваемых треков пользователя, полученный от Spotify
TrackDto	Класс, хранящий данные о треке, полученные от Spotify
TrackFeaturesDto	Класс, хранящий список музыкальных характеристик нескольких треков, полученный от Spotify
SpotifyController	Класс, позволяющий взаимодействовать с сервисом Spotify
SpotifyService	Сервис, используемый для доступа к Api Spotify
GenresDto	Модель, хранящая жанры и их частоту в наиболее прослушиваемых треках пользователя.
StatisticsService	Сервис, используемый при получении статистики пользователя
StatisticsController	Класс, позволяющий получать статистику по аккаунту Spotify пользователя

User	Класс, хранящий данные о пользователе Spotify
UsersRepository	Репозиторий для взаимодействия с пользователями в базе данных
UserService	Сервис, позволяющий взаимодействовать со списком пользователей

Разработка

Реализация серверного API

В качестве описания программного интерфейса был выбран инструмент, поддерживающий стандарт OAS 3.0 – Swagger. Далее представлена полученная документация API полученная автоматически по директивам, указанным в декораторах различных методов и структурах данных внутри разрабатываемой информационной системы.

Spotify

Direct Spotify endpoints

^

GET

/api/spotify/top

World Top Tracks

🔒 ^

Retrieves world top tracks using authenticated user

Parameters

Try it out

Name	Description
limit integer(\$int32) (query)	<div>limit</div>

Responses

Code	Description	Links
400	Invalid parameters <div><div>Media type</div><div>*/*</div><div>Example Value Schema</div><div><pre>{ "status": 0, "message": "string"} </pre></div></div>	No links
401	User not authorized <div><div>Media type</div><div>*/*</div><div>Example Value Schema</div><div><pre>{ "status": 0, "message": "string"} </pre></div></div>	No links
403	Not enough permissions <div><div>Media type</div><div>*/*</div><div>Example Value Schema</div><div><pre>{ "status": 0, "message": "string"} </pre></div></div>	No links
default	Object holding list of tracks <div><div>Media type</div><div>*/*</div><div>Controls Accept header.</div><div>Example Value Schema</div><div><pre>{ "items": [{ "track": { "album": { "total_tracks": 0, "id": "string", "images": [{ "url": "string" }], "name": "string", "genres": ["string"], "artists": [{ "name": "string", "id": "string", "genres": ["string"] }, { "name": "string", "id": "string", "genres": ["string"] }], "images": [{ "url": "string" }] } }]]</pre></div></div>	No links

Рисунок 3. Интерфейс модуля Spotify

Playlists

Modify Spotify playlists

⌵

POST

/api/playlists/restore

Restore playlist

🔒 ⌵

Restores already created playlist to Spotify

Parameters

Try it out

Name	Description
id * required string (query)	<div>id</div>

Responses

Code	Description	Links
401	User not authorized <div>Media type */*</div> <div>Example Value Schema</div> <div><pre>{ "status": 0, "message": "string"}</pre></div>	No links
403	Not enough permissions <div>Media type */*</div> <div>Example Value Schema</div> <div><pre>{ "status": 0, "message": "string"}</pre></div>	No links
404	Playlist not found <div>Media type */*</div> <div>Example Value Schema</div> <div><pre>{ "status": 0, "message": "string"}</pre></div>	No links
default	Object holding response status <div>Media type */*</div> <div>Controls Accept header.</div> <div>Example Value Schema</div> <div><pre>{ "status": 0, "message": "string"}</pre></div>	No links

POST

/api/playlists/create/world

Create World Top playlist

🔒 ⌵

POST

/api/playlists/create/personal

Create Personal Top Tracks playlist

🔒 ⌵

POST

/api/playlists/create/artists

Create Personal Artist Top playlist

🔒 ⌵

GET

/api/playlists/playlists

All playlists

🔒 ⌵

Рисунок 4. Интерфейс модуля создания плейлистов

Statistics

Get Spotify statistics of authenticated user

^

GET

/api/statistics/tracks

Top tracks

🔒 ^

Retrieves authenticated User top tracks

Parameters

Try it out

Name	Description
limit	<div>limit</div>
<code>integer(\$int32)</code>	<code>(query)</code>

Responses

Code	Description	Links
401	User not authorized <div>Media type<div>*/*</div></div> <div>Example Value Schema</div> <div><pre>{ "status": 0, "message": "string"}</pre></div>	No links
403	Not enough permissions <div>Media type<div>*/*</div></div> <div>Example Value Schema</div> <div><pre>{ "status": 0, "message": "string"}</pre></div>	No links
default	List of top tracks <div>Media type<div>*/*</div></div> <div>Controls Accept header.</div> <div>Example Value Schema</div> <div><pre>[{ "album": { "total_tracks": 0, "id": "string", "images": [{ "url": "string" }], "name": "string", "genres": ["string"], "artists": [{ "name": "string", "id": "string", "genres": ["string"], "images": [{ "url": "string" }] }] }]</pre></div>	No links

GET

/api/statistics/profile

Music profile

🔒 v

GET

/api/statistics/genres

Top genres

🔒 v

GET

/api/statistics/artists

Top artists

🔒 v

Рисунок 5. Интерфейс модуля получения статистики

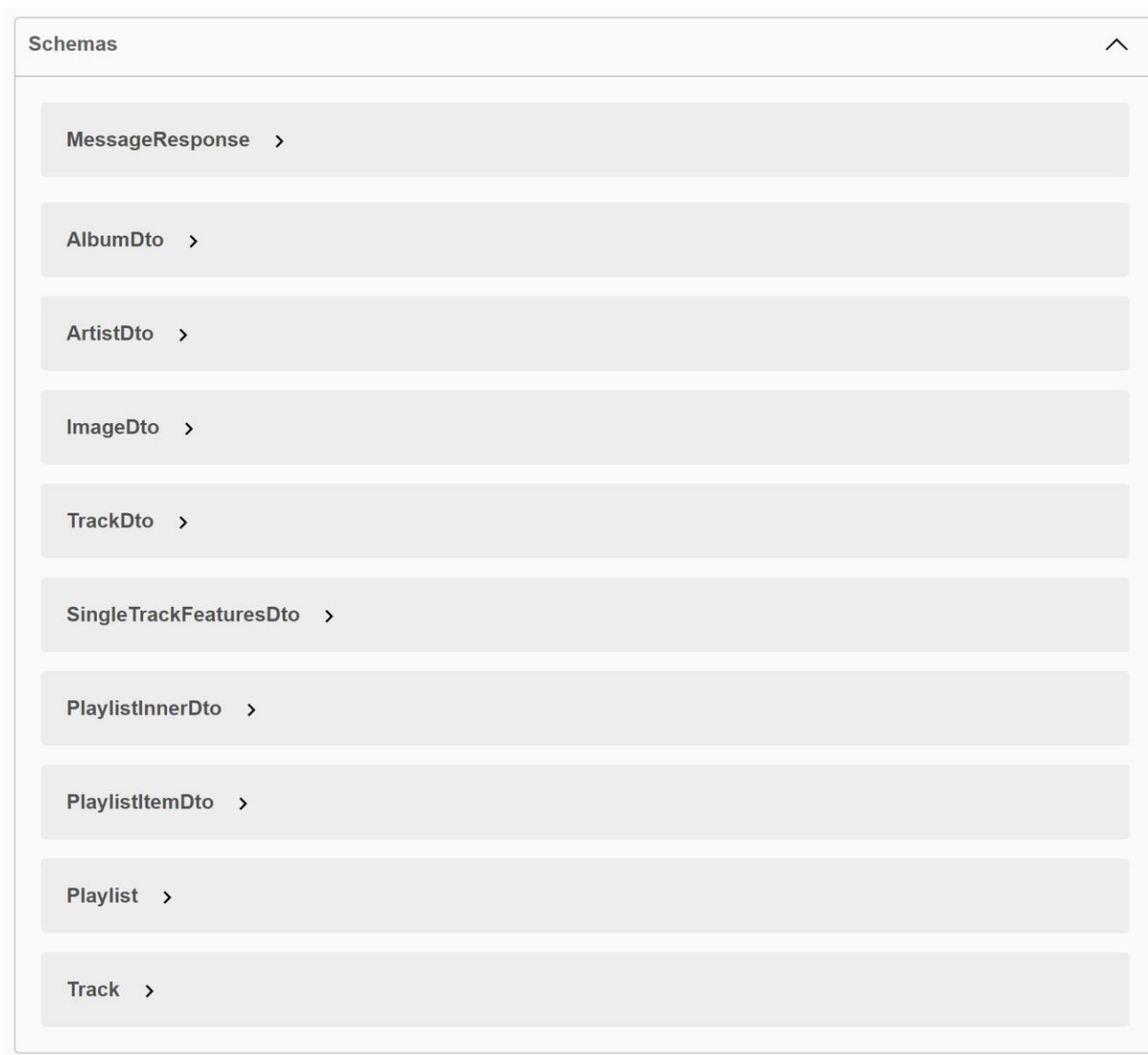


Рисунок 6. Используемые схемы ответов

Реализация пользовательского интерфейса

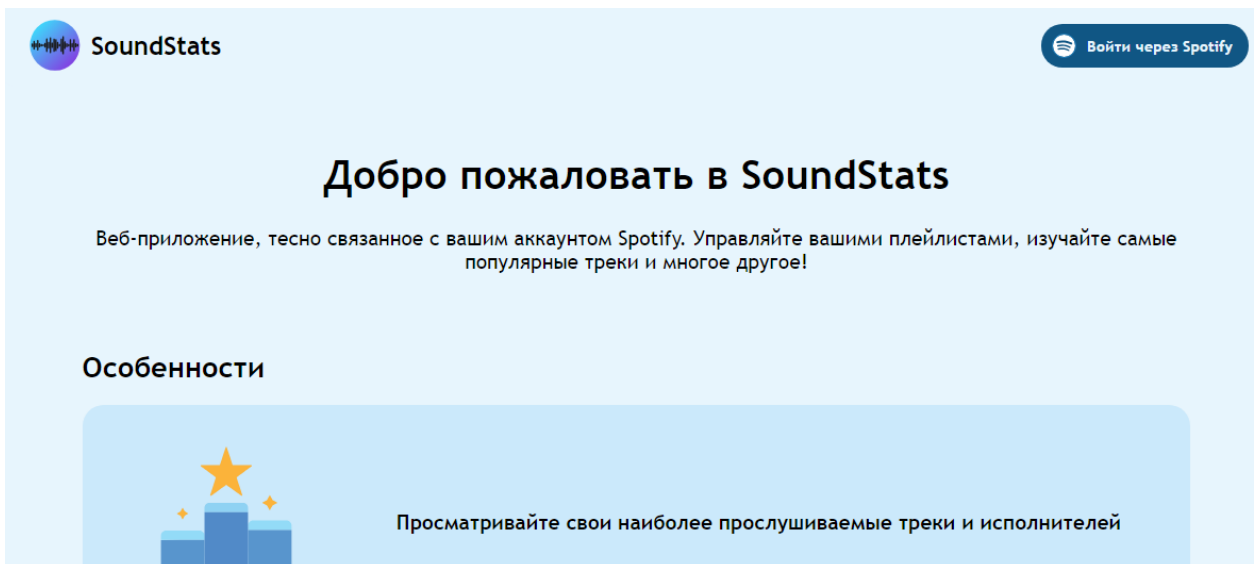


Рисунок 7. Главная страница не аутентифицированного пользователя

Для не аутентифицированного пользователя единственная доступная опция – нажатием на кнопку в правом верхнем углу пройти аутентификацию и авторизацию в сервисе используя аккаунт Spotify

The screenshot shows the Spotify login page. At the top is the heading "Войти в Spotify" in a large, bold, white font. Below it are three rounded rectangular buttons for social login: "Войти через Google" with the Google logo, "Войти через Facebook" with the Facebook logo, and "Войти через Apple" with the Apple logo. Below these is a horizontal line. Then, there are two input fields: "Электронная почта или имя пользователя" (Email or username) and "Пароль" (Password). Below the password field is a toggle switch labeled "Запомнить меня" (Remember me), which is currently turned on. At the bottom is a large, rounded rectangular button labeled "Войти" (Login) in white text on a dark background.

Рисунок 8. Страница входа

При нажатии на кнопку, либо при переходе на защищенную страницу пользователю предлагается, в соответствии с протоколом OAuth2, пройти аутентификацию в Spotify и предоставить необходимые разрешения

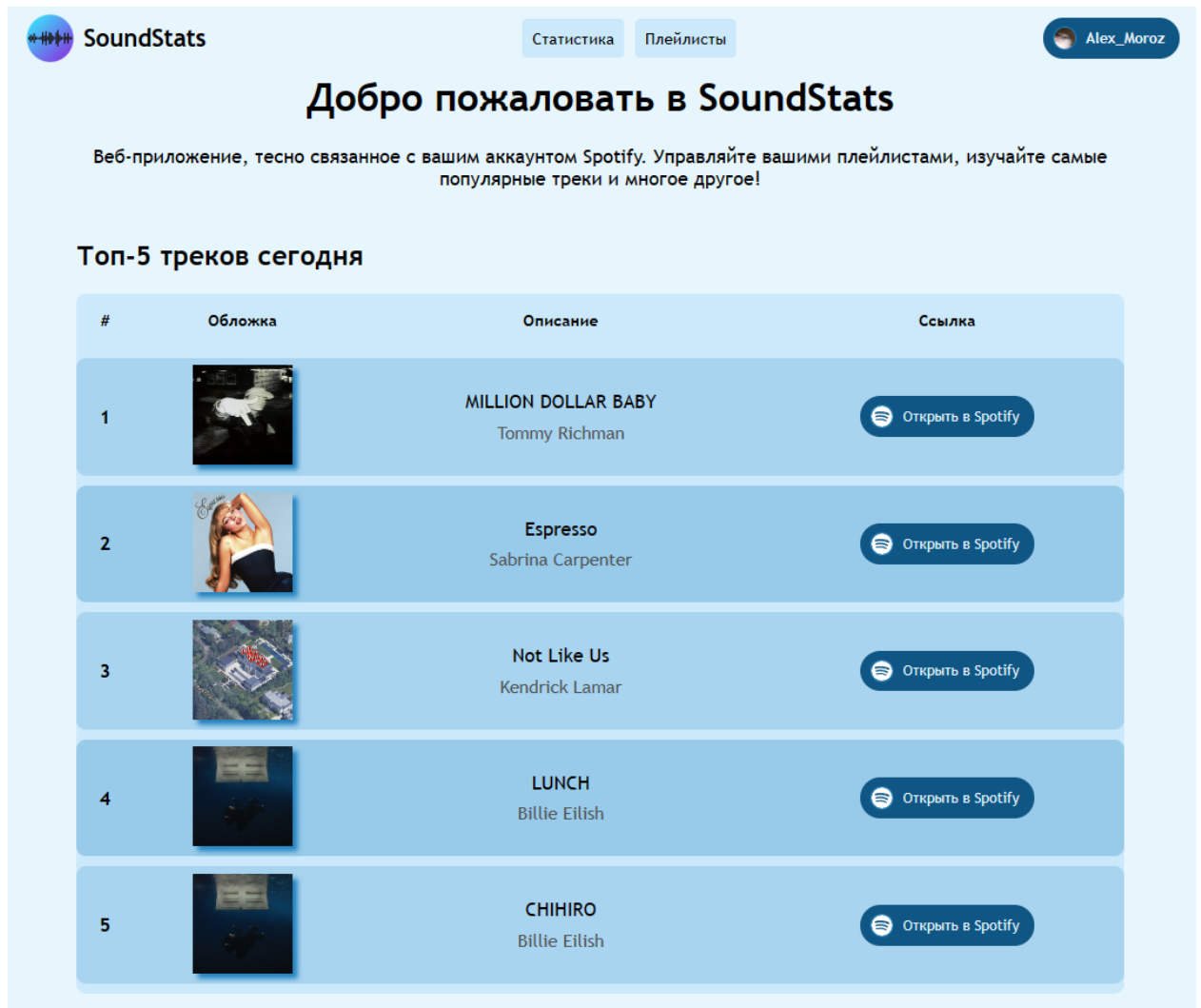


Рисунок 9. Главная страница после аутентификации

После прохождения аутентификации и авторизации пользователь возвращается на главную страницу, где он может получить первые 5 треков из мирового рейтинга треков за текущий день

Профиль

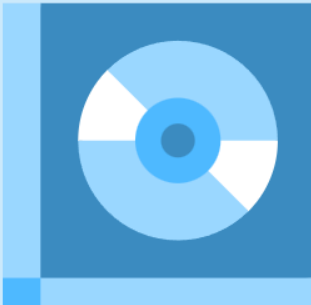


Alex_Moroz

Выйти

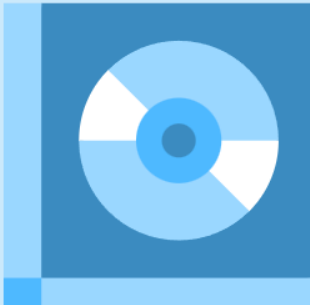
Ваши плейлисты

Ваши плейлисты, когда-либо созданные через сервис. Нажатием на плейлист вы можете восстановить его



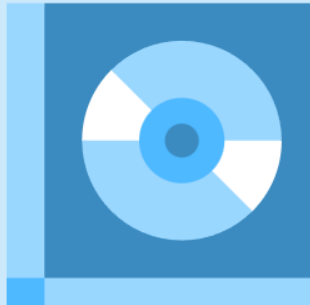
Artists Top
21:37:03 11.05.2024

Восстановить



World Top
21:37:04 11.05.2024

Восстановить



Personal Top
21:37:04 11.05.2024

Восстановить

Рисунок 10. Профиль пользователя

При нажатии аутентифицированным пользователем на кнопку в правом верхнем углу откроется его профиль, где он может просматривать свои ранее сгенерированные плейлисты и при желании восстанавливать их. При нажатии на кнопку «Выйти» пользователь выйдет из аккаунта и вернется на главную страницу

Статистика

Здесь вы сможете просматривать свою статистику по вашим трекам, исполнителям, любимым жанрам, а также подписываться на ваши самые прослушиваемые треки и исполнителей

Любимые исполнители



Foreign
Gnomes



Открыть в
Spotify



Sati Akura



Открыть в
Spotify



CthulhuSeeker



Открыть в
Spotify



Aviators



Открыть в
Spotify



Battle Tapes



Открыть в
Spotify

Анализ предпочтений в музыке

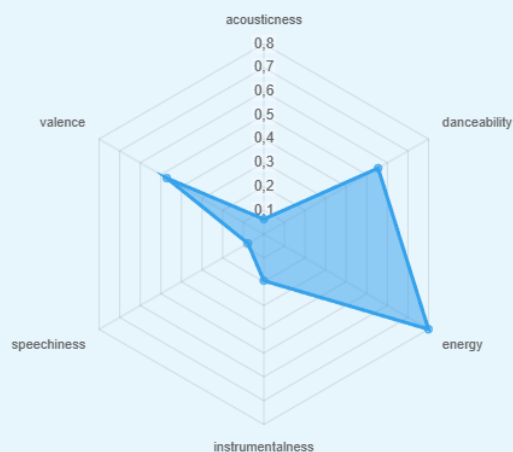


Рисунок 11. Страница статистики. Любимые исполнители и музыкальный профиль

Страница статистики имеет следующий вид. Здесь пользователь может просмотреть список своих наиболее прослушиваемых исполнителей и свой музыкальный профиль






Любимые треки			
#	Обложка	Описание	Ссылка
1		The Man Who Sold the World - 2010 Remaster Midge Ure	Открыть в Spotify
2		Heavy - Lolita as Roxy Foreign Gnomes, KITCALIBER	Открыть в Spotify
3		City Ruins - Shade (NieR:Automata) - Synthwave Arrangement CthulhuSeeker, Kaobnir	Открыть в Spotify
4		Be The Legend TIA RAY, Rtruenahmean, League of Legends	Открыть в Spotify
5		Lemon Tree Fools Garden	Открыть в Spotify

Рисунок 12. Страница статистики. Любимые треки

Здесь пользователь может увидеть список своих наиболее прослушиваемых треков

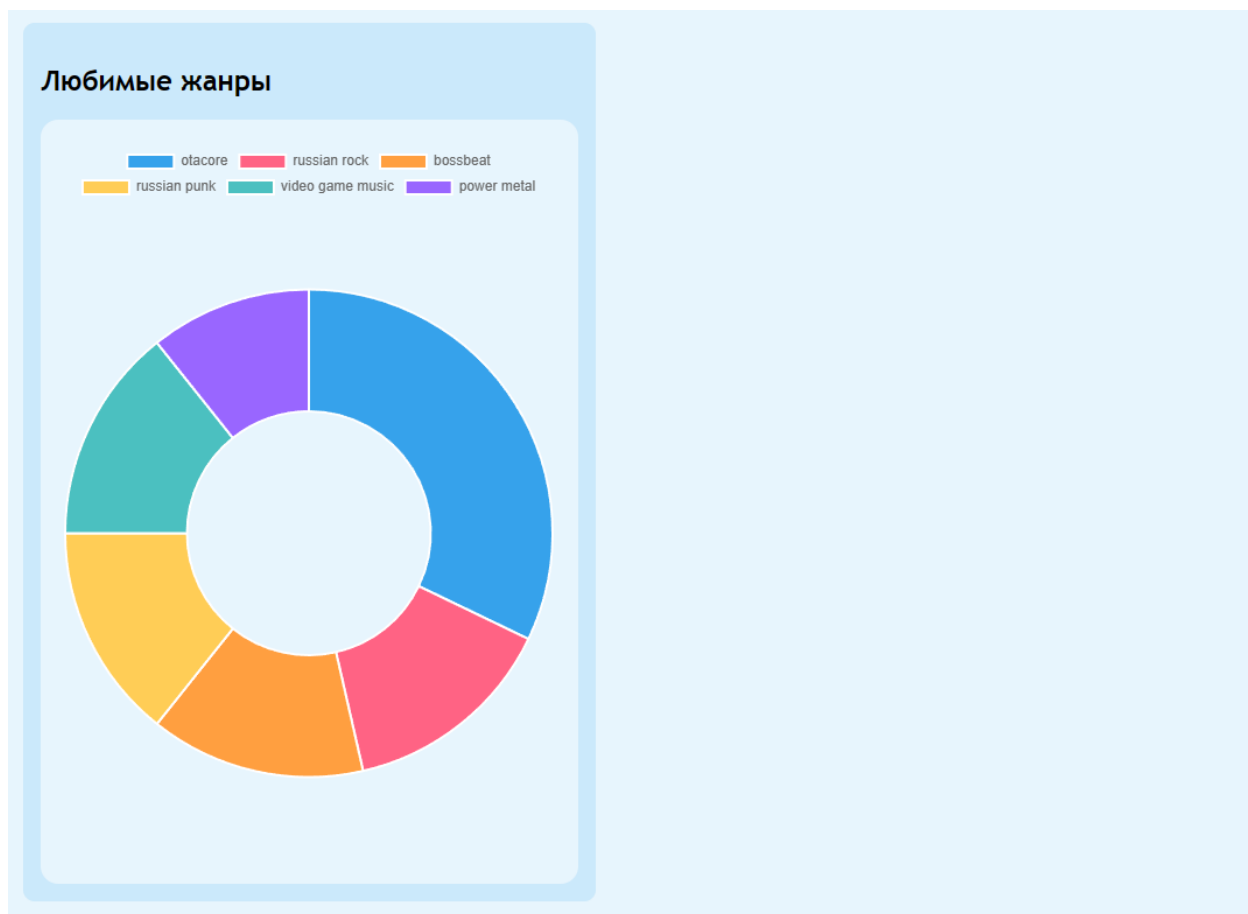


Рисунок 13. Страница статистики. Любимые жанры

Здесь пользователь увидит свою статистику по наиболее прослушиваемым жанрам

Создание плейлистов

На данной странице вы сможете генерировать новые плейлисты на основе как уже знакомых вам треков, так и еще не прослушанных используя самые разные алгоритмы

Стандартные алгоритмы

Персональные рекомендации Spotify

Базовые рекомендации от алгоритмов Spotify на основе 50 Ваших наиболее прослушиваемых треков за последние 6 месяцев. В будущем может появиться настройка параметров

Создать

Рекомендации на основе самых популярных треков

Название говорит само за себя - позволяет создавать плейлист на основе самых популярных треков в мире, используя алгоритмы Spotify. Зачем - понятия не имею, почему бы и нет

Создать

Рекомендации по любимым исполнителям

Позволяет создавать плейлисты с рекомендациями по Вашим любимым исполнителям. Использует алгоритмы Spotify

Создать

Рисунок 14. Страница создания плейлистов

На данной странице пользователь может сгенерировать плейлист и сохранить его в своих аккаунтах Spotify и данного сервиса

Заключение

В ходе выполнения курсовой работы был проведён анализ работы классических веб-приложений, исходя из которого были выявлены и сформированы требования к разрабатываемому веб-приложению.

Исходя из выбранной архитектуры и наложенных ограничений были сформированы требования к используемым технологиям внутри модулей. Была спроектирована архитектура данных, программная и системная архитектура в виде набора диаграмм в нотации UML.

Опираясь на выше изложенные требования и стек технологий было разработано веб-приложение и пользовательский интерфейс в рамках дисциплины «Web-программирование».

Таким образом, все поставленные ранее цели были выполнены.

Разработанное приложение является результатом данной курсовой работы.

Список использованной литературы

1. Мартин Фаулер - Архитектура корпоративных программных приложений. Издательский дом "Вильямс". 2006 г.
2. Kotlin Docs | Kotlin Documentation [Электронный ресурс]. – Режим доступа: <https://kotlinlang.org/docs/home.html>. – Дата доступа: 23.05.2024.
3. Spring Framework Documentation :: Spring Framework [Электронный ресурс]. – Режим доступа: <https://docs.spring.io/spring-framework/reference/>. – Дата доступа: 23.05.2024
4. Baeldung [Электронный ресурс]. – Режим доступа: <https://www.baeldung.com/>. Дата доступа: 23.05.2024