Министерство науки и высшего образования Российской Федерации Федеральное государственное бюджетное образовательное учреждение высшего образования

ТОМСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ СИСТЕМ

УПРАВЛЕНИЯ И РАДИОЭЛЕКТРОНИКИ

(ТУСУР)

Кафедра экономической математики, информатики и статистики (ЭМИС)

ОБЩЕЕ ОПИСАНИЕ СИСТЕМЫ

Отчет по практической работе №1 по дисциплине

«Технология программирования»

Студент гр. з-39-2

\_\_\_\_\_\_\_\_\_\_\_\_ / В.О. Михайленко

« » 20 г

Проверил

Ассистент каф. ЭМИС

\_\_\_\_\_\_\_\_\_\_\_\_ / Я.В. Костелей

« \_\_ » \_\_\_\_\_\_\_ 20\_\_ г

Томск 2023

# Анализ предметной области

Пользователь может взаимодействовать с проигрывателем музыки, используя пользовательский интерфейс приложения или веб-страницы. Чтобы выбрать плейлист для воспроизведения, пользователь может просмотреть список доступных плейлистов и выбрать нужный из них, либо создать новый плейлист. При выборе плейлиста, соответствующая информация о треках будет загружена из базы данных и отображена в пользовательском интерфейсе. Далее пользователь может выбрать конкретный трек из списка и начать его воспроизведение. Проигрыватель также может предоставлять дополнительные функции, такие как поиск, фильтрацию и сортировку треков внутри плейлиста. Все изменения, вносимые пользователем в плейлист, будут храниться в базе данных для последующего использования.

# Глоссарий

Таблица 1 – Глоссарий проекта

|  |  |
| --- | --- |
| Термин предметной области | Описание |
| Пользователь | Человек, который пользуется музыкальным проигрывателем. |
| Музыкальный проигрыватель | Программное обеспечение, предназначенное для воспроизведения музыки. |
| Плейлист | Список музыкальных треков, отсортированных в определенном порядке для последующего воспроизведения. |
| База данных | Организованная структура данных, используемая для хранения, управления и доступа к информации. |
| Пользовательский интерфейс | Часть приложения, которая обеспечивает взаимодействие пользователя с системой. |
| Воспроизведение | Процесс проигрывания музыкального трека. |
| Трек | Аудиозапись, содержащая музыкальную композицию или звуковые эффекты. |

# Нефункциональные требования к системе

1. Производительность: система должна быть способна обрабатывать большие объемы данных и быстро загружать треки с сервера.

2. Надежность: система должна быть стабильной и не иметь сбоев в работе приложения или базы данных, чтобы пользователь мог слушать музыку без прерываний.

3. Безопасность: система должна обеспечивать защиту личных данных пользователей, а также защиту от возможности злоумышленников получить доступ к базе данных и изменять ее содержимое.

4. Удобство использования: система должна обладать интуитивно понятным интерфейсом, который облегчит пользователю поиск и проигрывание музыки.

5. Масштабируемость: система должна позволять добавлять новые функции и увеличивать ее мощность по мере роста количества пользователей и объема данных.

6. Поддержка различных форматов аудио файлов: проигрыватель должен поддерживать различные форматы музыкальных файлов, чтобы пользователь мог воспроизводить любимую музыку в нужном формате.

7. Доступность: система должна быть доступна пользователям на различных устройствах (компьютеры, мобильные телефоны, планшеты) и иметь возможность работать в автономном режиме.

# Диаграмма вариантов использования и описание потоков событий

Для системы «проигрыватель музыки, который хранит плейлисты в файловой базе данных» можно определить следующие действующие лица:

- Пользователь (User) - Человек, который пользуется музыкальным проигрывателем.

- База данных (Database) - Организованная структура данных, используемая для хранения, управления и доступа к плейлистам.

На основе описания актеров можно определить следующие варианты использования (прецеденты):

- *add track* – добавить трек в плейлист;

- *play track* – воспроизвести трек;

- *delete track* – удаление трека из плейлиста;

- *create playlist –* создать плейлист;

- *select playlist* – выбрать плейлист;

- *edit playlist* – редактировать плейлист;

*- delete playlist* – удалить плейлист из базы данных.

На основе описания действующих лиц и вариантов использования можно определить диаграмму вариантов использования (Рисунок 1).

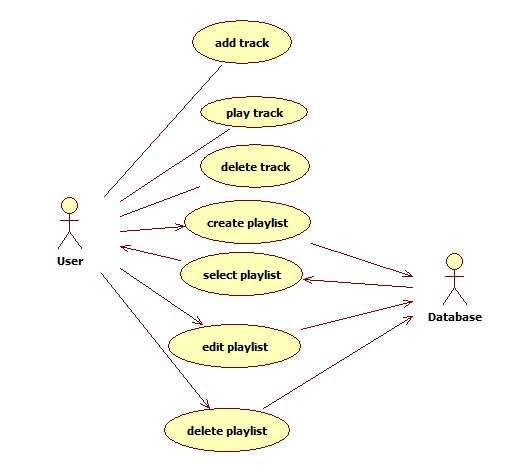


Рисунок 1 – Диаграмма вариантов использования

Определим потоки событий для каждого варианта использования:

1) *play track*:

- Пользователь выбирает трек, который хочет проиграть.

- Система загружает трек из базы данных и начинает его воспроизведение на устройстве пользователя.

2) *add track*:

- Пользователь открывает плейлист, в котором хочет добавить трек.

- Пользователь выбирает опцию "Добавить трек".

- Система отображает список доступных треков из базы данных.

- Пользователь выбирает нужный трек и подтверждает добавление в плейлист.

- Система добавляет выбранный трек в плейлист.

3) *create playlist*:

- Пользователь выбирает опцию "Создать новый плейлист".

- Система отображает окно для создания нового плейлиста.

- Пользователь вводит название и описание плейлиста.

- Пользователь сохраняет новый плейлист в базу данных.

- Система сохраняет новый плейлист в базу данных.

4) *select playlist*:

- Пользователь выбирает опцию "Выбрать плейлист".

- Система отображает список доступных плейлистов.

- Пользователь выбирает нужный плейлист.

- Система загружает содержимое выбранного плейлиста из базы данных.

5) *delete playlist*:

- Пользователь выбирает опцию "Удалить плейлист".

- Система отображает список доступных плейлистов.

- Пользователь выбирает плейлист, который нужно удалить.

- Система запрашивает подтверждение удаления от пользователя.

- Если пользователь подтверждает удаление, система удаляет плейлист из базы данных.

6) *edit playlist*:

- Пользователь выбирает плейлист, который нужно изменить.

- Пользователь выбирает опцию "Редактировать плейлист".

- Система отображает окно редактирования плейлиста с возможностью изменения названия и описания.

- Пользователь вносит изменения в информацию о плейлисте.

- Пользователь подтверждает изменения.

- Система сохраняет обновленную информацию о плейлисте в базе данных.

7) *delete track*:

- Пользователь выбирает плейлист, из которого нужно удалить трек.

- Пользователь выбирает опцию "Редактировать плейлист".

- Система отображает список всех треков, находящихся в выбранном плейлисте.

- Пользователь выбирает трек, который нужно удалить.

- Пользователь нажимает кнопку "Удалить трек".

- Система запрашивает подтверждение удаления от пользователя.

- Если пользователь подтверждает удаление, система удаляет выбранный трек из плейлиста.

# Соглашение по моделированию

Для данного моделирования были использованы следующие соглашения.

1. Имена вариантов использования задают короткими глагольными фразами. Для каждого варианта использования необходимо создать пакет *Use-Case Realization*, включающий, во-первых, по крайней мере одну реализацию варианта использования.

2. Имена классов (и актеров) должны быть существительными, соответствующими по возможности понятиям предметной области (глоссарию проекта).

3. Имена классов (и актеров) начинают с заглавной буквы.

4. Имена атрибутов и операций (и вариантов использования) начинают со строчной буквы.

5. Составные имена должны быть сплошными, без подчеркиваний, каждое отдельное слово следует начинать с заглавной буквы.

# Создание диаграмм последовательности

На рисунке 2 изображена диаграмма последовательности для потока создания плейлиста. После нажатия на кнопку создания плейлиста (onCreateClick()) в форме, происходит проверка пустое ли поле имени плейлиста (checkIfEmpty()). Далее выполняется попытка соединения с локальной базой данных (tryConnectDB()), и имя введенное пользователем проверяется на совпадение с уже существующими в базе данных (checkIfNameExist()). Если совпадений нет – добавляем плейлист в базу данных, и выводим его на экран функцией ShowPlaylist()

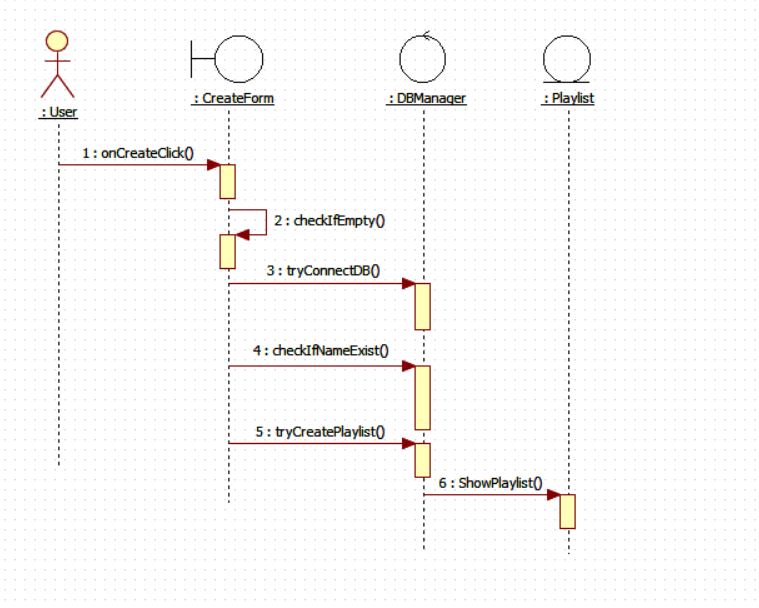


Рисунок 2 – Поток создания плейлиста

На рисунке 3 изображен альтернативный поток, в котором произошло совпадение по имени плейлиста в базе данных. В таком случае, когда функция checkIfNameExist() возвращает True, класс CreateForm возвращает ошибку пользователю returnError().

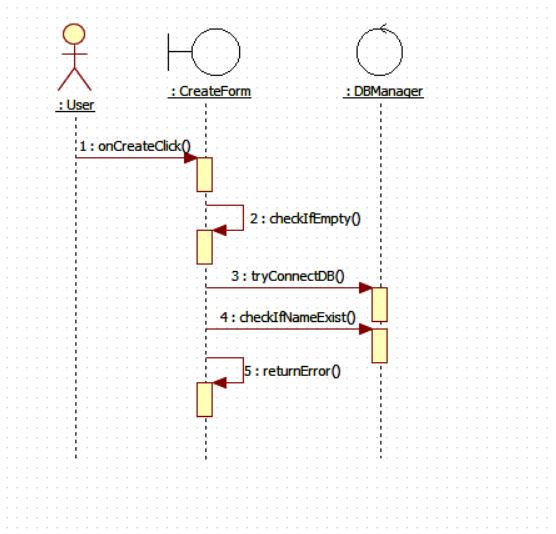


Рисунок 3 – Поток ошибки создания плейлиста

# Разработка диаграммы классов на уровне сущностей

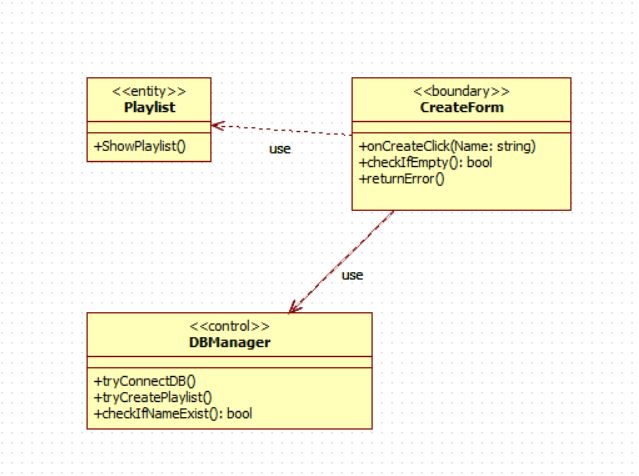
Диаграмма классов, для проекта «музыкальный плеер с сохранением плейлистов в локальную базу данных» представлен на рисунке 4.

Рисунок 4 – Диаграмма классов

Для диаграммы были определены связи между классами. Так как от класса CreateForm зависят классы Playlist и DBManager - был выбран тип связи Зависимость (dependency).

# Генерация кода

Через меню в **Tools-C#-Generate Code** была выполнена генерация кода на C#. В результате генерации были получены файлы:

CreateForm.cs

Playlist.cs

DBManager.cs

Содержимое файлов представлено на рисунках 5, 6 и 7.



Рисунок 5 – Класс CreateFrom



Рисунок 6 – Класс Playlist

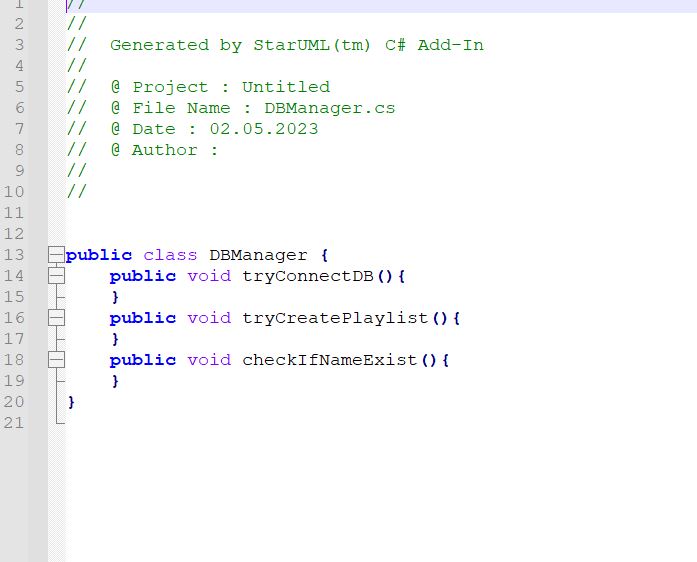


Рисунок 7 – Класс DBManager

# Разработка диаграммы компонентов

Диаграмма компонентов представлена на рисунке 8. В ней отображен состав системных компонентов и зависимости между ними.

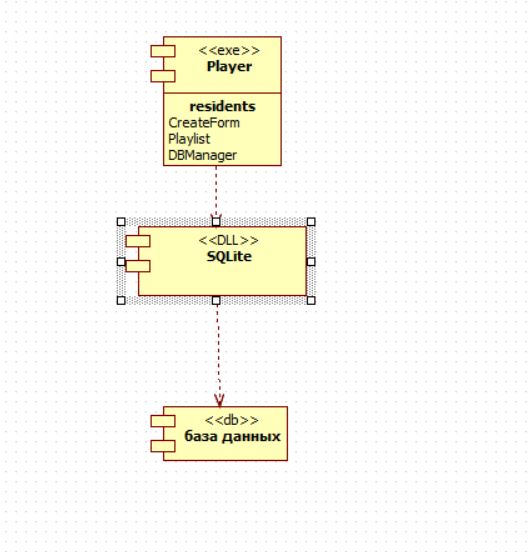


Рисунок 8 – Диаграмма компонентов

# Разработка диаграммы размещения

На рисунке 9 представлена диаграмма размещения, исходя из которой видим, что все компоненты размещены приложения могут быть размещены на одном ПК.

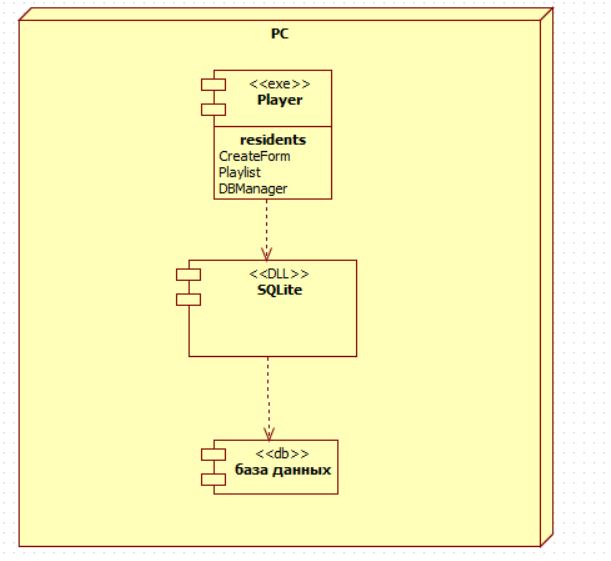


Рисунок 9 – Диаграмма размещения

# Выводы

В результате выполнения практических работ закреплен теоретический материал, изложенный в курсе, направленный на изучения проектирования информационных систем по описанию предметной области с помощью диаграмм различного типа. Проведено моделирование предметной области.