МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

Учреждение образования «БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ

ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ»

Факультет Информационных технологий

Кафедра Информационные системы и технологии

Специальность 1–40 01 01 Программное обеспечение информационных технологий

**ПОЯСНИТЕЛЬНАЯ ЗАПИСКА**

**К КУРСОВОЙ РАБОТЕ НА ТЕМУ:**

Web-приложение «Издательская платформа»

Выполнил студент Пузиков Алексей Алексеевич

(Ф.И.О.)

Руководитель работы ассист. Нистюк Ольга Александровна (учен. степень, звание, должность, Ф.И.О., подпись)

И.о. зав. кафедрой ст. преп. Блинова Е.А.

(учен. степень, звание, должность, Ф.И.О., подпись)

Курсовая работа защищена с оценкой

Минск 2024

Содержание

[Введение 4](#_Toc181088891)

[1 Постановка задачи и обзор аналогичных решений 5](#_Toc181088892)

[1.1 Аналитический обзор аналогов 5](#_Toc181088893)

[1.1.1 Интернет-ресурс «Quiz» 5](#_Toc181088894)

[1.1.2 Интернет-ресурс «Kahoot!» 6](#_Toc181088895)

[1.1.3 Интернет-ресурс «Jackbox Games» 7](#_Toc181088896)

[1.2 Постановка задачи 8](#_Toc181088897)

[1.3 Средства разработки 8](#_Toc181088898)

[2 Проектирование web-приложения 10](#_Toc181088899)

[2.1 Архитектура приложения 10](#_Toc181088900)

[2.2 Проектирование структуры базы данных 10](#_Toc181088901)

[3 Реализация web-приложения 14](#_Toc181088902)

[3.1 Реализация серверной части приложения 14](#_Toc181088903)

[3.1.1 Слой доступа к данным 14](#_Toc181088904)

[3.1.2 Слой бизнес-логики 15](#_Toc181088905)

[3.1.3 Слой представления (API) 17](#_Toc181088906)

[3.1.4 API Gateway 18](#_Toc181088907)

[3.1.5 Дополнительные библиотеки 18](#_Toc181088908)

[3.2 Разработка клиентской части приложения 19](#_Toc181088909)

[4 Тестирование web-приложения 20](#_Toc181088910)

[4.1 Пункт 20](#_Toc181088911)

[5 Руководство пользователя 21](#_Toc181088912)

[5.1 Вход в аккаунт 21](#_Toc181088913)

[5.2 Регистрация пользователя 21](#_Toc181088914)

[5.3 Взаимодействие с профилем пользователя 21](#_Toc181088915)

[Заключение 22](#_Toc181088916)

[Список используемых источников 23](#_Toc181088917)

[Приложение А 24](#_Toc181088918)

[Приложение Б 25](#_Toc181088919)

[Приложение В 26](#_Toc181088920)

[Приложение Г 27](#_Toc181088921)

[Приложение Д 29](#_Toc181088922)

Введение

В современном мире цифровых технологий и глобальной коммуникации всё больше людей участвуют в интеллектуальных играх, как в формате живого взаимодействия, так и онлайн. Эти игры не только развивают мышление, но и способствуют активному общению, социальному взаимодействию и обмену знаниями. В то же время, платформы с элементами социальной сети предоставляют пользователям возможность обмениваться впечатлениями, обсуждать свои достижения и результаты игр.

В рамках данного курсового проекта будет разработано web-приложение, которое объединяет функционал для проведения интеллектуальных игр и элементы социальной сети. Цель – создать удобное и интуитивно понятное web-приложение, которое позволит пользователям участвовать в интеллектуальных играх, отслеживать свою статистику, делиться результатами и общаться с другими участниками. Для достижения этой цели необходимо выполнить следующие задачи:

* провести анализ предметной области и определить основные сущности и атрибуты системы;
* разработать архитектуру приложения, используя трёхуровневую модель (слой доступа к данным, слой бизнес-логики, слой представления);
* реализовать ключевые модули системы, включая модули аутентификации, управления играми, профилей и статистики;
* разработать API Gateway для маршрутизации запросов к соответствующим микросервисам и обеспечения их взаимодействия;
* создать клиентскую часть приложения на основе React, обеспечив удобный интерфейс для взаимодействия с пользователем;
* наполнить базу данных тестовыми данными, такими как пользователи, игры и результаты;
* разработать функции и методы для реализации бизнес-логики системы, включая системы уведомлений, рейтинг игроков, комментарии и другие элементы социальной сети.

Данный проект направлен на улучшение опыта участников интеллектуальных игр и создание социальной среды, способствующей обмену знаниями и поддержанию интереса к играм.

1 Постановка задачи и обзор аналогичных решений

* 1. Аналитический обзор аналогов

В современном мире платформы для интеллектуальных игр и организации социальных взаимодействий играют важную роль в создании сообществ, обмене знаниями и проведении досуга. Такие платформы позволяют пользователям не только участвовать в играх, но и делиться результатами, обсуждать свои достижения и взаимодействовать с другими участниками, что создает активное и мотивирующее игровое сообщество. В данном разделе будут рассмотрены существующие интернет-ресурсы и web-приложения, которые предоставляют возможности для организации интеллектуальных игр с элементами социальной сети.

* + 1. Интернет-ресурс «Quiz»

Quiz – это онлайн-платформа, предназначенная для создания и прохождения интерактивных викторин. Пользователи могут выбирать из широкого ассортимента готовых викторин или создавать собственные, используя разнообразные настройки для персонализации. Платформа ориентирована на развлечение и обучение, что делает её популярной как среди учащихся, так и в корпоративной среде.

Интерфейс интернет-ресурса «Quiz» представлен на рисунке 1.1.

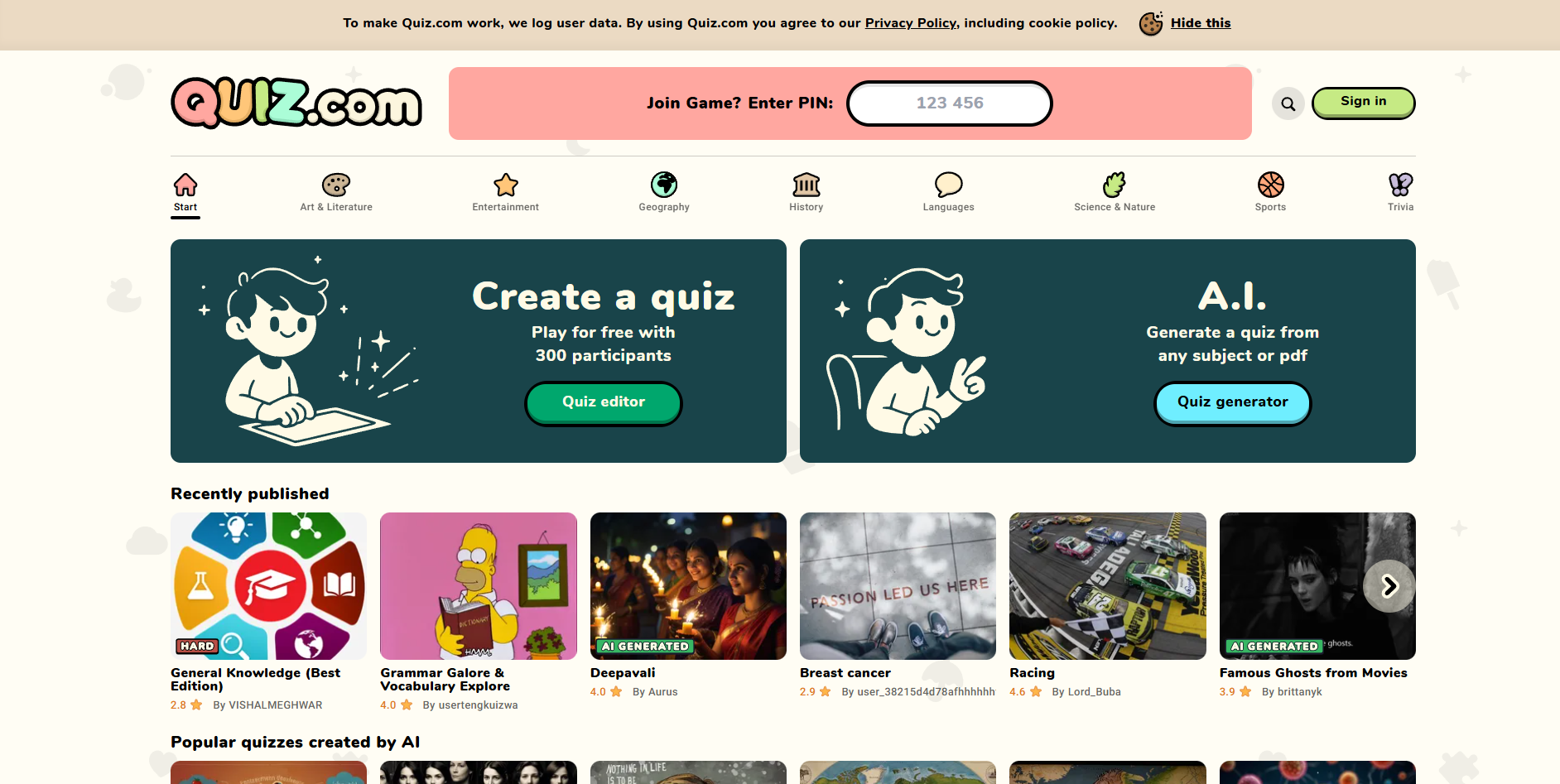


Рисунок 1.1 – Интерфейс интернет-ресурса «Quiz»

Достоинства:

* Широкий выбор тем: Quiz предлагает викторины на самые разные темы, от науки и искусства до поп-культуры и спорта, что позволяет привлечь интересы широкого круга пользователей.
* Возможность создания собственных викторин: Пользователи могут легко создавать уникальные тесты, что делает платформу удобной для преподавателей и организаторов мероприятий.
* Интерактивность и мгновенные результаты: Платформа предоставляет мгновенные результаты, что делает процесс прохождения викторин более увлекательным и познавательным.

Недостатки:

* Ограниченные социальные функции: Платформа имеет ограниченные возможности для взаимодействия пользователей вне процесса прохождения викторин.
* Зависимость от интернета: Quiz требует стабильного интернет-соединения, что может быть неудобным в условиях ограниченного доступа к сети.
  + 1. Интернет-ресурс «Kahoot!»

Kahoot! – образовательная платформа для создания викторин, которая часто используется в школах и университетах для проведения интерактивных занятий. Она ориентирована на группы участников, что делает её идеальной для обучения и совместных мероприятий.

Интерфейс интернет-ресурса «Kahoot!» представлен на рисунке 1.2.

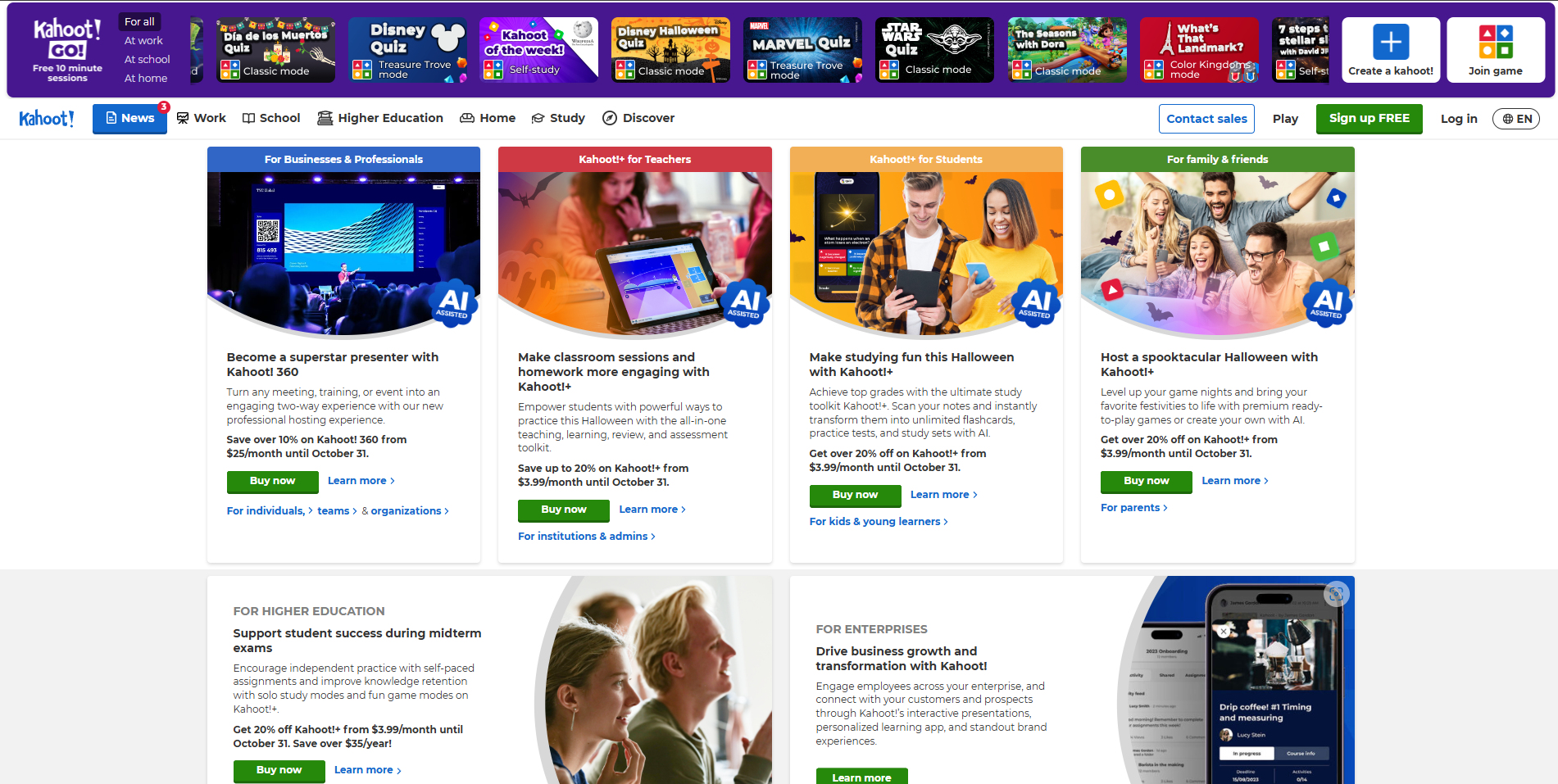


Рисунок 1.2 – Интерфейс интернет-ресурса «Kahoot!»

Достоинства:

* Поддержка живых игр: Kahoot! позволяет проводить викторины в режиме реального времени, что делает процесс участия более динамичным и интерактивным.
* Ориентированность на образование: Платформа широко используется в образовательных учреждениях для тестирования знаний и вовлечения учеников.
* Простота создания игр: Пользователи могут легко создавать свои викторины и адаптировать их под нужды учебного процесса.

Недостатки:

* Ограниченные возможности персонализации: Несмотря на возможность создания собственных викторин, пользователи не могут изменять многие аспекты интерфейса или игровых сценариев.
* Фокус на группе: Платформа больше подходит для групповых мероприятий, чем для индивидуального использования.
  + 1. Интернет-ресурс «Jackbox Games»

Jackbox Games – это набор многопользовательских игр, которые можно играть на различных устройствах с участием большого количества людей. Игры отличаются оригинальными сценариями и креативными задачами, ориентированными на взаимодействие с друзьями.

Интерфейс интернет-ресурса «Jackbox Games» представлен на рисунке 1.3.

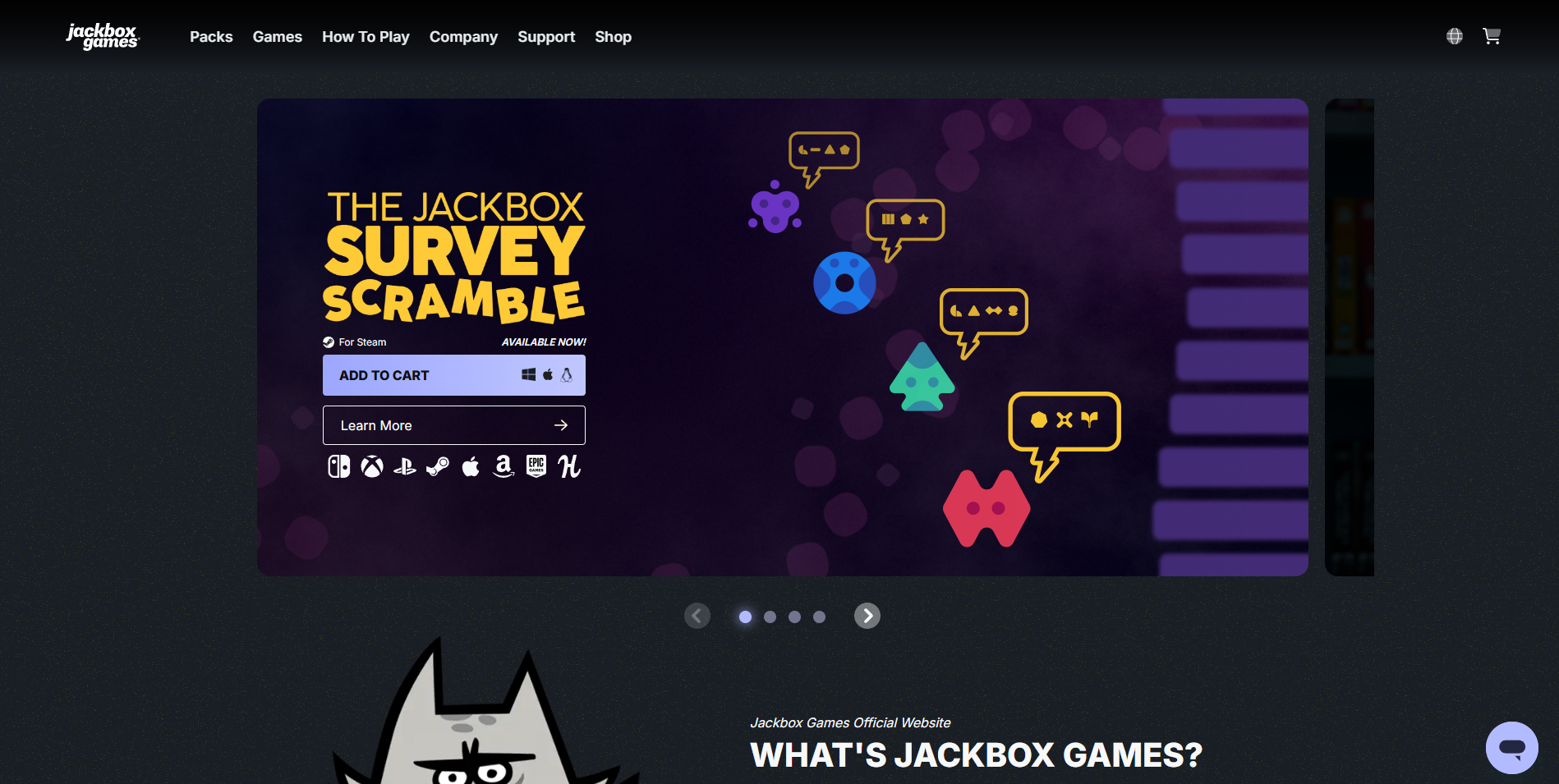


Рисунок 1.3 – Интерфейс интернет-ресурса «Jackbox Games»

Достоинства:

* Веселые и креативные игры: Jackbox Games предлагает оригинальные игровые сценарии, которые делают каждый игровой сеанс уникальным.
* Поддержка большого числа участников: Играть в Jackbox Games можно в больших компаниях, что делает его идеальным для вечеринок и социальных мероприятий.
* Мульти-платформенность: Игры можно запускать на любом устройстве, а участники подключаются через свои смартфоны.

Недостатки:

* Отсутствие социальных функций: Платформа не предоставляет возможностей для взаимодействия пользователей вне игрового процесса.
* Ограниченное количество игр: Несмотря на разнообразие игр в каждом наборе, их количество ограничено, и игроки могут быстро потерять интерес.
  1. Постановка задачи

Анализ аналогов позволяет выявить сильные и слабые стороны существующих платформ для проведения интеллектуальных игр и определить ключевые требования к разработке программного средства в рамках данного курсового проекта. Web-приложение должно выполнять следующие функции:

* поддерживать роль «Гость»:
  + обеспечивать возможность регистрации;
* поддерживать роль «Пользователь»:
  + обеспечивать возможность авторизации и аутентификации;
  + обеспечивать возможность присоединения к играм;
  + предоставлять возможность просмотра личной статистики;
  + обеспечивать возможность просмотра рейтингов игроков;
  + обеспечивать возможность комментирования игр;
  + предоставлять возможность управления профилем;
  + обеспечивать возможность отправки приглашений другим пользователям для участия в игре;
* поддерживать роль «Ведущий»:
  + обеспечить возможность создания игровых комнат;
  + обеспечивать возможность выбора категорий и сложности вопросов при создании игры;
  + обеспечивать возможность комментирования игр.

Диаграмма вариантов использования представлена в приложении А.

* 1. Средства разработки

Для разработки программного средства, предназначенного для сбора статистики и организации интеллектуальных игр с элементами социальной сети, будут использованы следующие инструменты и технологии:

* Интегрированная среда разработки Visual Studio – для создания и отладки кода на языке C#;
* Язык программирования C# – для разработки серверной части приложения;
* ASP.NET Core Web API – для создания REST-сервиса, обеспечивающего взаимодействие с клиентской частью и другими компонентами системы;
* React (TypeScript) – для разработки клиентской части приложения с использованием современного фреймворка для создания интерактивных пользовательских интерфейсов;
* Redis – для кеширования данных и увеличения производительности системы;
* MS SQL Server – для хранения информации о пользователях, играх, статистике и других данных;
* Shadcn UI – библиотека компонентов для создания удобных и современных пользовательских интерфейсов.

Использование данных технологий предоставляет несколько важных преимуществ при разработке программного средства:

* Visual Studio обеспечивает удобную и мощную среду для разработки на C#, предоставляя инструменты для отладки, анализа кода и повышения производительности разработки.
* C# и ASP.NET Core Web API обеспечивают высокую производительность и расширяемость серверной части, что позволяет эффективно обрабатывать запросы пользователей и обеспечивать высокую нагрузку.
* React с TypeScript помогает создавать динамичные и отзывчивые пользовательские интерфейсы, позволяя пользователям взаимодействовать с системой в реальном времени.
* Redis обеспечивает быструю обработку данных и минимизирует задержки при частом доступе к информации, что особенно важно для системы с высокой нагрузкой, связанной с интеллектуальными играми.
* MS SQL Server используется как надежное и производительное хранилище данных, обеспечивая целостность информации и быструю работу с большими объемами данных.

Web-приложение должно быть выполнено с использованием асинхронного программирования, взаимодействовать с базой данных, реализовано под разными платформами. Web-приложение должно представлять собой приложение с асинхронным UI. Отображение, бизнес логика и хранилище данных должны быть максимально независимы друг от друга для возможности расширения. Диаграмму вариантов использования разработать на основе UML, также необходимо разработать логическую схему базы данных и структурную схему приложения. Язык разработки проекта C#, платформа «.NET». Развёртывание конечного приложения для последующего использования должно осуществляться с использованием Docker. Web-приложение должно быть логически завершенным. Управление программой должно быть интуитивно понятным и удобным. Листинги проекта должны содержать комментарии.

Исходя из описанных выше технологий, можно сделать вывод о том, что для реализации данного проекта были использованы современные инструменты и технологии, что позволило повысить эффективность работы. В целом, использование таких технологий позволило реализовать проект на высоком уровне и обеспечить его успешное выполнение.

В данной главе были сформулированы основные требования к приложению на основе преимуществ и недостатков некоторых аналогов.

2 Проектирование web-приложения

2.1 Функционалньость web-приложения

В системе предусмотрены различные типы пользователей с различными уровнями доступа и возможностями. Для лучшего понимания взаимодействия ролей и функций, была построена UseCase диаграмма, которая отражает основные действия пользователей в приложении. Она отображает ключевые сценарии использования, охватывая все важные процессы и действия. Диаграмма представлена на рисунке 2.1.

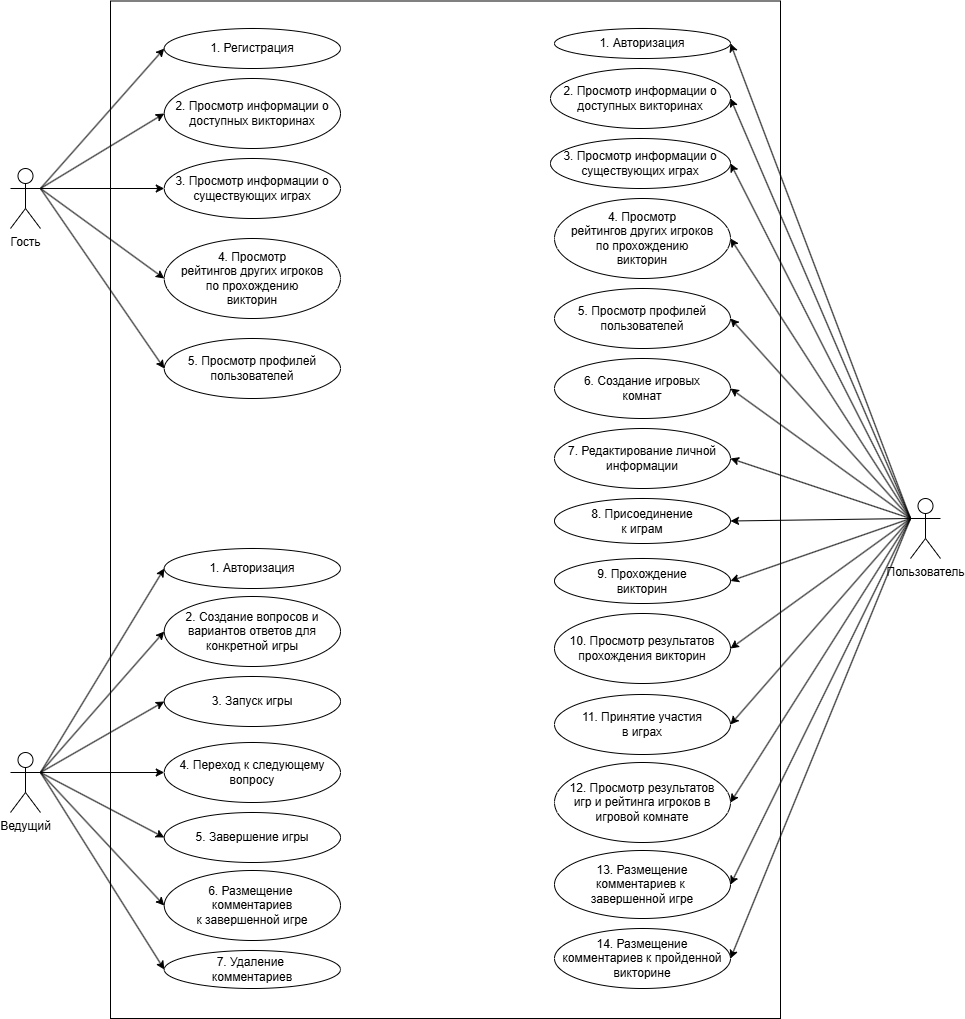


Рисунок 2.1 – Диаграмма UseCase

В системе предусмотрены следующие роли: Гость, Пользователь и Ведущий. Каждая из этих ролей имеет определённый набор прав и ограничений, в зависимости от своей функциональности. В таблице 2.1 представлено описание ролей.

Таблица 2.1 – Описание ролей

|  |  |
| --- | --- |
| Роль | Описание роли |
| Гость | Пользователь, не авторизованный в системе, имеющий доступ к ограниченному функционалу. Может просматривать информацию о викторинах, играх, рейтингах и профилях пользователей. Не может создавать игровые комнаты или участвовать в играх. |
| Пользователь | Авторизованный пользователь, имеющий расширенные права. Может просматривать информацию о викторинах, играх, профилях пользователей, создавать игровые комнаты, редактировать личную информацию, участвовать в играх, проходить викторины, оставлять комментарии и просматривать результаты. |
| Ведущий | Пользователь с правами администратора для проведения игр. Может создавать вопросы и варианты ответов, запускать и завершать игру, переходить между вопросами, а также управлять комментариями к завершенным играм. Ведущий несет ответственность за управление игровым процессом. |

Далее, в таблицах расписаны функции, доступные для каждой роли в системе. В таблице 2.2 представлены функции для роли Гость

Таблица 2.2 – Описание функциональности для роли Гость

|  |  |
| --- | --- |
| Номер | Описание функции |
| 1 | Регистрация в системе |
| 2 | Просмотр информации о доступных викторинах |
| 3 | Просмотр информации о существующих играх |
| 4 | Просмотр рейтингов других игроков по прохождению викторин |
| 5 | Просмотр профилей пользователей |

В таблице 2.3 представлены функции для роли Пользователь.

Таблица 2.3 – Описание функциональности для роли Пользователь

|  |  |
| --- | --- |
| Номер | Описание функции |
| 1 | Авторизация в системе |
| 2 | Просмотр информации о доступных викторинах |
| 3 | Просмотр информации о существующих играх |
| 4 | Просмотр рейтингов других игроков по прохождению викторин |
| 5 | Просмотр профилей пользователей |
| 6 | Создание игровых комнат |
| 7 | Редактирование личной информации |
| 8 | Присоединение к играм |
| 9 | Прохождение викторин |
| 10 | Просмотр результатов прохождения викторин |
| 11 | Принятие участия в играх |
| 12 | Просмотр результатов игр и рейтинга игроков в игровой комнате |
| 13 | Размещение комментариев к завершенной игре |
| 14 | Размещение комментариев к пройденной викторине |

В таблице 2.4 представлены функции для роли Ведущий.

Таблица 2.4 – Описание функциональности для роли Ведущий

|  |  |
| --- | --- |
| Номер | Описание функции |
| 1 | Авторизация в системе |
| 2 | Создание вопросов и вариантов ответов для конкретной игры |
| 3 | Запуск игры |
| 4 | Переход к следующему вопросу |
| 5 | Завершение игры |
| 6 | Размещение комментариев к завершенной игре |
| 7 | Удаление комментариев |

2.2 Проектирование структуры базы данных

Структура базы данных состоит из семи таблиц, каждая из которых отвечает за хранение и управление определенными данными. Таблица 2.1 предоставляет краткую информацию о каждой из них, включая название, назначение и основные функции. Эти таблицы взаимодействуют между собой с помощью внешних ключей, обеспечивая целостность данных и поддержку ключевых функциональных возможностей системы, таких как управление пользователями, хранение викторин, вопросов и ответов, а также отслеживание прогресса и результатов.

Таблица 2.1 – Структура базы данных

|  |  |
| --- | --- |
| Название таблицы | Описание использования |
| AnswerOptions | Хранит варианты ответов для вопросов, включая текст ответа, его корректность и связь с вопросами. |
| CompletedAnswers | Хранит данные об ответах игроков на вопросы, включая информацию о выбранном варианте и его корректности. |
| GameComments | Содержит комментарии пользователей к играм, включая текст комментария, имя пользователя, аватар и дату. |
| GameResults | Хранит результаты игр для пользователей, включая количество правильных ответов и общее количество вопросов. |
| Games | Содержит информацию об играх, таких как название, ведущий, время начала/окончания, статус, приватность и пароль. |
| GameStatuses | Хранит возможные статусы игр (например, "Активная", "Завершённая"). |
| Notifications | Содержит уведомления для пользователей, включая заголовок, текст сообщения, дату создания и статус прочтения. |
| Players | Хранит данные о игроках игр, включая имя, счёт, связь с игрой и пользователем. |
| Questions | Содержит текст вопросов для игр, а также информацию о кастомных вопросах. |
| QuizAnswerOptions | Хранит варианты ответов для викторин, включая текст ответа и информацию о корректности. |
| QuizComments | Содержит комментарии пользователей к викторинам, включая текст комментария, имя пользователя и дату. |
| QuizQuestions | Хранит вопросы викторин с текстом и привязкой к конкретной викторине. |
| QuizResults | Содержит результаты прохождения викторин, включая количество правильных ответов, время завершения и пользователя. |
| Quizzes | Хранит информацию о викторинах, включая название, обложку, время начала/окончания. |
| Users | Хранит информацию о пользователях |

Логическая схема базы данных представлена в приложении В.

AnswerOptions – хранит варианты ответов для вопросов, включая текст ответа, его корректность и связь с вопросами. Её структура представлена в таблице 2.2.

Таблица 2.2 – Структура таблицы AnswerOptions

|  |  |  |
| --- | --- | --- |
| Столбец | Тип данных | Ограничение целостности |
| Id | INT | PRIMARY KEY IDENTITY |
| Text | NVARCHAR(200) | NOT NULL |
| IsCorrect | BIT | NOT NULL |
| QuestionId | INT | FOREIGN KEY, ON DELETE CASCADE |

CompletedAnswers – хранит данные об ответах игроков на вопросы, включая информацию о выбранном варианте и его корректности. Структура представлена в таблице 2.3.

Таблица 2.3 – Структура таблицы CompletedAnswers

|  |  |  |
| --- | --- | --- |
| Название | Тип данных | Ограничения целостности |
| Id | INT | PRIMARY KEY, IDENTITY(1,1) |
| GameId | INT | NOT NULL, FOREIGN KEY |
| PlayerId | INT | NOT NULL, FOREIGN KEY |
| QuestionId | INT | NOT NULL, FOREIGN KEY |
| SelectedOptionId | INT | NOT NULL, FOREIGN KEY |
| IsCorrect | BIT | NOT NULL |

GameComments – содержит комментарии пользователей к играм, включая текст комментария, имя пользователя, аватар и дату. Структура представлена в таблице 2.4.

Таблица 2.4 – Структура таблицы GameComments

|  |  |  |
| --- | --- | --- |
| Столбец | Тип данных | Ограничение целостности |
| Id | INT | PRIMARY KEY IDENTITY |
| GameId | INT | FOREIGN KEY, ON DELETE CASCADE |
| Username | NVARCHAR(MAX) | NOT NULL |
| UserId | INT | NOT NULL |
| Content | NVARCHAR(500) | NOT NULL |
| CommentDate | DATETIME2(7) | NOT NULL |
| ProfileImageUrl | NVARCHAR(MAX) | NOT NULL |

GameResults – хранит результаты игр для пользователей, включая количество правильных ответов и общее количество вопросов. Структура представлена в таблице 2.5.

Таблица 2.5 – Структура таблицы GameResults

|  |  |  |
| --- | --- | --- |
| Название | Тип данных | Ограничения целостности |
| Id | INT | PRIMARY KEY, IDENTITY(1,1) |
| NumberOfCorrectAnswers | INT | NOT NULL |
| NumberOfQuestions | INT | NOT NULL |
| UserId | INT | NOT NULL, FOREIGN KEY |
| GameId | INT | NOT NULL, FOREIGN KEY |

Games – содержит информацию об играх, таких как название, ведущий, время начала/окончания, статус, приватность и пароль. Её структура представлена в таблице 2.6.

Таблица 2.6 – Структура таблицы Games

|  |  |  |
| --- | --- | --- |
| Столбец | Тип данных | Ограничение целостности |
| Id | INT | PRIMARY KEY IDENTITY |
| GameName | NVARCHAR(100) | NOT NULL |
| HostId | INT | NOT NULL |
| CreateTime | DATETIME2(7) | NOT NULL |
| StartTime | DATETIME2(7) | NOT NULL |
| EndTime | DATETIME2(7) | NOT NULL |
| MaxPlayers | INT | NOT NULL |
| GameCoverImageUrl | NVARCHAR(MAX) | NOT NULL |
| IsPrivate | BIT | NOT NULL |
| RoomPassword | NVARCHAR(50) | NOT NULL |
| GameStatusId | INT | FOREIGN KEY |

GameStatuses – содержит возможные статусы игр. Структура представлена в таблице 2.7.

Таблица 2.7 – Структура таблицы GameStatuses

|  |  |  |
| --- | --- | --- |
| Столбец | Тип данных | Ограничение целостности |
| Id | INT | PRIMARY KEY |
| Name | NVARCHAR(100) | NOT NULL |

Notifications – содержит уведомления для пользователей, включая заголовок, текст сообщения, дату создания и статус прочтения. Структура представлена в таблице 2.8.

Таблица 2.8 – Структура таблицы Notifications

|  |  |  |
| --- | --- | --- |
| Название | Тип данных | Ограничения целостности |
| Id | INT | PRIMARY KEY, IDENTITY(1,1) |
| UserId | INT | NOT NULL, FOREIGN KEY |
| Title | NVARCHAR(100) | NOT NULL |
| Message | NVARCHAR(1000) | NOT NULL |
| CreatedAt | DATETIME2(7) | NOT NULL |
| IsRead | BIT | NOT NULL (DEFAULT 0) |

Players – Хранит данные о игроках игр, включая имя, счёт, связь с игрой и пользователем. Структура таблицы представлена в таблице 2.9.

Таблица 2.9 – Структура таблицы Players

|  |  |  |
| --- | --- | --- |
| Столбец | Тип данных | Ограничение целостности |
| Id | INT | PRIMARY KEY IDENTITY |
| GameId | INT | FOREIGN KEY, ON DELETE CASCADE |
| Name | NVARCHAR(100) | NOT NULL |
| Score | INT | NOT NULL |
| UserId | INT | FOREIGN KEY |

Questions – содержит информацию о вопросах, связанных с играми. Структура представлена в таблице 2.10.

Таблица 2.10 – Структура таблицы Questions

|  |  |  |
| --- | --- | --- |
| Столбец | Тип данных | Ограничение целостности |
| Id | INT | PRIMARY KEY IDENTITY |
| Text | NVARCHAR(500) | NOT NULL |
| GameId | INT | FOREIGN KEY, ON DELETE CASCADE |
| IsCustom | BIT | NOT NULL |

QuizAnswerOptions – хранит варианты ответов для викторин, включая текст ответа и информацию о корректности. Структура представлена в таблице 2.11.

Таблица 2.11 – Структура таблицы QuizAnswerOptions

|  |  |  |
| --- | --- | --- |
| Столбец | Тип данных | Ограничение целостности |
| Id | INT | PRIMARY KEY IDENTITY |
| Text | NVARCHAR(200) | NOT NULL |
| IsCorrect | BIT | NOT NULL |
| QuizQuestionId | INT | FOREIGN KEY, ON DELETE CASCADE |

QuizComments – Содержит комментарии пользователей к викторинам, включая текст комментария, имя пользователя и дату. Структура представлена в таблице 2.12.

Таблица 2.12 – Структура таблицы QuizComments

|  |  |  |
| --- | --- | --- |
| Название | Тип данных | Ограничение целостности |
| Id | INT | PRIMARY KEY, IDENTITY |
| QuizId | INT | FOREIGN KEY REFERENCES Quizzes(Id) ON DELETE CASCADE |
| UserId | INT | NOT NULL |
| Content | NVARCHAR (500) | NOT NULL |
| Username | NVARCHAR (max) | NOT NULL |
| ProfileImageUrl | NVARCHAR (max) | NOT NULL |
| CommentDate | DATETIME2(7) | NOT NULL |

QuizQuestions – содержит вопросы викторины. Структура представлена в таблице 2.13.

Таблица 2.13 – Структура таблицы QuizQuestions

|  |  |  |
| --- | --- | --- |
| Столбец | Тип данных | Ограничение целостности |
| Id | INT | PRIMARY KEY IDENTITY |
| Text | NVARCHAR(500) | NOT NULL |
| QuizId | INT | FOREIGN KEY, ON DELETE CASCADE |

QuizResults – таблица для хранения результатов викторины. Структура представлена в таблице 2.14.

Таблица 2.14 – Структура таблицы QuizResults

|  |  |  |
| --- | --- | --- |
| Столбец | Тип данных | Ограничение целостности |
| Id | INT | PRIMARY KEY IDENTITY |
| NumberOfCorrectAnswers | INT | NOT NULL |
| NumberOfQuestions | INT | NOT NULL |
| EndTime | DATETIME2(7) | NOT NULL |
| UserId | INT | FOREIGN KEY, ON DELETE CASCADE |
| QuizId | INT | FOREIGN KEY, ON DELETE CASCADE |

Quizzes – хранит информацию о викторинах, связанных с играми. Структура представлена в таблице 2.15.

Таблица 2.15 – Структура таблицы Quizzes

|  |  |  |
| --- | --- | --- |
| Столбец | Тип данных | Ограничение целостности |
| Id | INT | PRIMARY KEY IDENTITY |
| GameName | NVARCHAR(200) | NOT NULL |
| GameCoverImageUrl | NVARCHAR(500) | NOT NULL |
| StartTime | DATETIME2(7) | NOT NULL |
| EndTime | DATETIME2(7) | NOT NULL |

Users – хранит информацию о пользователях, зарегистрированных в системе. Структура представлена в таблице 2.16.

Таблица 2.16 – Структура таблицы Users

|  |  |  |
| --- | --- | --- |
| Столбец | Тип данных | Ограничение целостности |
| Id | INT | PRIMARY KEY IDENTITY |
| Email | NVARCHAR(MAX) | NOT NULL |
| Username | NVARCHAR(MAX) | NULL |
| Password | NVARCHAR(MAX) | NULL |
| IsEmailVerified | BIT | NOT NULL |
| IsActive | BIT | NOT NULL |
| FirstName | NVARCHAR(MAX) | NULL |
| LastName | NVARCHAR(MAX) | NULL |
| ProfileImageUrl | NVARCHAR(MAX) | NULL |

В данной главе была спроектирована архитектура приложения, включая серверную и клиентскую часть, а также базы данных. Скрипт создания всех таблиц представлен в приложении Г.

2.3 Архитектура приложения

Приложение представляет собой веб-приложение, использующее трехуровневую архитектуру, состоящую из следующих слоев: слой доступа к данным, слой бизнес-логики и слой репрезентации.

Слой доступа к данным и слой бизнес-логики организованы как библиотеки классов C#. В слое доступа к данным находятся модели данных, класс контекста Entity Framework, интерфейсы репозиториев и их реализации. Для хранения данных используется MS SQL Server. Слой бизнес-логики содержит DTO-объекты, интерфейсы сервисов и их реализации для различных задач, таких как сервис уведомлений, регистрация, сервисы профилей и другие. Слой репрезентации представлен совокупностью ASP.NET Core Web API приложений, включающих контроллеры, использующие методы сервисов для обработки запросов от клиентов. В качестве кеширования данных используется Redis, API Gateway (Ocelot) применяется для организации централизованной точки входа в backend, а Nginx используется для управления трафиком.

Для клиентской части приложения используется React с TypeScript и UI-фреймворк ShadCN UI. Все компоненты приложения разворачиваются с использованием Docker Compose на платформе Ubuntu. Разработка проводилась на операционной системе Windows 11, а тестирование — в браузере Microsoft Edge. Для разработки серверной и клиентской части использовались Visual Studio 2022 и WebStorm.

Операционные системы: Ubuntu 22.04 LTS для развертывания приложения и Windows 11 для разработки.

Серверные технологии: ASP.NET Core 7.0, C# 11.0, Entity Framework Core 7.0, Microsoft SQL Server 2019 для базы данных, Redis 7.0 для кеширования, Ocelot 17.0 для API Gateway, Nginx 1.22.1 для управления трафиком.

Клиентская сторона: React 18.2.0, TypeScript 5.2.0, ShadCN UI 1.0.0 для создания интерфейса пользователя.

Инструменты разработки: Visual Studio 2022 для разработки серверной и клиентской частей, WebStorm 2022 для разработки и тестирования фронтенда.

Контейнеризация и оркестрация: Docker 24.0.0 для контейнеризации и Docker Compose 2.17.0 для оркестрации.

Браузеры: Microsoft Edge 116 для тестирования приложения.

Протоколы: HTTP/HTTPS для обмена данными между клиентом и сервером, WebSockets для реального времени.

Диаграмма развертывания программного средства представлена в приложении Б.

3 Реализация web-приложения

3.1 Реализация серверной части приложения

Проект реализован на основе трехуровневой архитектуры, что обеспечивает разделение функциональности на независимые слои: слой доступа к данным, слой бизнес-логики и слой представления. Данное архитектурное решение позволяет увеличить гибкость и масштабируемость системы, упростить поддержку кода, а также повысить отказоустойчивость.

3.1.1 Слой доступа к данным

Слой доступа к данным в проекте играет ключевую роль в управлении и хранении данных, обеспечивая безопасное и эффективное взаимодействие с базой данных. Этот слой спроектирован как библиотека классов на языке C#, а в качестве ORM используется Entity Framework с подходом Code First. Это упрощает поддержку структуры данных и позволяет гибко адаптировать её по мере изменения бизнес-требований.

Классы моделей представляют объекты базы данных, описывающие основные сущности и их взаимосвязи. Они используют атрибуты и методы конфигурации Entity Framework для указания типа данных, ограничений, зависимостей и других характеристик. Это позволяет Entity Framework автоматически создавать соответствующие таблицы и связи в базе данных. Базовые поля для всех моделей и их общее поведение определены в абстрактном классе BaseModel, что упрощает создание моделей, наследующих такие базовые свойства, как Id, дата создания и модификации.

Для реализации паттерна "репозиторий" в слое доступа к данным выделены интерфейсы и соответствующие классы репозиториев. Репозитории предоставляют стандартный набор методов CRUD (создание, чтение, обновление, удаление), а также могут включать специфичные для приложения методы, упрощающие взаимодействие с данными. Базовая функциональность всех репозиториев объединена в абстрактный класс BaseRepository, который предоставляет общие методы для работы с Entity Framework и может быть расширен для реализации более специализированных операций.

Настройки для моделей данных и их отображения в базе данных определены с использованием интерфейса IEntityTypeConfiguration<T>. Классы конфигурации управляют характеристиками, такими как индексы, внешние ключи, уникальные ограничения, а также типы данных, используемые в базе данных. Этот подход позволяет отделить настройки базы данных от логики модели, сохраняя чистоту кода и увеличивая гибкость.

Entity Framework Code First позволяет управлять миграциями для обновления структуры базы данных при изменениях в модели.. Благодаря миграциям процесс поддержания базы данных в актуальном состоянии становится предсказуемым и управляемым, а история изменений фиксируется и доступна для отката при необходимости.

Взаимодействие данных между слоями приложения происходит через репозитории, обеспечивающие контролируемый доступ к данным, что позволяет скрыть детали взаимодействия с базой данных.

3.1.2 Слой бизнес-логики

Слой бизнес-логики в проекте отвечает за реализацию функциональности и бизнес-процессов, обеспечивая взаимодействие между слоем данных и представлением. Этот слой также создан в виде библиотеки классов на языке C#, что обеспечивает его модульность и независимость. Все функции и процессы, необходимые для работы приложения, сосредоточены в сервисах, которые обеспечивают основную бизнес-логику, а также позволяют централизованно управлять правилами и обработкой данных.

В рамках слоя бизнес-логики реализованы интерфейсы и классы сервисов, которые инкапсулируют основные операции над данными и взаимодействуют с репозиториями для выполнения CRUD-операций. Интерфейсы определяют набор обязательных методов и контрактов для работы с бизнес-логикой, что делает код легко расширяемым и позволяет создавать новые сервисы без изменения существующих. Каждый сервис представлен в виде отдельного класса, который реализует интерфейс и содержит конкретную бизнес-логику, необходимую для обработки данных. Пример интерфейса сервиса авторизации представлен в листинге 3.1.

|  |
| --- |
| public interface ILoginService  {  Task<TokenDto> AuthenticateAsync(LoginDto loginDto, CancellationToken cancellationToken = default);  Task<TokenDto> RefreshTokenAsync(string refreshToken, CancellationToken cancellationToken = default);  public string GenerateRefreshToken(User user);  public string GenerateAccessToken(User user);  } |

Листинг 3.1 – Код интерфейса сервиса авторизации

В слое бизнес-логики также используются DTO (Data Transfer Object) объекты, которые служат для передачи данных между слоями и обеспечивают защиту внутренней структуры данных от внешнего вмешательства. DTO объекты описывают только те свойства, которые необходимы для конкретных операций, и помогают избегать избыточной передачи информации. Чтобы обеспечить корректность данных, передаваемых в приложении, DTO объекты проходят проверку с помощью валидаторов, реализованных на основе библиотеки FluentValidation. Это позволяет автоматически проверять свойства, такие как длина, наличие обязательных полей и формат данных, что упрощает управление ошибками и повышает надёжность. Пример реализации валидатора для DTO объекта входных данных от пользователя при авторизации представлен в листинге 3.2.

|  |
| --- |
| public class LoginDtoValidator : AbstractValidator<LoginDto> {  public LoginDtoValidator() {  RuleFor(x => x.Email).NotEmpty().  WithMessage("Email cannot be empty.");  RuleFor(x => x.Email).  EmailAddress(EmailValidationMode.AspNetCoreCompatible).  WithMessage("Email is not valid");  RuleFor(x => x.Password).NotEmpty().  WithMessage("Password cannot be empty.");  RuleFor(x => x.Password).Length(8, 60).  WithMessage("Password length must be > 8 and < 60");  }  } |

Листинг 3.2 – Код интерфейса сервиса авторизации

Для обработки ошибок в слое бизнес-логики созданы собственные исключения, которые помогают управлять возникающими проблемами и обрабатывать их в рамках бизнес-логики. Собственные исключения позволяют точно указать на причину ошибки и передать её в представление, а также облегчают диагностику, благодаря централизованному управлению обработкой ошибок.

В рамках слоя бизнес-логики реализован набор ключевых сервисов, которые поддерживают основные бизнес-операции приложения: регистрацию и авторизацию пользователей, управление профилями, комментариями и кеширование данных.

Регистрация и авторизация реализованы через контроллеры регистрации и авторизации. Сервисы регистрации и авторизации выполняют валидацию данных, проверяют существующие учетные записи и обеспечивают создание новых профилей. В процессе авторизации используется безопасная система обработки и хранения данных пользователей, а также внедрены механизмы шифрования и защиты информации, что гарантирует надежную аутентификацию. Кроме того, пользовательские данные, такие как профиль и статус, кешируются в Redis, что позволяет значительно ускорить доступ к информации, снижая нагрузку на базу данных.

Присоединение к игре и её проведение в реальном времени обеспечивается с помощью технологии SignalR, позволяющей организовать двустороннюю передачу данных в реальном времени. На уровне бизнес-логики используется специализированный сервис для организации и управления игровыми сессиями, что позволяет пользователям моментально подключаться к играм и участвовать в игровом процессе. Хабы SignalR, используемые на уровне представления, обрабатывают запросы в реальном времени и взаимодействуют с сервисом игр на уровне бизнес-логики, обеспечивая плавный и интерактивный игровой процесс для пользователей.

Управление профилем осуществляется с помощью сервиса профилей, отвечающего за выполнение основных операций по работе с информацией о пользователях. Сервис профилей позволяет обновлять данные, такие как имя, аватар, контактные сведения и другие настройки пользователя, обеспечивая при этом валидацию и сохранение изменений.

Управление комментариями обрабатывается через сервис комментариев, который поддерживает добавление, обновление и удаление пользовательских комментариев. Этот сервис отвечает за корректное и безопасное взаимодействие с хранилищем данных и обеспечивает валидацию содержимого, чтобы поддерживать качество и точность пользовательского контента.

3.1.3 Слой представления (API)

Слой представления реализует интерфейс взаимодействия пользователя с функциональностью приложения и разработан на основе ASP.NET Core Web API. Этот слой является общей точкой доступа к сервисам бизнес-логики и предоставляет API-интерфейсы для управления различными компонентами приложения, такими как аутентификация, профили пользователей, уведомления и игровые функции.

Каждое Web API приложение в проекте разработано для выполнения своей части бизнес-логики, что способствует модульности и структурированности архитектуры. Например, Web API для аутентификации и регистрации пользователей предоставляет методы для создания и управления учетными записями, Web API для работы с профилями пользователей управляет персональной информацией, а Web API для управления игрой включает контроллеры вопросов, игр и викторин. Такой подход позволяет делегировать обработку различных задач отдельным контроллерам, что облегчает их поддержку и тестирование, а также делает приложение более гибким и масштабируемым.

Контроллеры ориентированы на высокую эффективность взаимодействия с бизнес-логикой. Каждый контроллер выполняет только необходимые действия, делегируя основную обработку данных сервисам слоя бизнес-логики. Это достигается благодаря использованию механизма внедрения зависимостей (Dependency Injection, DI), что упрощает доступ к нужным сервисам. Таким образом, контроллеры взаимодействуют с бизнес-логикой без непосредственной работы с данными, обеспечивая строгое разделение ответственности. Пример реализации контроллера на примере сервиса авторизации представлен в листинге 3.3.

|  |
| --- |
| [ApiController] [Route("api/auth/[controller]")]  public class LoginController : ControllerBase {  private readonly ILoginService \_loginService;  public LoginController(ILoginService authService)  =>\_loginService = authService;  [HttpPost]  public async Task<ActionResult<TokenDto>> Login([FromBody] LoginDto loginDto, CancellationToken cancellationToken = default)  => Ok( await \_loginService.AuthenticateAsync(loginDto, cancellationToken));  } |

Листинг 3.3 – Реализация контроллера авторизации

Подобная тонкость в реализации контроллеров повышает производительность и делает интерфейс API более простым и понятным для клиента.

Таким образом, слой представления играет ключевую роль в обеспечении взаимодействия конечных пользователей с приложением и предоставляет доступ к функциональности, скрывая при этом детали внутренней реализации и управляя данными через контроллеры, которые обращаются к бизнес-логике через строго типизированные сервисы.

3.1.4 API Gateway

API Gateway – это web-приложение, реализованное на основе Ocelot, которое служит единой точкой доступа к микросервисам. Его основное назначение заключается в упрощении взаимодействия клиентов с несколькими сервисами через один конечный пункт, что улучшает производительность и снижает задержки.

Настройка API Gateway с Ocelot включает создание файла конфигурации ocelot.json, где определяются маршруты, по которым запросы перенаправляются на соответствующие микросервисы. В этом файле указываются параметры маршрутизации, такие как DownstreamPathTemplate и UpstreamPathTemplate. Для корректной работы необходимо зарегистрировать Ocelot в методе ConfigureServices в Startup.cs и вызвать UseOcelot в методе Configure.

Управление маршрутизацией запросов позволяет гибко распределять трафик между микросервисами, основываясь на HTTP методах и правилах обработки. Важно также обеспечить безопасность, используя механизмы аутентификации, такие как JWT, что защищает доступ к сервисам.

Таким образом, API Gateway на основе Ocelot упрощает архитектуру взаимодействия с микросервисами и обеспечивает высокий уровень безопасности и управления трафиком.

3.1.5 Дополнительные библиотеки

Также, в рамках курсового проекта, были реализованы две дополнительные библиотеки, которые облегчают разработку и повышают качество приложения: EmailSenderLibrary и GlobalExceptionHandlerLibrary.

GlobalExceptionHandlerLibrary реализует глобальный обработчик исключений, который используется во всех микросервиса. Эта библиотека представлена в виде расширения Middleware, что позволяет легко интегрировать её в существующие приложения. Глобальный обработчик обеспечивает централизованное управление ошибками, позволяя перехватывать и обрабатывать исключения на уровне приложения, а также возвращать понятные сообщения об ошибках клиентам. Это улучшает отладку и поддерживаемость приложения, а также обеспечивает более стабильную работу микросервисов.

EmailSenderLibrary представляет собой библиотеку для отправки электронных сообщений с использованием пакета MimeKit. Эта библиотека предоставляет удобный клиент, который упрощает процесс настройки и отправки email-уведомлений. Благодаря использованию MimeKit, библиотека поддерживает различные функции, такие как отправка писем с вложениями, форматирование сообщений в HTML и управление адресами получателей, что делает её мощным инструментом для работы с электронной почтой в приложении.

3.2 Разработка клиентской части приложения

Клиентская часть приложения предоставляет интерфейс для взаимодействия с серверным приложением, используя современные технологии, которые делают взаимодействие более интуитивным и гибким. В качестве основного фронтенд-фреймворка мы используем React с TypeScript, который обеспечивает строгую типизацию и высокую производительность, что особенно важно для крупного и интерактивного приложения.

Shadcn UI – это библиотека компонентов для React, ориентированная на гибкость и кастомизацию. В отличие от традиционных UI-библиотек, Shadcn UI поставляется не в виде готовых компонентов, а в виде шаблонов, которые можно адаптировать под конкретные нужды проекта. Каждый компонент представляет собой настроенную функциональную и визуальную единицу, которая легко интегрируется и адаптируется к общему стилю приложения. Этот подход позволяет нам избежать чрезмерной зависимости от заранее заданных стилей, обеспечивая высокую степень контроля над интерфейсом и возможность внедрения уникального стиля. Это делает интерфейс адаптивным и легко подстраивающимся под любые требования.

Для стилизации мы используем Tailwind CSS – утилитарный CSS-фреймворк, который предоставляет набор готовых классов для оформления пользовательских интерфейсов. Tailwind отличается от классических CSS-фреймворков тем, что использует утилитарные классы, позволяя описывать стиль компонента прямо в его коде. Этот подход делает код более структурированным и снижает количество дублирующегося CSS. Tailwind позволяет легко адаптировать дизайн, делая его отзывчивым и управляемым, а также улучшает производительность за счёт минимизации и оптимизации CSS.

React Router DOM используется для управления маршрутизацией. Он позволяет создавать одностраничные приложения, где навигация происходит плавно, без перезагрузки страницы. В нашем приложении React Router DOM обеспечивает интуитивный переход между основными страницами и функциональными модулями, что значительно улучшает пользовательский опыт.

Для создания анимаций в интерфейсе мы используем библиотеку AOS (Animate On Scroll). AOS добавляет анимационные эффекты, которые активируются, когда элементы становятся видимыми на экране при прокрутке страницы. Эти динамические анимации оживляют интерфейс, делая его более интерактивным и привлекательным для пользователя. Анимации позволяют акцентировать внимание на важных элементах, улучшая общее восприятие интерфейса.

Совокупное использование этих технологий и библиотек помогает нам создавать продуманный, современный интерфейс, который не только эстетически привлекателен, но и удобен в использовании, а также оптимизирован для различных устройств и экранов.

4 Тестирование web-приложения

В данном разделе описано тестирование приложения с использованием клиентской части.

4.1 Пункт

5 Руководство пользователя

Руководство пользователя является важным компонентом любого web-приложения, поскольку оно предоставляет пользователю необходимую информацию о функционале приложения, его возможностях и способах использования. Руководство пользователя помогает пользователям быстро освоиться с интерфейсом и функциями приложения, снижает уровень путаницы и упрощает процесс взаимодействия с приложением.

Далее в руководстве пользователя будут рассмотрены ключевые функциональные возможности web-приложения. Они включают в себя функционал логина, регистрации, создания и изменения постов, просмотр и персонализация страницы пользователя, просмотр новостей, просмотр страницы подписок и подписчиков.

5.1 Вход в аккаунт

текст

5.2 Регистрация пользователя

текст

5.3 Взаимодействие с профилем пользователя

Текст

Таким образом, в данной главе было разработано руководство по использованию, которое облегчает пользователю освоение функционала приложения.

Заключение

В ходе выполнения дипломного проекта было разработано web-приложение на тему «Программное средство сбора статистики и организации проведения интеллектуальных игр с элементами социальной сети». Проект успешно реализует поставленные задачи, предоставляя пользователям удобные инструменты для участия в интеллектуальных играх, сбора и анализа статистики, а также взаимодействия с другими пользователями в рамках игровой платформы.

В процессе разработки были реализованы ключевые функциональные модули, такие как:

* регистрация и авторизация пользователей;
* управление профилем пользователя и его настройками;
* создание и участие в интеллектуальных играх;
* сбор и отображение статистики по результатам игр;
* система комментариев и обсуждений;
* интеграция элементов социальной сети для взаимодействия между игроками.

Для обеспечения безопасности системы была реализована стратегия авторизации и аутентификации, а также внедрен механизм защиты данных с использованием SSL-сертификатов. Важным компонентом системы стала возможность масштабирования с применением микросервисной архитектуры, что позволяет системе обрабатывать большое количество пользователей и игр одновременно.

Важным этапом разработки стало тестирование программного продукта, которое позволило выявить и устранить ошибки, повысить производительность системы и убедиться, что все требования к проекту выполнены. Тестирование подтвердило, что система работает стабильно и предоставляет корректные данные в ходе взаимодействия с пользователями и сбора статистики.

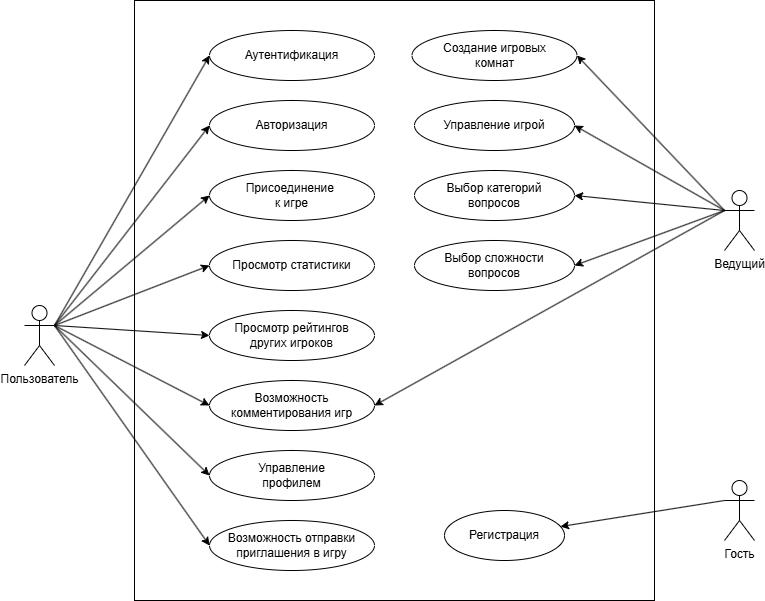
Использование современных технологий, таких как ASP.NET Core Web API, Redis, API Gateway и клиентская часть на React, позволило создать гибкую, надежную и удобную платформу. Web-приложение демонстрирует эффективное применение этих технологий и подходов, что обеспечивает возможность его дальнейшего расширения и добавления новых функциональных возможностей.

Таким образом, цель проекта была достигнута. Созданное web-приложение предоставляет пользователям функциональный и безопасный инструмент для организации интеллектуальных игр, обмена мнениями и анализа результатов. Это делает его ценным решением для всех участников интеллектуальных игр и открывает возможности для дальнейшего развития и улучшения системы.

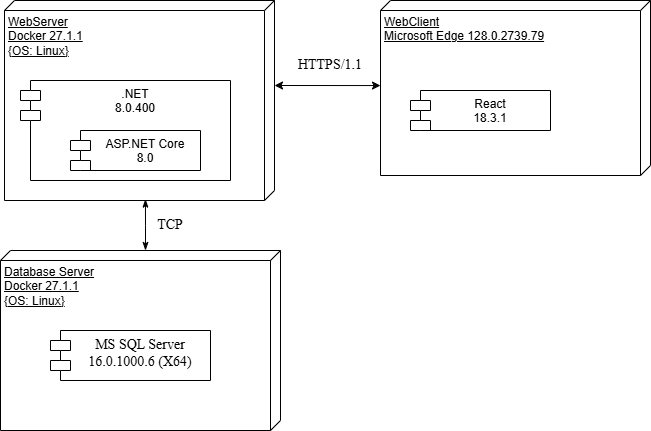
Список используемых источников

1. METANIT.COM Сайт о программировании [Электронный ресурс] / Режим доступа: https://metanit.com – Дата доступа: 15.03.2024.
2. Koa.js [Электронный ресурс] / Режим доступа: https://koajs.com. – Дата доступа: 19.03.2024.
3. Stackoverflow.com [Электронный ресурс] / Режим доступа: https://stackoverflow.com – Дата доступа: 23.03.2024.
4. React.js документация [Электронный ресурс] / Режим доступа: https://react.dev/learn. – Дата доступа: 19.04.2024.
5. Socket.io документация [Электронный ресурс] / Режим доступа: https://socket.io/docs/v4/. – Дата доступа: 27.04.2024.

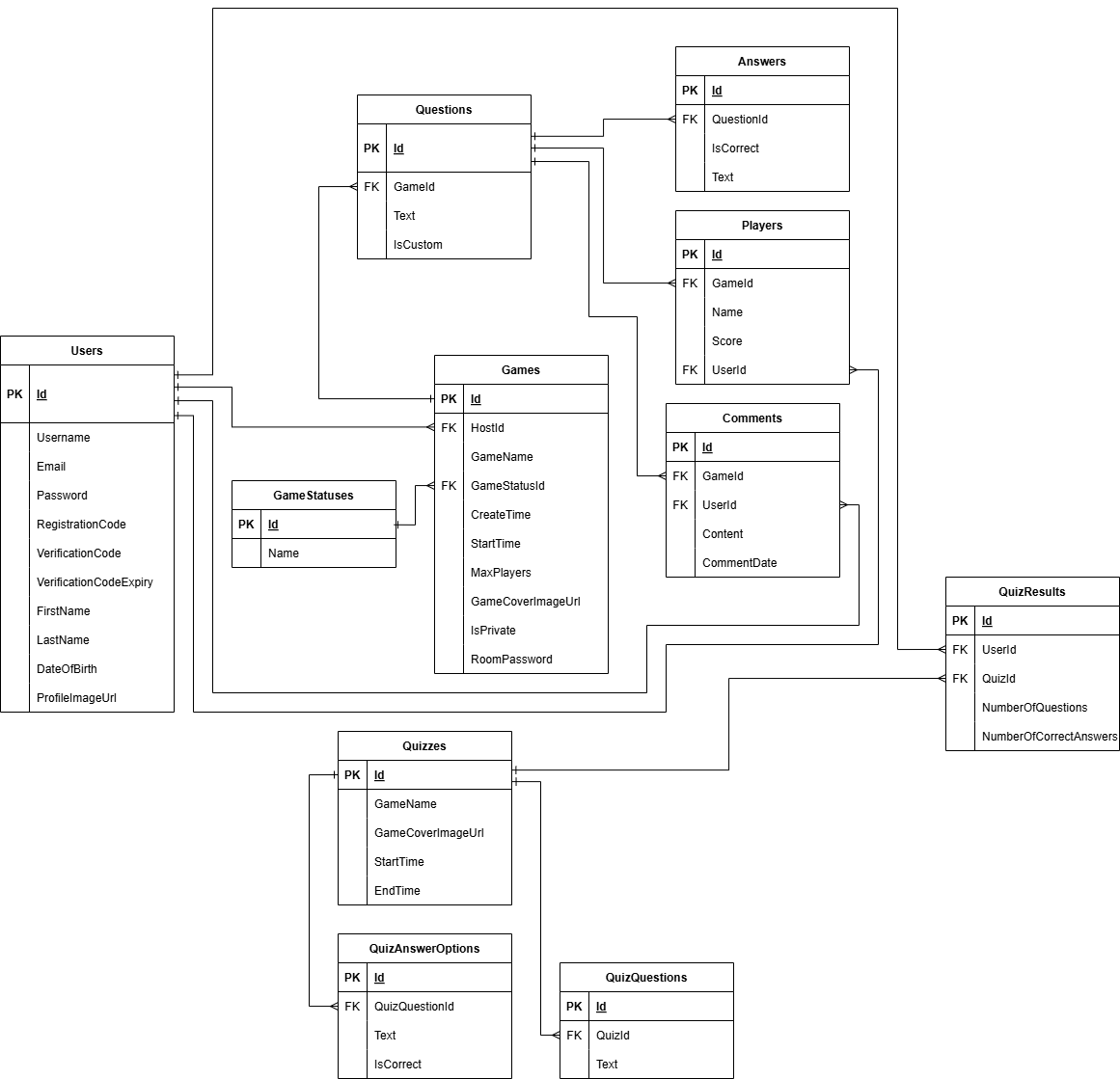
Приложение А



Приложение Б



Приложение В



Приложение Г

|  |
| --- |
| CREATE DATABASE BELLINI;  GO  CREATE TABLE [dbo].[Questions](  [Id] [int] IDENTITY(1,1) NOT NULL,  d] [int] NOT NULL,  [IsCustom] [bit] NOT NULL,  CONSTRAINT [PK\_Questions] PRIMARY KEY ([Id])  );  ALTER TABLE [dbo].[Questions]  ADD CONSTRAINT [FK\_Questions\_Games\_GameId] FOREIGN KEY([GameId])  REFERENCES [dbo].[Games]([Id]) ON DELETE CASCADE;  CREATE TABLE [dbo].[QuizAnswerOptions](  [Id] [int] IDENTITY(1,1) NOT NULL,  NOT NULL,  [IsCo  [QuizQuestionId] [int] NOT NULL,  CONSTRAINT [PK\_QuizAnswerOptions] PRIMARY KEY ([Id])  );  ALTER TABLE [dbo].[QuizAnswerOptions]  ADD CONSTRAINT [FK\_QuizAnswerOptions\_QuizQuestions\_QuizQuestionId] FOREIGN KEY([QuizQuestionId])  REFERENCES [dbo].[QuizQuestions]([Id]) ON DELETE CASCADE;  CREATE TABLE [dbo].[QuizComments](  [Id] [int] IDENTITY(1,1) NOT NULL,  [QuizId] [int] NOT NULL,  [UserId] [int] NOT NULL,  NOT NULL,  [Username] [nvarchar](max)ageUrl] [nvarchar](max) NOT NULL,  NOT NULL,  CONSTRAINT [PK\_QuizComments] PRIMARY KEY ([Id])  );ments]  ADD CONSTRAINT [FK\_QuizComments\_Quizzes\_QuizId] FOREIGN KEY([QuizId])  REFERENCES [dbo].[Quizzes]([Id]) ON DELETE CASCADE;  CREATE TABLE [dbo].[QuizQuestions](  [Id] [int] IDENTITY(1,1) NOT NULL,  NOT NULL,  [QuizId] [int] NOT NULL,  CONSTRAINT [PK\_QuizQuestions] PRIMARY KEY ([Id]).[QuizQuestions]  ADD CONSTRAINT [FK\_QuizQuestions\_Quizzes\_QuizId] FOREIGN KEY([QuizId])  REFERENCES [dbo].[Quizzes]([Id]) ON DELETE CASCADE;  CREATE TABLE [dbo].[QuizResults](  [Id] [int] IDENTITY(1,1) NOT NULL,  [NumberOfCorrectAnswers] [int] NOT NULL,  [NumberOfQuestions] [int] NOT NULL,  [UserId] [int] NOT NULL,  NOT NULL,  [QuizId] [int] NOT NULL,  CONSTRAINT [PK\_QuizResults] PRIMARY KEY ([Id])  );  ALTER TABLE [dbo].ONSTRAINT [FK\_QuizResults\_Quizzes\_QuizId] FOREIGN KEY([QuizId])  REFERENCES [dbo].[Quizzes]([Id]) ON DELETE CASCADE;  ALTER TABLE [dbo].[QuizResults]  ADD CONSTRAINT [FK\_QuizResults\_Users\_UserId] FOREIGN KEY([UserId])  REFERENCES [dbo].[Users]([Id]) ON DELETE CASCADE;  CREATE TABLE [dbo].[Quizzes](  [Id] [int] IDENTITY(1,1) NOT NULL,  NOT NULL,  [GameCoverImageUrl] [nvarchar](500) NOT NULL,  NOT NULL,  NOT NULL,  CONSTRAINT [PK\_Quizzes] PRIMARY KEY ([I.[Users](  [ NOT NULL,  [IsEmailVerif [IsActive] [bit] NOT NULL,  NULL,  NULL,  [ProfileImageUrl] [nvarchar](255) NULL,  CONSTRAINT [PK\_Users] PRIMARY KEYE [dbo].[AnswerIDENTITY(1,L,  [IsCorrect] [bit] NOT NULL,  [QuestionId] [int] NOT NULL,  CONSTRAIARY KEY ([I.[AnswerOptwerOptions\_Questions\_QuestionId] FOREIGN KEY([QuestionId])  REFERENCES [dbo].[Questions]([Id]) ON DELETE CASCADE;  CREATE TABLE [dbo [Id] [int] IDENTITY(1,1) NOT NULL,  [GameId] [int] NOT NULL,  [PlayerId] [int] NOT NULL,  [QuestionId] [int] NOT NULL,  [SelectedOptionId] [int] NOT NULL,  [IsCorrect] [bit] NOT NULL,  CONSTRAINT [PK\_CompletedAnswers] PRIMARY KEY ([Id])  );  ALTER TABLE [dbo].[CompletedAnswers]  ADD CONSTRAINT [FK\_CompletedAnswers\_AnswerOptions\_SelectedOptionId] FOREIGN KEY([SelectedOptionId])  REFERENCES [dbo].[AnswerOptions]([Id]) ON DELETE CASCADE;  ALTER TABLE [dbo].[CompletedAnswers]  ADD CONSTRAINT [FK\_CompletedAnswers\_Games\_GameId] FOREIGN KEY([GameId])  REFERENCES [dbo].[Games]([Id]);  ALTER TABLE [dbo].[CompletedAnswers]  ADD CONSTRAINT [FK\_CompletedAnswers\_Players\_PlayerId] FOREIGN KEY([PlayerId])  REFERENCES [dbo].[Players]([Id]);  ALTER TABLE [dbo].[CompletedAnswers]  ADD CONSTRAINT [FK\_CompletedAnswers\_Questions\_QuestionId] FOREIGN KEY([QuestionId])  REFERENCES [dbo].[Questions]([Id]);  CREATE TABLE [dbo].[Games](  [Id] [int] IDENTITY(1,1) NOT NULL,  NOT NULL,  [HostId] [int] NOT NULL,  NOT NULL,  NOT NULL,  NOT NULL,  [MaxPlayers] [int] NOT NULL,  [GameCoverImageUrl] [nvarchar](max) NOT NULL,  [IsPrivate] [bit] NOT NULL,  NOT NULL,  [GameStatusId] [int] NOT NULL,  CONSTRAINT [PK\_Games] PRIMARY KEY ([Id])  );  ALTER TABLE [dbo].[Games]  ADD CONSTRAINT [StatusId] FOREIGN KEY([GameStatusId])  REs]([Id]);  CREAes](  [Id] [ NULL,  CONSTRAINT [PK\_GameStatuses] PRIMARY KEY ([Id])  );  CREATE TABLE [dbo].[Notifications](  [Id] [int] IDENTITY(1,1) NOT NULL,  NOT NULL,  NOT NULL,  NOT NULL,  [IsRead] [bit] NOT NULL,  CONSTRAINT [PK\_Notifications] PRIMARY KEY ([Id])  );  ALTER TABLE [dbo].[Notifications]  ADD CONSTRAINT [FK\_Notifications\_Users\_UserId] FOREIGN KEY([UserId])  REFERENCES [dbo].[Users]([Id]) ON DELETE CASCADE;  ); GO |

Приложение Д

|  |
| --- |
|  |