Утверждаю
Руководитель предприятия

Чайчиц А.Н._____

(Ф.И.О.)

_____

(подпись, печать предприятия)

« ___» _____20 г.

Учреждение образования
«Белорусский государственный технологический университет»

Факультет информационных технологий
Кафедра программной инженерии
Специальность 1-40 01 01 Программное обеспечение информационных технологий

ОТЧЕТ

по производственной преддипломной практике

на предприятии ЗАСО «Белнефтестрах» с 25 марта 2024 г. по 19 апреля 2024 г._
(наименование предприятия, сроки практики)

Исполнитель
Студент 4 курса 6 группы _____ Хлыстов Г.Г.
(подпись, дата) (Ф.И.О.)

Руководитель практики
от предприятия
Начальник сектора тех. поддержки
отдела информационных
технологий и развития _____ Ярошевич А.В.
(должность, печать предприятия) (подпись, дата) (Ф.И.О.)

Руководитель практики
от университета
преп-стажер _____ Север. А.С.
(должность, уч. звание) (подпись, дата) (Ф.И.О.)

Отчет защищен с оценкой _____

Минск 2024

Содержание

Введение

Прохождение практики в период с 10 февраля 2025 по 21 марта 2025 происходило в компании ООО «Модсен».

Целью преддипломной практики является закрепление теоретических знаний и навыков, полученных в вузе, а также приобретение нового опыта, который пригодится в будущей профессиональной деятельности. Важным аспектом также является развитие практических навыков самостоятельной работы и умения применять их для решения конкретных задач.

Темой дипломного проекта выбрано «Веб-приложение: Интеллектуальные игры и викторины». Разработка приложения ведётся с использованием микросервисной архитектуры и включает серверную и клиентскую части. В качестве стека технологий используются *ASP.NET Core* [1] для серверной части и *React* с *Vite* [2, 3] для клиентской. Для хранения данных применяется *SQL Server*, а для кэширования – *Redis* [4, 5]. Взаимодействие компонентов обеспечивается с помощью *Docker*, а для *UI* используется библиотека *ShadCN UI* [6, 7].

1 Постановка цели и формулировка задач

Целью преддипломной практики являются закрепление и углубление полученных теоретических знаний; овладение необходимыми навыками и умениями.

В результате прохождения преддипломной практики студент должен закрепить полученные теоретические и практические знания в области разработки программного обеспечения.

Преддипломная практика обучающихся является составной частью учебного процесса и обеспечивает закрепление у них профессиональных навыков и умений, а также выработку умений и навыков выполнения практических задач.

Задачи практики:

– изучить стандарт БГТУ по дипломному проектированию;

– провести анализ существующих систем, программных решений, технологий программирования по теме дипломного проекта;

– провести анализ требований, разработать функциональные требования и архитектуру проектируемого приложения;

– выполнить реализацию приложения по теме дипломного проекта.

Темой дипломного проекта выбрано «Веб-приложение: Интеллектуальные игры и викторины».

Приложение ориентировано на широкую аудиторию, включая любителей интеллектуальных игр, активных пользователей социальных сетей, а также тех, кто заинтересован в саморазвитии и участии в командных или индивидуальных викторинах.

Разрабатываемое приложение представляет собой *web*-приложение, доступное через интернет-браузеры на компьютерах, планшетах и смартфонах. Оно обеспечит взаимодействие пользователей с сервером и сохранение данных в базе данных

2 Анализ и сравнительный обзор аналогов

В соответствии с заданием преддипломной практики следует разработать веб-приложение, которое должно реализовывать функционал в соответствии с листом задания. Для формулировки окончательных требований к проектируемому программному средству рассмотрим аналоги программных средств того же направления.

Для реализации проекта требуется анализ аналогичных решений, проработка данных и создание удобного интерфейса.

2.1 Интернет-ресурс «Quiz»

«Quiz» – это онлайн-платформа, предназначенная для создания и прохождения интерактивных викторин [8]. 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 2.1.

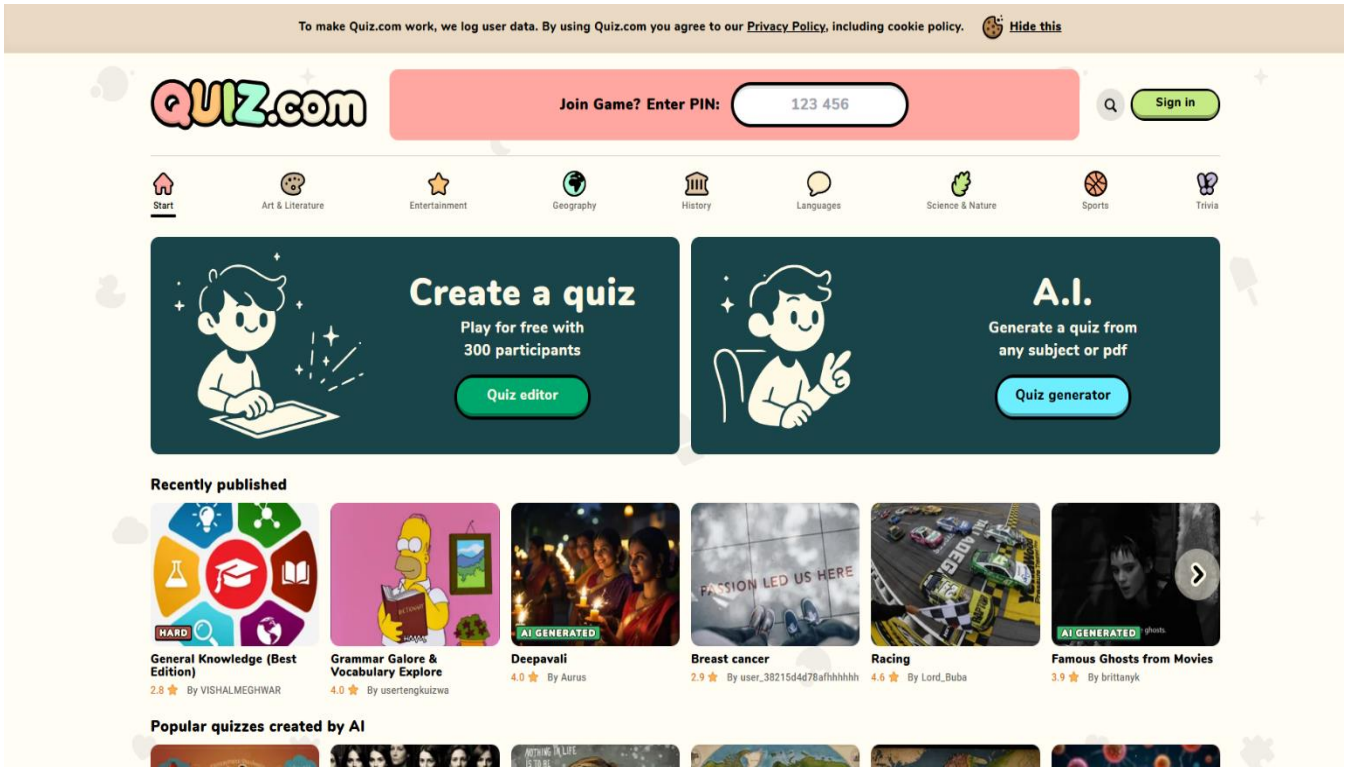

Рисунок 2.1 – Интерфейс интернет-ресурса «Quiz»

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.

2.2 Интернет-ресурс «Kahoot!»

«Kahoot!» – образовательная платформа для создания викторин, которая часто используется в школах и университетах для проведения интерактивных занятий [9]. Она ориентирована на группы участников, что делает её идеальной для обучения и совместных мероприятий.

Интерфейс интернет-ресурса «*Kahoot!*» представлен на рисунке 2.2.

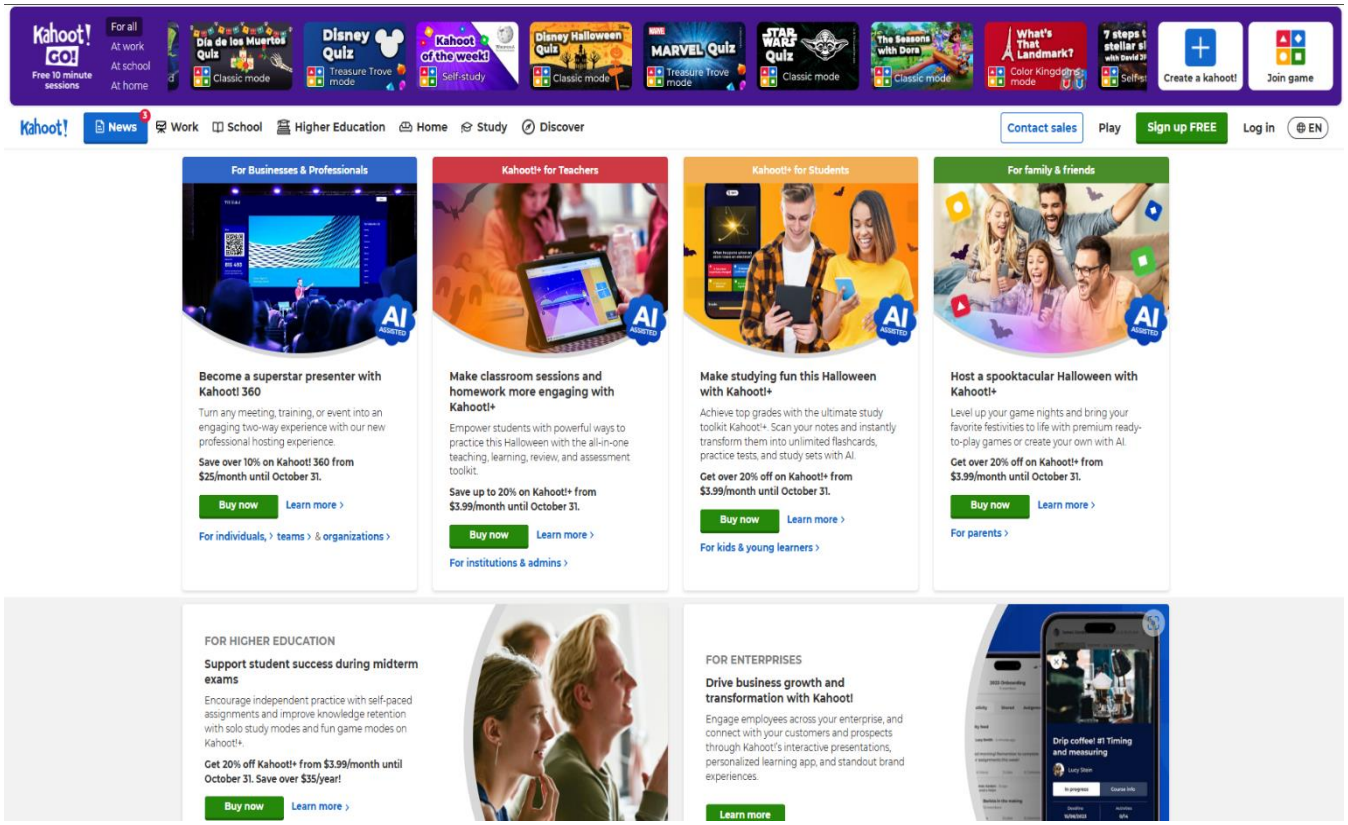

Рисунок 2.2 – Интерфейс интернет-ресурса «Kahoot!»

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.

2.3 Интернет-ресурс «Jackbox Games»

«Jackbox Games» – это набор многопользовательских игр, которые можно играть на различных устройствах с участием большого количества людей [10]. Игры отличаются оригинальными сценариями и креативными задачами, ориентированными на взаимодействие с друзьями.

Интерфейс интернет-ресурса «*Jackbox Games*» представлен на рисунке 2.3.

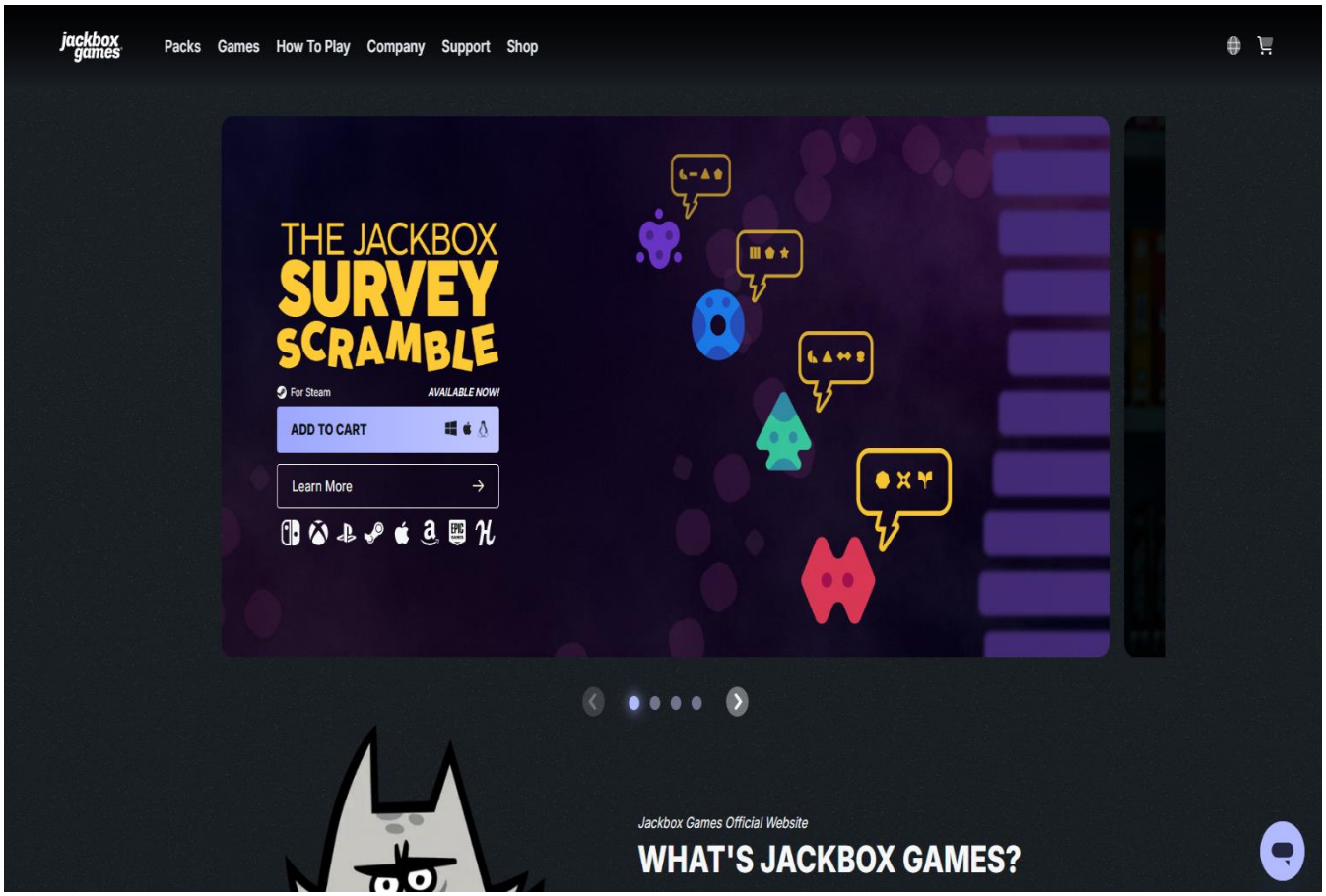

Рисунок 2.3 – Интерфейс интернет-ресурса «Jackbox Games»

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: данная 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.

2.4 Выводы по разделу

1. Основной задачей разработки веб-приложения является создание платформы с поддержкой ролей «Гость», «Пользователь», «Ведущий» и «Администратор», а также реализация функционала для взаимодействия с викторинами и игровыми комнатами, управления профилями и данными пользователей.

2. В ходе анализа существующих решений, таких как «*Quiz*», «*Kahoot*!» и «*Jackbox Games*», были выявлены их сильные и слабые стороны. «*Quiz*» предоставляет простую платформу для создания викторин. «*Kahoot!*» ориентирован на интерактивность и образовательное использование. «*Jackbox Games*» фокусируется на развлекательных многопользовательских играх.

Разрабатываемое приложение объединяет сильные стороны аналогов, расширяя функционал за счет интеграции элементов взаимодействия между пользователями и ведения интеллектуальных игр.

3 Проектирование web-приложения

В данной главе описывается принципы проектирования веб-приложения, структурная схема которого представлена на рисунке 3.1.

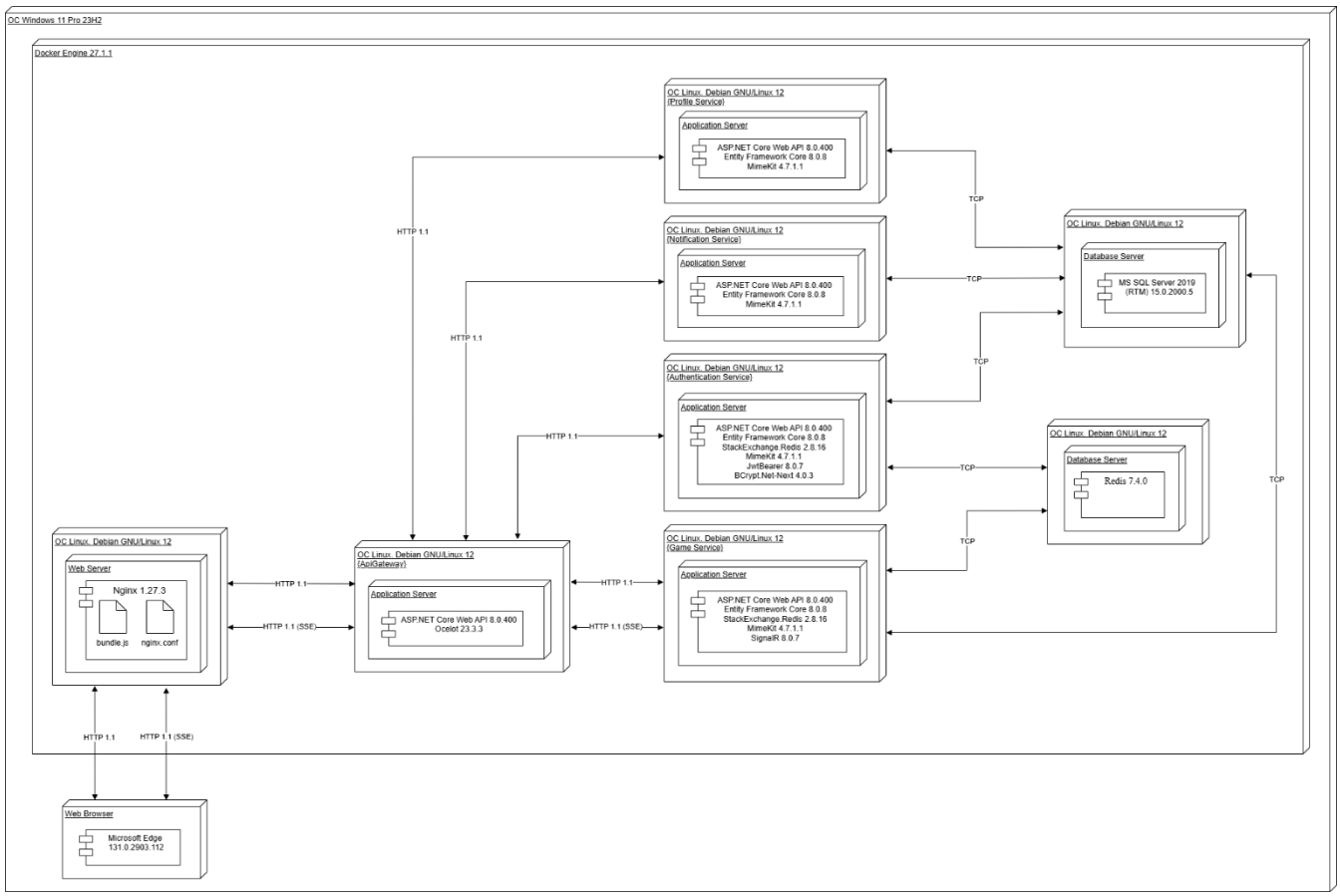

Рисунок 3.1 – Структурная схема web-приложения

Серверная часть построена на *ASP.NET Core 8.0* и включает несколько сервисов:

– *Profile Service* отвечает за логику взаимодействия с пользователями.
– *Game Service* обрабатывает игровую функциональность.
– *Notification Service* передаёт уведомления пользователям.
– *Authentication Service* управляет аутентификацией и авторизацией.
– *Admin Service* позволяет администратору управлять всеми сущностями.

Эти сервисы взаимодействуют между собой и с клиентскими компонентами по *HTTP 1.1*. Для отправки событий в реальном времени используется протокол *Server-Sent Events* (*SSE*) [11].

Хранение данных обеспечивается *MS SQL Server*, развернутым на той же операционной системе. Для передачи данных с сервисами используется *TCP*-соединение, что гарантирует надёжный и упорядоченный обмен информацией. Кэширование реализовано с помощью *Redis 7.4.0*, который ускоряет доступ к данным и уменьшает нагрузку на основную базу [12].

Клиентская часть разработана на *ReactJS 18.3.1* с использованием *Vite 5.4.0* для сборки и *Node.js 20.17.0* для обработки сборочного процесса [13]. Динамическое обновление данных и взаимодействие с сервером происходит через *HTTP* 1.1 и *SSE*.

Маршрутизация запросов осуществляется с помощью *Nginx 1.27.3* [14]. Он обслуживает статические ресурсы фронтенда: файлы index.html, bundle.js и другие. Файл *bundle.js* будет находиться внутри контейнера *Nginx* в директории */usr/share/nginx/html*.

Клиенты системы – это веб-приложения, запускаемые в браузерах (например, *Microsoft Edge*). Данные передаются с использованием *HTTP* 1.1 для *REST*-запросов и *SSE* для односторонней связи. Двусторонняя коммуникация реализована с помощью *SignalR 8.0.7*, где *SSE* используется для отправки событий от сервера клиенту [15].

Таким образом, система использует надёжную распределённую архитектуру, включающую серверные сервисы на *ASP.NET Core* 8.0, *React Vite* для фронтенда, *MS SQL Server* для хранения данных и *Redis* для кэширования. *Nginx* и использование протоколов *HTTP* 1.1, *SSE* и *TCP* обеспечивают эффективное взаимодействие компонентов и передачу данных в реальном времени.

3.1 Диаграмма вариантов использования

В системе предусмотрены следующие роли: «Гость», «Пользователь», «Ведущий» и «Администратор». Каждая из этих ролей имеет определённый набор прав и ограничений, в зависимости от набора доступных функций. Для лучшего понимания взаимодействия ролей и функций, была построена диаграмма вариантов использования, которая отражает основные действия пользователей в приложении. Она отображает ключевые сценарии использования, охватывая все важные процессы и действия.

В таблице 3.1 представлено описание ролей.

Таблица 3.1 – Описание ролей

Роль	Описание роли
Гость	Пользователь, не авторизованный в системе, имеющий доступ к ограниченному количеству функций. 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 игровых сессий
Ведущий	Пользователь с расширенными правами для проведения игр. Может создавать вопросы и варианты ответов, запускать и завершать игру, переходить между вопросами, а также удалять комментариями к завершенным играм. Ведущий несет ответственность за управление игровым процессом

Продолжение таблицы 3.1

Роль	Описание роли
Администратор	Авторизованный пользователь, имеющий расширенные права. Может управлять и изменять пользователей, викторины и игры. Владеет максимально допустимым набором прав

Проектирование функциональности веб-приложения позволяет создать удобный и интуитивно понятный интерфейс, удовлетворяющий потребности различных групп пользователей. Диаграмма вариантов использования и таблицы ролей помогают визуализировать и структурировать ключевые процессы взаимодействия с системой.

3.2 Проектирование базы данных

Структура базы данных состоит из четырнадцати таблиц, каждая из которых отвечает за хранение и управление определенными данными. Логическая схема базы данных представлена на рисунке 3.2.

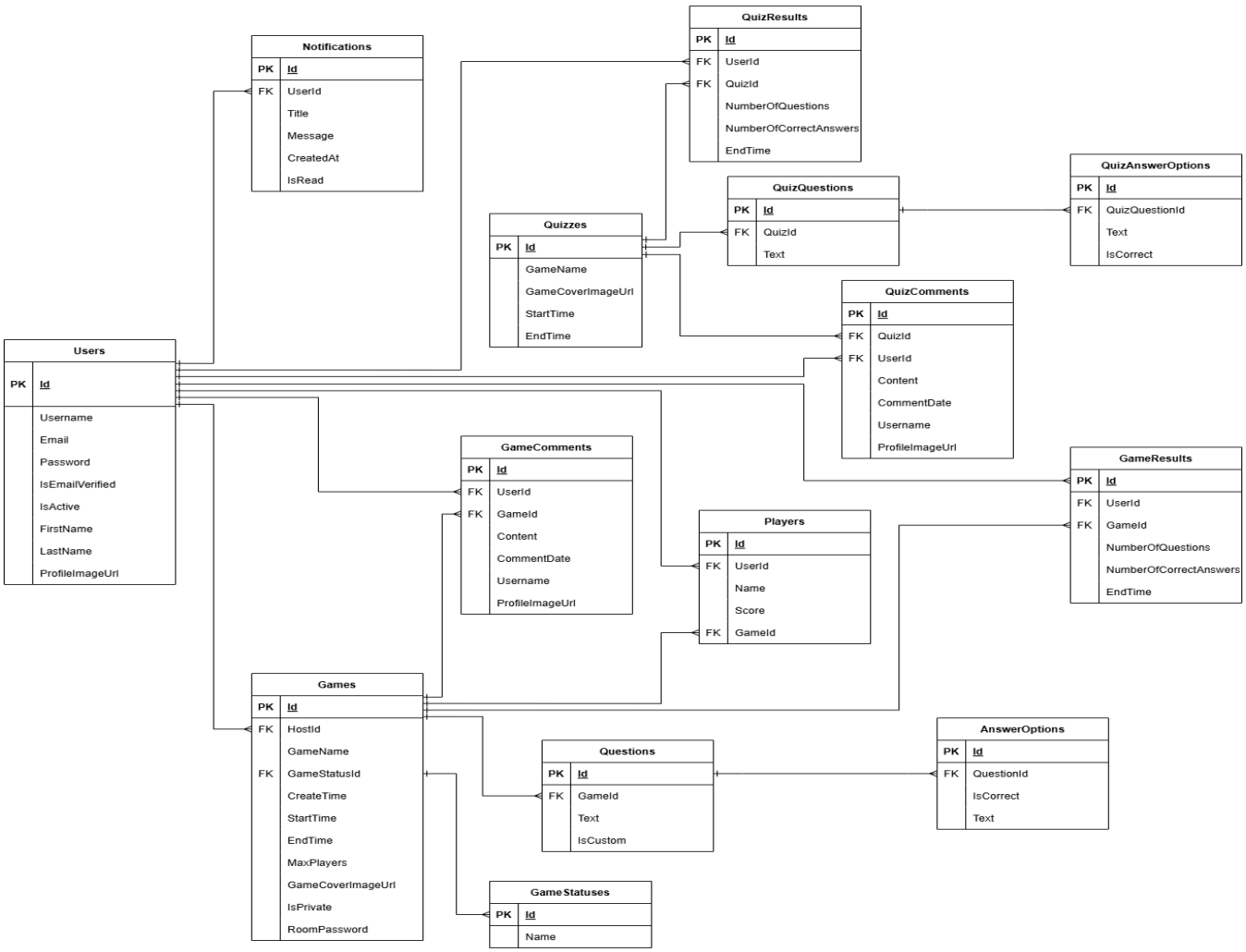

Рисунок 3.2 – Логическая схема базы данных

В таблице 3.2 предоставлена краткая информация о каждой таблице, включая наименования и описание все сущностей базы данных. 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 3.2 – Таблицы базы данных

Название таблицы	Описание использования
AnswerOptions	Хранит варианты ответов для вопросов, включая текст ответа, его корректность и связь с вопросами
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 и пароль
GameStatuses	Хранит возможные статусы игр
Notifications	Содержит уведомления для пользователей, включая заголовок, текст сообщения, дату создания и статус
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
QuizAnswrOptions	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 правильные ответы, время завершения и пользователя
Quizzes	Хранит информацию о викторинах, включая название, обложку, время начала/окончания
Users	Хранит информацию о пользователях

AnswerOptions – хранит варианты ответов для вопросов, включая текст ответа, его корректность и связь с вопросами. Её структура представлена в таблице 3.3.

Таблица 3.3 – Структура таблицы *AnswerOptions*

Название	Тип данных	Назначение
Id	*INT*	Уникальный идентификатор варианта ответа
Text	*NVARCHAR(200)*	Текст варианта ответа
IsCorrect	*BIT*	Признак корректности ответа
QuestionId	*INT*	Идентификатор связанного вопроса, к которому относится данный ответ

GameComments – содержит комментарии пользователей к играм, включая текст комментария, имя пользователя, аватар и дату. Структура представлена в таблице 3.4.

Таблица 3.4 – Структура таблицы *GameComments*

Название	Тип данных	Назначение
Id	*INT*	Уникальный идентификатор комментария
GameId	*INT*	Идентификатор игры, к которой относится комментарий
Username	*NVARCHAR(MAX)*	Имя пользователя, оставившего комментарий
UserId	*INT*	Идентификатор пользователя
Content	*NVARCHAR(500)*	Текст комментария
CommentDate	*DATETIME2(7)*	Дата и время создания комментария
ProfileImageUrl	*NVARCHAR(MAX)*	Ссылка на аватар пользователя

GameResults – хранит результаты игр для пользователей, включая количество правильных ответов и общее количество вопросов. Структура представлена в таблице 3.5.

Таблица 3.5 – Структура таблицы *GameResults*

Название	Тип данных	Назначение
Id	*INT*	Уникальный идентификатор результата игры
NumberOfCorrectAnswers	*INT*	Количество правильных ответов
NumberOfQuestions	*INT*	Общее количество вопросов
UserId	*INT*	Идентификатор пользователя, к которому относится результат
GameId	*INT*	Идентификатор игры, связанной с результатом

Games – содержит информацию об играх, таких как название, ведущий, время начала/окончания, статус, приватность и пароль. Её структура представлена в таблице 3.6.

Таблица 3.6 – Структура таблицы *Games*

Название	Тип данных	Назначение
Id	*INT*	Уникальный идентификатор игры.
GameName	*NVARCHAR(100)*	Название игры
HostId	*INT*	Идентификатор ведущего игры
CreateTime	*DATETIME2(7)*	Время создания игры
StartTime	*DATETIME2(7)*	Время начала игры
EndTime	*DATETIME2(7)*	Время окончания игры
MaxPlayers	*INT*	Максимальное количество участников

Продолжение таблицы 3.6

Название	Тип данных	Назначение
GameCoverImageUrl	*NVARCHAR(MAX)*	Ссылка на изображение обложки игры
IsPrivate	*BIT*	Признак приватности игры
RoomPassword	*NVARCHAR(50)*	Пароль для доступа к приватной игре
GameStatusId	*INT*	Идентификатор текущего статуса игры

GameStatuses – содержит возможные статусы игр. Структура представлена в таблице 3.7.

Таблица 3.7 – Структура таблицы *GameStatuses*

Название	Тип данных	Назначение
Id	*INT*	Уникальный идентификатор статуса
Name	*NVARCHAR(100)*	Название статуса игры

Notifications – содержит уведомления для пользователей, включая заголовок, текст сообщения, дату создания и статус прочтения. Структура представлена в таблице 3.8.

Таблица 3.8 – Структура таблицы *Notifications*

Название	Тип данных	Назначение
Id	*INT*	Уникальный идентификатор уведомления
UserId	*INT*	Идентификатор пользователя, получившего уведомление
Title	*NVARCHAR(100)*	Заголовок уведомления
Message	*NVARCHAR(1000)*	Текст сообщения уведомления
CreatedAt	*DATETIME2(7)*	Дата и время создания уведомления
IsRead	*BIT*	Признак прочтения уведомления

Players – Хранит данные о игроках игр, включая имя, счёт, связь с игрой и пользователем. Структура таблицы представлена в таблице 3.9.

Таблица 3.9 – Структура таблицы *Players*

Название	Тип данных	Назначение
Id	*INT*	Уникальный идентификатор игрока
GameId	*INT*	Идентификатор игры, к которой относится игрок
Name	*NVARCHAR(100)*	Имя игрока
Score	*INT*	Текущий счёт игрока
UserId	*INT*	Идентификатор пользователя, связанного с игроком

Questions – содержит информацию о вопросах, связанных с играми. Структура представлена в таблице 3.10.

Таблица 3.10 – Структура таблицы *Questions*

Название	Тип данных	Назначение
Id	*INT*	Уникальный идентификатор вопроса
Text	*NVARCHAR(500)*	Текст вопроса
GameId	*INT*	Идентификатор игры, к которой относится вопрос
IsCustom	*BIT*	Признак кастомного вопроса

QuizAnswerOptions – хранит варианты ответов для викторин, включая текст ответа и информацию о корректности. Структура представлена в таблице 3.11.

Таблица 3.11 – Структура таблицы *QuizAnswerOptions*

Название	Тип данных	Назначение
Id	*INT*	Уникальный идентификатор варианта ответа для викторины
Text	*NVARCHAR(200)*	Текст варианта ответа
IsCorrect	*BIT*	Признак корректности ответа
QuizQuestionId	*INT*	Идентификатор вопроса викторины, связанного с ответом

QuizComments – Содержит комментарии пользователей к викторинам, включая текст комментария, имя пользователя и дату. Структура представлена в таблице 3.12.

Таблица 3.12 – Структура таблицы *QuizComments*

Название	Тип данных	Назначение
Id	*INT*	Уникальный идентификатор комментария к викторине
QuizId	*INT*	Идентификатор викторины, к которой относится комментарий
UserId	*INT*	Идентификатор пользователя, оставившего комментарий
Content	*NVARCHAR (500)*	Текст комментария
Username	*NVARCHAR (max)*	Пользователь, оставивший комментарий
ProfileImageUrl	*NVARCHAR (max)*	Ссылка на аватар пользователя
CommentDate	*DATETIME2(7)*	Дата и время создания комментария

QuizQuestions – содержит вопросы викторины. Структура представлена в таблице 3.13.

Таблица 3.13 – Структура таблицы *QuizQuestions*

Название	Тип данных	Назначение
Id	*INT*	Уникальный идентификатор вопроса викторины
Text	*NVARCHAR(500)*	Текст вопроса викторины
QuizId	*INT*	Идентификатор викторины, к которой относится вопрос

QuizResults – таблица для хранения результатов викторины. Структура представлена в таблице 3.14.

Таблица 3.14 – Структура таблицы *QuizResults*

Название	Тип данных	Назначение
Id	*INT*	Уникальный идентификатор результата викторины
NumberOfCorrectAnswers	*INT*	Количество правильных ответов
NumberOfQuestions	*INT*	Общее количество вопросов в викторине
EndTime	*DATETIME2(7)*	Дата и время завершения викторины
UserId	*INT*	Идентификатор пользователя, связанного с результатом
QuizId	*INT*	Идентификатор викторины, связанной с результатом

Quizzes – хранит информацию о викторинах, связанных с играми. Структура представлена в таблице 3.15.

Таблица 3.15 – Структура таблицы *Quizzes*

Название	Тип данных	Назначение
Id	*INT*	Уникальный идентификатор викторины
GameName	*NVARCHAR(200)*	Название викторины
GameCoverImageUrl	*NVARCHAR(500)*	Ссылка на изображение обложки викторины
StartTime	*DATETIME2(7)*	Дата и время начала викторины
EndTime	*DATETIME2(7)*	Дата и время окончания викторины

Users – хранит информацию о пользователях, зарегистрированных в системе. Структура представлена в таблице 3.16.

Таблица 3.16 – Структура таблицы *Users*

Название	Тип данных	Назначение
Id	*INT*	Уникальный идентификатор пользователя
Email	*NVARCHAR(MAX)*	Адрес электронной почты пользователя
Username	*NVARCHAR(MAX)*	Имя пользователя
Password	*NVARCHAR(MAX)*	Пароль пользователя
IsEmailVerified	*BIT*	Признак подтверждения электронной почты
IsActive	*BIT*	Признак активности пользователя
FirstName	*NVARCHAR(MAX)*	Имя пользователя
LastName	*NVARCHAR(MAX)*	Фамилия пользователя
ProfileImageUrl	*NVARCHAR(MAX)*	Ссылка на аватар пользователя

Таким образом, представленные таблицы и связи формируют основу базы данных, обеспечивая возможность гибкого хранения и обработки всех данных, необходимых для функционирования системы.

3.3 Проектирование 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. Для реализации приложения выбрана платформа *.Net* с использованием языка *C#.*

3.3.1 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. Дабы отличать классы моделей от остальных классов, они были унаследованы от абстрактного класса *BaseModel*. Для хранения данных используется СУБД *MS SQL Server*.

Для реализации паттерна «Репозиторий» в слое доступа к данным выделены интерфейсы и соответствующие классы репозиториев [16]. Репозитории предоставляют стандартный набор методов *CRUD* (создание, чтение, обновление, удаление), а также могут включать специфичные для приложения методы, упрощающие взаимодействие с данными.

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

3.3.2 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 (классах), 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.

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.

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.

3.3.3 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.

3.3.4 API Gateway

API Gateway – это *web*-приложение, реализованное на основе *Ocelot 23.3.3*, которое служит единой точкой доступа к микросервисам [17, 18]. Его основное назначение заключается в упрощении взаимодействия клиентов с несколькими сервисами через один конечный пункт, что улучшает производительность и снижает задержки.

API Gateway также является паттерном проектирования, который позволяет централизовать обработку входящих запросов и распределять их между различными микросервисами.

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 *Program.cs* и вызвать *UseOcelot* в методе у объекта *app*.

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.

3.4 Проектирование 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 [19].

Shadcn UI 2.1.8 – 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.

Для стилизации мы используем *Tailwind CSS 3.4.9* – 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 6.26.0 используется для управления маршрутизацией. Он позволяет создавать одностраничные приложения, где навигация происходит плавно, без перезагрузки страницы. В приложении *React Router DOM* обеспечивает интуитивный переход между основными страницами и функциональными модулями, что значительно улучшает пользовательский опыт. Настройки маршрутизации указываются в файле *App.tsx*.

Для создания анимаций в интерфейсе мы используем библиотеку *AOS 2.3.4* (*Animate On Scroll*). *AOS* добавляет анимационные эффекты, которые активируются, когда элементы становятся видимыми на экране при прокрутке страницы.

3.5 Выводы по разделу

1. Для серверной части приложения будут реализованы слои доступа к данным, бизнес-логики и представления (*API*). Применена архитектура с использованием *API Gateway*, обеспечивающего маршрутизацию запросов. Использованы дополнительные библиотеки для повышения производительности и обеспечения безопасности.
2. В клиентской части приложения будут разработаны пользовательские интерфейсы с учетом удобства использования и адаптации к различным устройствам. Интеграция с серверной частью выполнена через *API*.
3. Настроен *Nginx*, выполняющий функции обратного прокси-сервера и маршрутизации запросов. Обеспечена поддержка SSL-соединений и статических файлов.
4. Реализовано 20 функций, включая регистрацию, авторизацию, просмотр и редактирование профилей пользователей, создание и прохождение викторин, взаимодействие с игровыми комнатами, управление комментариями и рейтинговыми данными.
5. В проекте будут применены современные технологии, обеспечивающие масштабируемость, безопасность и удобство использования системы. Для серверной части был использован паттерн *DI* совместно с *DTO* и *Api Gateway*.

Заключение

В ходе производственной преддипломной практики, проходившей в период с 10 февраля 2025 по 21 марта 2025, было разработано веб-приложение для проведения интеллектуальных игр и викторин.

Во время производственной практики была изучена структура предприятия ООО «Модсен». Были изучены основные нормативно-правовые акты, углублены и закреплены знания, полученные при изучении специальных дисциплин.

В ходе производственной практики были закреплены, расширены и углублены полученные теоретические знания в области веб-разработки, приобретены практические навыки самостоятельной работы, выработаны умения применять их при решении конкретных вопросов и задач.

Было разработано веб-приложение по теме дипломного проекта: Веб-приложение «Интеллектуальные игры и викторины». В соответствии с полученным результатом работы веб-приложения можно сделать вывод, что разработанное программное средство работает корректно.

В результате проделанной работы было создано работоспособное веб-приложение, соответствующее требованиям дипломного проекта.

Список используемых источников

1 ASP.NET Core Web API [Электронный ресурс] / Режим доступа: https://learn.microsoft.com/en-us/aspnet/core/web-api/?view=aspnetcore-9.0 – Дата доступа: 13.02.2025

2 React [Электронный ресурс] / Режим доступа: https://react.dev – Дата доступа: 13.02.2025

3 Vite [Электронный ресурс] / Режим доступа: https://vite.dev – Дата доступа: 13.02.2024

4 MS SQL Server [Электронный ресурс] / Режим доступа: https://www.microsoft.com/en-us/sql-server – Дата доступа: 13.02.2025

5 Redis [Электронный ресурс] / Режим доступа: https://redis.io – Дата доступа: 13.02.2025

6 Docker [Электронный ресурс]. – Режим доступа: https://docs.docker.com – Дата доступа: 13.02.2025

7 Shadcn UI [Электронный ресурс] / Режим доступа: https://ui.shadcn.com – Дата доступа: 13.02.2025

8 Quiz [Электронный ресурс] / Режим доступа: https://quiz.com – Дата доступа: 18.02.2025

9 Kahoot! [Электронный ресурс] / Режим доступа: https://kahoot.com – Дата доступа: 18.02.2025

10 Jackbox Games [Электронный ресурс] / Режим доступа: https://www.jackboxgames.com/ – Дата доступа: 18.02.2025

11 HTTP 1.1 [Электронный ресурс] / Режим доступа: https://www.w3.org/Protocols/ – Дата доступа: 02.03.2025

12 TCP [Электронный ресурс] / Режим доступа: https://www.rfc-editor.org/rfc/rfc793 – Дата доступа: 03.03.2025

13 Node.js [Электронный ресурс] / Режим доступа: https://nodejs.org/en – Дата доступа: 04.03.2025

14 Nginx [Электронный ресурс] / Режим доступа: https://nginx.org – Дата доступа: 04.03.2025

15 SignalR [Электронный ресурс] / Режим доступа: https://learn.microsoft.com/en-us/aspnet/core/signalr/introduction?view=aspnetcore-9.0 – Дата доступа: 04.03.2025

16 Репозиторий паттерн [Электронный ресурс] / Режим доступа: https://learn.microsoft.com/ru-ru/aspnet/mvc/overview/older-versions/getting-started-with-ef-5-using-mvc-4/implementing-the-repository-and-unit-of-work-patterns-in-an-asp-net-mvc-application – Дата доступа: 09.03.2025

17 ApiGateway паттерн [Электронный ресурс] / Режим доступа: https://dotnetfullstackdev.medium.com/api-gateway-in-net-microservice-architecture-411cdf52c22d – Дата доступа: 11.03.2025

18 Ocelot [Электронный ресурс] / Режим доступа: https://ocelot.readthedocs.io/en/latest/ – Дата доступа: 12.03.2025

19 TypeScript [Электронный ресурс] / Режим доступа: https://www.typescriptlang.org – Дата доступа: 16.03.2025