

Содержание

Реферат	6
Abstract	7
Введение.....	8
1 Постановка задачи и аналитический обзор аналогичных решений	9
1.1 Постановка задачи	9
1.2 Аналитический обзор аналогичных решений.....	9
1.2.1 Приложение «Спортивный вызов».....	9
1.2.2 Приложение «Life +»	11
1.2.3 Приложение SPOBI	12
1.3 Функциональные требования	12
1.4 Выводы по разделу	13
2 Проектирование веб-приложения.....	14
2.1 Функциональность веб-приложения.....	14
2.2 Логическая схема базы данных	15
2.3 Архитектура веб-приложения	20
2.4 Проектирование алгоритма предложения вызова другу	21
2.5 Проектирование алгоритма добавления активности	22
2.6 Выводы по разделу	23
3 Реализация веб-приложения	23
3.1 Обоснование выбора программной платформы	24
3.2 Система управления базами данных PostgreSQL	24
3.3 EntityFramework	24
3.4 Программные библиотеки.....	28
3.5 Структура серверной части.....	29
3.6 Реализация функций для роли «Гость»	30
3.6.1 Регистрация	30
3.6.2 Авторизация	31
3.7 Реализация функций для роли «Пользователь»	32
3.7.1 Ввод данных о себе.....	32
3.7.2 Получение уведомлений	33
3.7.3 Получение вызова.....	33
3.7.4 Ввод ежедневной активности	34
3.7.5 Возможность бросить вызов другу	35
3.7.6 Добавление пользователей в друзья	36
3.8 Реализация функций для роли «Администратор»	37
3.8.1 Добавление и удаление активностей	37
3.8.2 Ведение статистики	37
3.9 Структура клиентской части.....	38

					ДП 00.00.ПЗ		
		ФИО	Подпись	Дата			
Разраб.	Викторович И.С.				Оглавление	Лит.	Лист
Пров.	Гончар Е.А.						1
							3
Н. контр.	Гончар Е.А.					БГТУ 1-40 01 01, 2025	
Утв.	Смелов В.В.						

3.10 Выводы по разделу	39
4 Тестирование веб-приложения	40
4.1 Функциональное тестирование	40
4.2 Выводы по разделу	43
5 Руководство пользователя.....	44
5.1 Руководство пользователя для роли «Гость».....	44
5.1.1 Авторизация	44
5.1.2 Регистрация	44
5.2 Руководство пользователя для роли «Пользователь»	45
5.2.1 Ввод данных о себе	45
5.2.2 Получение уведомлений	46
5.2.3 Получение вызова.....	46
5.2.4 Ввод ежедневной активности.....	47
5.2.5 Возможность бросить вызов другу.....	48
5.2.6 Добавление пользователей в друзья.....	49
5.2.7 Добавление своей цели	50
5.2.8 Добавление шагов к выполнению цели	51
5.3 Руководство пользователя для роли «Администратор»	52
5.3.1 Добавление и удаление активностей.....	52
5.3.2 Поиск пользователей.....	53
5.3.3 Ведение статистики.....	54
5.4 Выводы по разделу	55
6 Техничко-экономическое обоснование проекта.....	56
6.1 Общая характеристика разрабатываемого программного средства	56
6.2 Исходные данные для проведения расчетов и маркетинговый анализ..	56
6.3 Обоснование цены программного средства.....	58
6.3.1 Расчет затрат рабочего времени на разработку программного средства	58
6.3.2 Расчет основной заработной платы	58
6.3.3 Расчет дополнительной заработной платы	59
6.3.4 Расчет отчислений в Фонд социальной защиты населения и по обязательному страхованию.....	60
6.3.5 Расчет суммы прочих прямых затрат	60
6.3.6 Расчет суммы накладных расходов	61
6.3.7 Сумма расходов на разработку программного средства.....	61
6.3.8 Расходы на сопровождение и адаптацию	61
6.3.9 Расчет полной себестоимости	62
6.3.10 Определение цены.....	62
6.4 Выводы по разделу	63
Заключение	66
Список используемых источников.....	67
Диаграмма вариантов использования ДП 01.00.ГЧ.....	69
Логическая схема базы данных ДП 02.00.ГЧ.....	70
Диаграмма развертывания ДП 03.00.ГЧ	71

Блок-схема алгоритма предложения вызова другу ДП 04.00.ГЧ.....	72
Диаграмма последовательности алгоритма добавления активности ДП 05.00.ГЧ....	73
Скриншот работы приложения ДП 06.00.ГЧ	74
Приложение А	75
Приложение Б	77
Приложение В.....	79
Приложение Г	81

Реферат

Пояснительная записка дипломного проекта включает 82 страницы, 30 таблиц, 31 иллюстрацию, 20 литературных источников и 4 приложения, а также содержит 18 листингов.

ВЕБ-ПРИЛОЖЕНИЕ, СПОРТИВНЫЙ ВЫЗОВ, REACT, ASP.NET CORE, POSTGRESQL, ENTITY FRAMEWORK, API

Целью дипломного проекта является создание веб-приложения «Спортивный вызов». Веб-приложение предназначено для мотивации пользователей к ведению активного образа жизни и регулярным тренировкам. Основная задача приложения – предоставить интуитивно понятный интерфейс для персонализированного подбора спортивных вызовов, отслеживания прогресса и взаимодействия с другими пользователями, а также оснастить администраторов инструментами для управления контентом и аналитикой. Веб-приложение ориентировано на людей, заинтересованных в спорте, стремящихся к самосовершенствованию.

Пояснительная записка проекта состоит из введения, содержания, пяти разделов, заключения и списка используемых источников. В первом разделе определена цель проекта и сформулированы технические требования к разработке, выполнен обзор существующих решений в области спортивных приложений. Во втором разделе представлен процесс проектирования, включая диаграмму вариантов использования, архитектуру приложения, схему базы данных и алгоритмы некоторых функций веб-приложения. Третий раздел посвящен детальному описанию этапов разработки программного обеспечения, включая выбор технологий и инструментов. Четвертый раздел содержит анализ результатов тестирования функциональности веб-приложения. Пятый раздел представляет инструкцию для пользователей, описывающую основные возможности веб-приложения. Шестой раздел включает расчет экономической эффективности и анализ затрат на реализацию проекта. В заключении подведены итоги работы, определены достигнутые результаты веб-приложения.

Графический и иллюстративный материал включает 31 иллюстрацию, таких как схемы архитектуры, блок-схемы алгоритмов и визуализации интерфейса, а также 30 таблиц с данными из базы данных, тестах и экономических расчетах.

					ДП 00.00.ПЗ		
		ФИО	Подпись	Дата			
Разраб.		Викторович И.С.			Реферат	Лит.	Лист
Пров.		Гончар Е.А.					1
							1
Н. контр.		Гончар Е.А.				БГТУ 1-40 01 01, 2025	
Утв.		Смелов В.В.					

Abstract

The explanatory note for the diploma project comprises 82 pages, includes 30 tables, 31 illustrations, 20 literary sources, and 4 appendices, as well as contains 18 listings.

WEB APPLICATION, SPORTS CHALLENGE, REACT, ASP.NET CORE, POSTGRESQL, ENTITY FRAMEWORK, API

The aim of the diploma project is to create the web application "Sports Challenge." The web application is designed to motivate users to lead an active lifestyle and engage in regular workouts. The primary objective of the application is to provide an intuitive interface for personalized selection of sports challenges, progress tracking, and interaction with other users, while equipping administrators with tools for managing content and analytics. The web application targets individuals interested in sports and striving for self-improvement.

The explanatory note for the project consists of an introduction, table of contents, five sections, a conclusion, and a list of used sources. The first section defines the project's goal and outlines technical requirements for development, including a review of existing solutions in the field of sports applications. The second section covers the design process, including a use case diagram, the application's architecture, database schema, and algorithms for some functions of the web application. The third section is dedicated to a detailed description of the software development stages, including the selection of technologies and tools. The fourth section contains an analysis of the testing results for the web application's functionality. The fifth section provides a user manual, describing the main features of the web application. The sixth section includes an economic efficiency calculation and an analysis of project implementation costs. The conclusion summarizes the work, identifies the achieved results of the web application.

The graphic and illustrative material includes 31 illustrations, such as architecture diagrams, algorithm flowcharts, and interface visualizations, as well as 30 tables with data from the database, test results, and economic calculations.

					ДП 00.00.ПЗ		
		ФИО	Подпись	Дата			
Разраб.	Викторович И.С.				Введение		
Пров.	Гончар Е.А.						
Н. контр.	Гончар Е.А.						
Утв.	Смелов В.В.						
					Лит.	Лист	Листов
						1	1
					БГТУ 1-40 01 01, 2025		

Введение

Веб-приложение «Спортивный вызов» представляет собой приложение для поощрения физической активности и мотивирования пользователей на достижение спортивных целей.

Спортивный вызов – это задача или цель, связанная с физической активностью, которую пользователи принимают и выполняют в установленный срок, соревнуясь с другими участниками или стремясь превзойти собственные результаты.

Веб-приложение – это клиент-серверное приложение, в котором клиент взаимодействует с сервером по протоколу HTTP.

Целью проекта является создание веб-приложения «Спортивный вызов». Веб-приложение должно поддерживать несколько ролей: «Гость», «Пользователь», «Администратор».

Для достижения цели необходимо решить следующие задачи:

- постановка задач и обзор аналогичных решений;
- проектирование веб-приложения;
- реализация веб-приложения;
- тестирование веб-приложения;
- технико-экономическое обоснование проекта.

Целевая аудитория веб-приложения включает широкий круг пользователей, заинтересованных в улучшении своего здоровья, которые желают следить за своим прогрессом.

Серверная часть приложения реализована на платформе ASP.NET 8.0 [1], а клиентская часть – с помощью библиотеки React v18 [2]. В качестве системы управления базами данных выбрана PostgreSQL 17 [3].

1 Постановка задачи и аналитический обзор аналогичных решений

1.1 Постановка задачи

Веб-приложение «Спортивный вызов» создается для мотивации людей к регулярной физической активности и достижению спортивных целей. Основная функциональность заключается в сервисе генерации вызовов(заданий) на основе данных пользователя, с которым он взаимодействует. Дополнительно пользователь может ставить свои задачи(цели), записывать свою активность, взаимодействовать с другими пользователями.

1.2 Аналитический обзор аналогичных решений

1.2.1 Приложение «Спортивный вызов»

Приложение «Спортивный вызов» [4] представляет собой мобильное приложение для участия в спортивных вызовах. Главная страница представлена на рисунке 1.1.

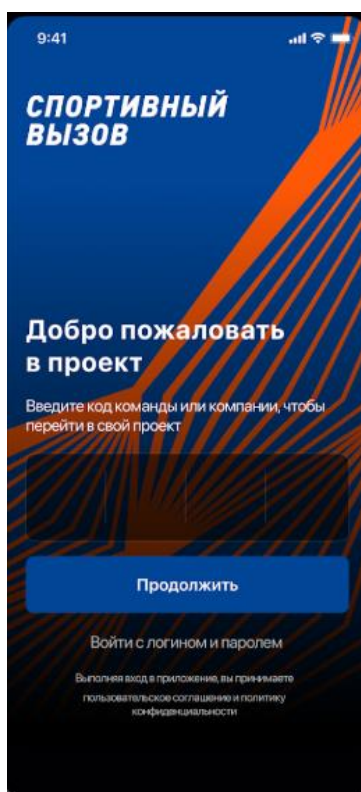


Рисунок 1.1 – Страница приложения «Спортивный вызов»

					ДП 01.00.ПЗ						
		ФИО	Подпись	Дата							
Разраб.		Викторович И.С.			1 Постановка задачи и аналитический обзор аналогичных решений			Лит.	Лист	Листов	
Пров.		Гончар Е.А.								1	5
								БГТУ 1-40 01 01, 2025			
Н. контр.		Гончар Е.А.									
Утв.		Смелов В.В.									

При входе сразу предлагается авторизоваться, чтобы иметь возможность использовать функции приложения. При успешной авторизации пользователю открывается доступ к возможностям приложения, таким как участие в вызовах, просмотр ленты активностей и другое. На рисунке 1.2а и 1.2б представлены страницы с основным функционалом приложения.

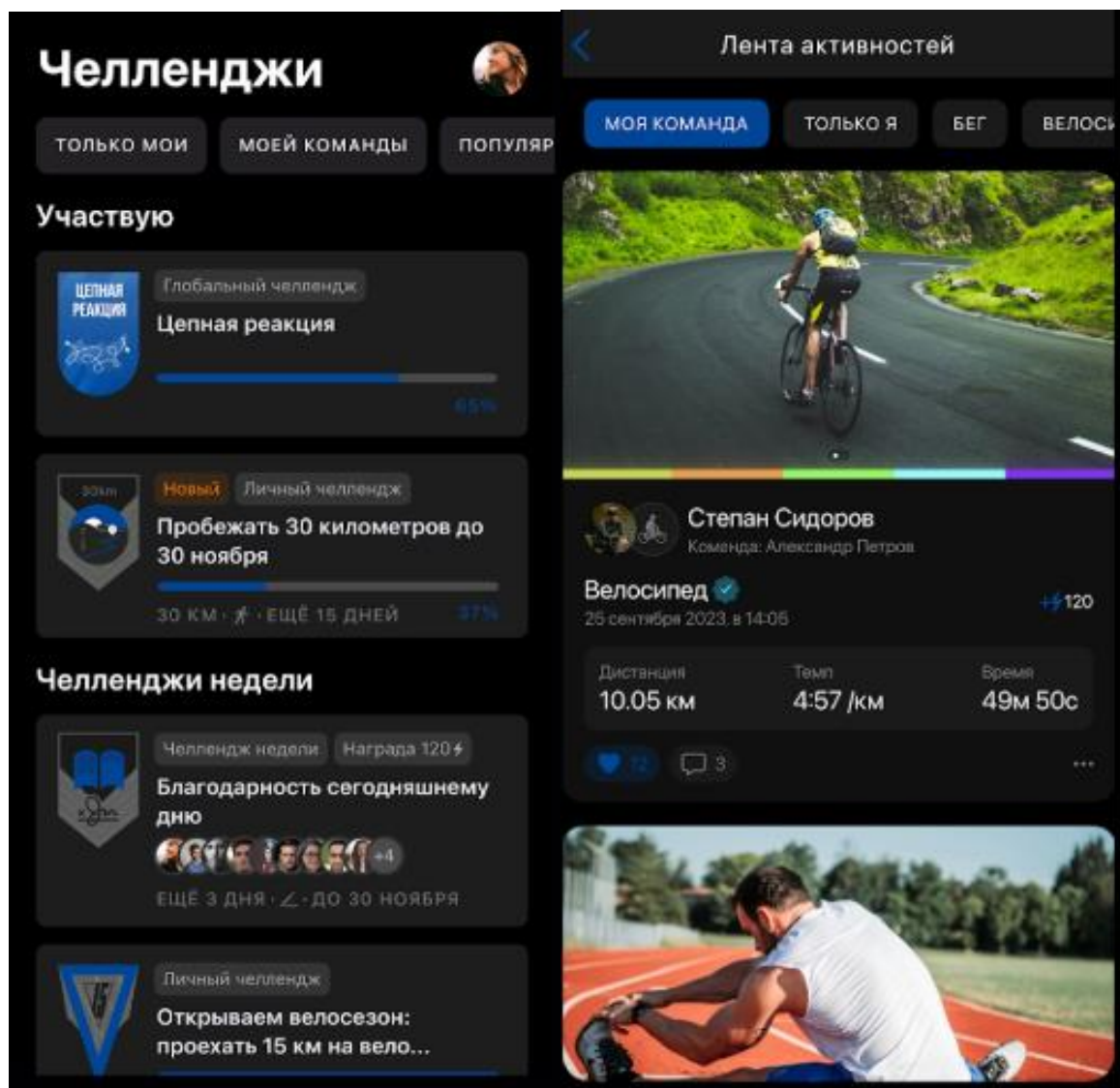


Рисунок 1.2а – Страница вызовов, 1.2б – Лента активностей

Достоинства:

- удобная система просмотра вызовов;
- простая и удобная навигация, позволяющая пользователям легко находить необходимые функции;
- возможность делиться достижениями, приглашать друзей, создавать группы и соревноваться с другими пользователями;

Недостатки:

- необходимость наличия современного смартфона или планшета, а также стабильного интернет-соединения для полноценного использования приложения.

1.2.2 Приложение «Life +»

Приложение для спорта «Life +» [5] – это многофункциональная платформа, которая обеспечивает возможность участия в спортивных вызовах и позволяет отслеживать свою активность. Благодаря интуитивно понятному интерфейсу пользователи могут легко создавать и присоединяться к различным вызовам, соревноваться с друзьями или другими спортсменами, а также получать мотивацию через систему наград и достижений. Страница со спортивными вызовами представлена на рисунке 1.3.

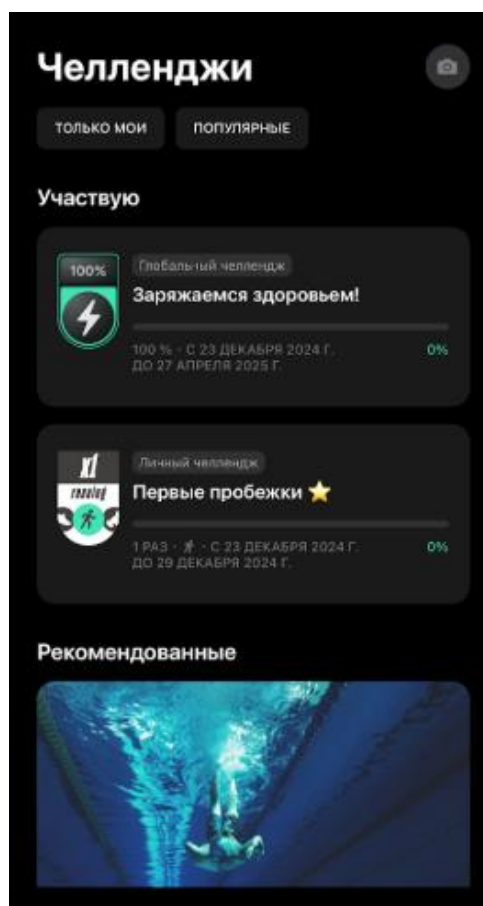


Рисунок 1.3 – Страница спортивных вызовов

Достоинства:

- возможность участвовать в соревнованиях, отслеживать результаты и прогресс;
- внедрение игровых элементов, таких как баллы, награды, уровни и таблицы лидеров, что повышает мотивацию и вовлеченность пользователей;
- возможность делиться достижениями.

Недостатки:

- возможные задержки, сбои или медленная работа приложения на менее мощных устройствах;
- необходимость регулярных обновлений для исправления ошибок и добавления новых функций, что может создавать неудобства для пользователей.

1.2.3 Приложение SPOBI

SPOBI [6] – это мобильное приложение, разработанное специально для спортсменов и любителей активного образа жизни, доступное на платформах Android и iOS.

Приложение представляет собой современную социальную сеть, созданную для удобного общения, обмена опытом и поддержания мотивации среди пользователей, независимо от их уровня подготовки – от начинающих энтузиастов до профессиональных спортсменов. Страница приложения представлена на рисунке 1.4.



Рисунок 1.4 – Страница статистики

Достоинства:

- возможность загрузки фото в приложение;
- возможность оставлять отзывы.

Недостатки:

- старомодный дизайн приложения.

1.3 Функциональные требования

Обзор вышеперечисленных аналогов позволяет проанализировать все преимущества и недостатки аналогичных решений, что позволяет сформулировать список требований, предъявляемых к программному средству, разрабатываемому в данном дипломном проекте.

Функционирование программного средства предусматривает поддержку нескольких ролей пользователей, каждая из которых обладает определенным набором прав и функциональных возможностей. Веб-приложение должно поддерживать роли: «Администратор», «Пользователь» и «Гость».

Функциональные возможности роли «Гость»:

- регистрация;
- авторизация.

Функциональные возможности роли «Пользователь»:

- ввод данных о себе;
- получение уведомлений;
- получение вызова;
- ввод ежедневной активности;
- возможность бросить вызов другу;
- добавление пользователей в друзья;
- добавление своей цели;
- добавление шагов к выполнению цели.

Функциональные возможности роли «Администратор»:

- добавление активностей;
- удаление активностей;
- поиск пользователей;
- ведение статистики.

Описанные выше требования обеспечат разработку функционального, безопасного и удобного в использовании веб-приложения.

1.4 Выводы по разделу

Веб-приложение «Спортивный вызов» успешно решает задачу мотивации пользователей к регулярной физической активности и достижению спортивных целей. Реализация сервиса генерации вызовов на основе персональных данных пользователя, а также предоставление возможностей для постановки собственных задач, записи активности и взаимодействия с другими пользователями создаёт основу для полноценного спортивного опыта.

Аналитический обзор аналогичных решений позволил детально изучить подобные веб-приложение. Выявленные достоинства, такие как удобная навигация, система просмотра вызовов и социальные функции, подчёркивают потенциал подобных решений. Однако недостатки, включая зависимость от современных устройств и стабильного интернета, указывают на необходимость устранения подобных ограничений в разработке. Эти выводы служат ориентиром для повышения конкурентоспособности нашего проекта.

Сформулированный список требований, включающий поддержку ролей «Администратор», «Пользователь» и «Гость» с их специфическими функциями обеспечивает основу для разработки безопасного, удобного и функционального продукта. Эти требования станут ключевым ориентиром на этапе проектирования и реализации.

2 Проектирование веб-приложения

2.1 Функциональность веб-приложения

Основные функциональные возможности веб-приложения для роли «Пользователь» представлены в диаграмме вариантов использования (рисунок 2.1).

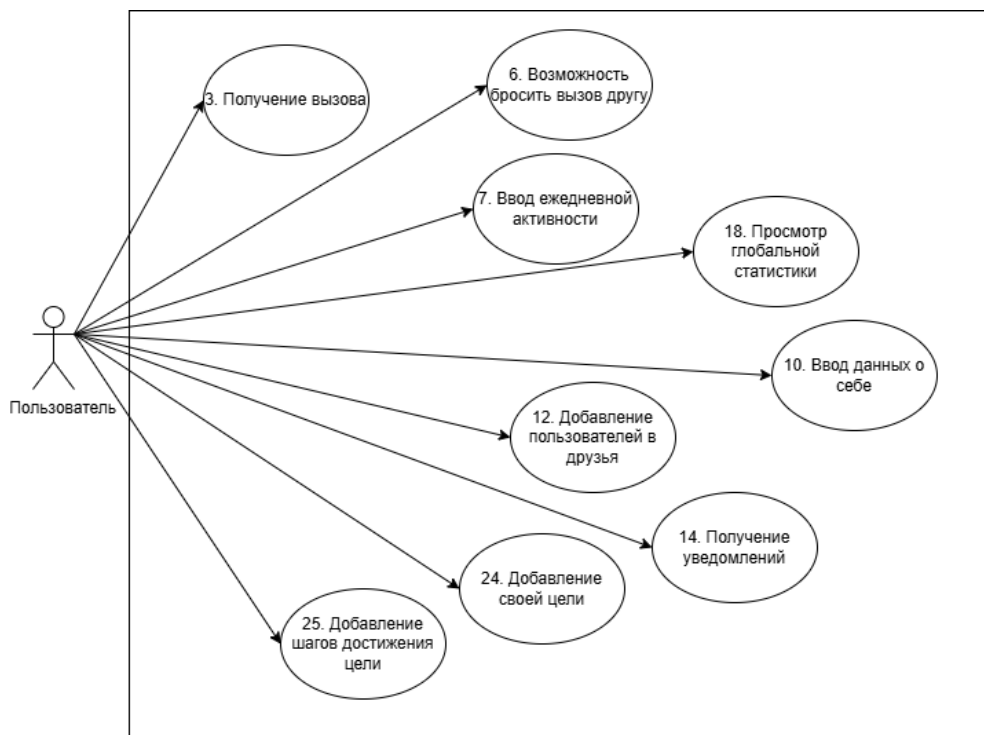


Рисунок 2.1 – Диаграмма вариантов использования веб-приложения для роли «Пользователь»

Полная версия диаграммы вариантов использования веб-приложения представлена в ДП 01.00.ГЧ. Описание ролей пользователей веб-приложения представлены в таблице 2.1.

Таблица 2.1 – Описание ролей пользователей веб-приложения

Роль	Описание
Гость	Пользователь, не прошедший регистрацию и авторизацию, не имеющий доступ к функциям приложения.
Пользователь	Пользователь, не прошедший регистрацию и аутентификацию, имеющий возможность получать вызовы, добавлять информацию о себе, добавлять друзей и заполнять ежедневную статистику.
Администратор	Уполномоченный пользователь, который может создавать виды активности, просматривать статистику и списки пользователей.

					ДП 01.00.ПЗ						
		ФИО	Подпись	Дата							
Разраб.		Викторович И.С.			2 Проектирование веб-приложения			Лит.	Лист	Листов	
Пров.		Гончар Е.А.								1	10
								БГТУ 1-40 01 01, 2025			
Н. контр.		Гончар Е.А.									
Утв.		Смелов В.В.									

Функциональные возможности пользователя, действующего в роли «Гость» представлены в таблице 2.2.

Таблица 2.2 – Функциональные возможности роли «Гость»

№	Вариант использования	Пояснение
1	Регистрация	Возможность создания новой учетной записи в системе.
2	Авторизация	Вход пользователя в систему с помощью данных учетной записи, таких как логин и пароль, получение прав доступа.

Функциональные возможности пользователя, действующего в роли «Пользователь» представлены в таблице 2.3.

Таблица 2.3 – Функциональные возможности роли «Пользователь»

№	Вариант использования	Пояснение
1	Ввод данных о себе	Добавление информации о предпочитаемом виде активности, рост, вес.
2	Ввод ежедневной активности	Ввод количества шагов и времени общей/другой активности, выполненных за день.
3	Получение вызова	Возможность получить ежедневное/еженедельное/ежемесячное задание.
4	Добавление пользователей в друзья	Возможность найти и добавить выбранного пользователя в друзья для последующей возможности бросать вызовы друзьям.
5	Возможность бросить вызов другу	Возможность отправить свой вызов для выполнения друзьям.
6	Получение уведомлений	Получение уведомлений о заявке в друзья, предложении о вызове
7	Добавление своей цели	Возможность добавить цель, которую пользователь желает достигнуть
8	Добавление шагов к выполнению цели	Возможность добавить действия, необходимые для достижения цели

Функциональные возможности пользователя, действующего в роли «Администратор» представлены в таблице 2.4.

Таблица 2.4 – Функциональные возможности роли «Администратор»

№	Вариант использования	Пояснение
1	Добавление и удаление активностями	Возможность создавать новый вид активности в системе или удалять.
2	Поиск пользователями	Поиск и просмотр статистики пользователей
3	Ведение статистики	Ведение глобальной статистики активности пользователей и просмотр статистики конкретного пользователя.

2.2 Логическая схема базы данных

Разработана диаграмма логической схемы реляционной базы данных, реализуемой в СУБД PostgreSQL. Логическая схема базы данных представлена на рисунке 2.2 и в ДП 02.00.ГЧ.

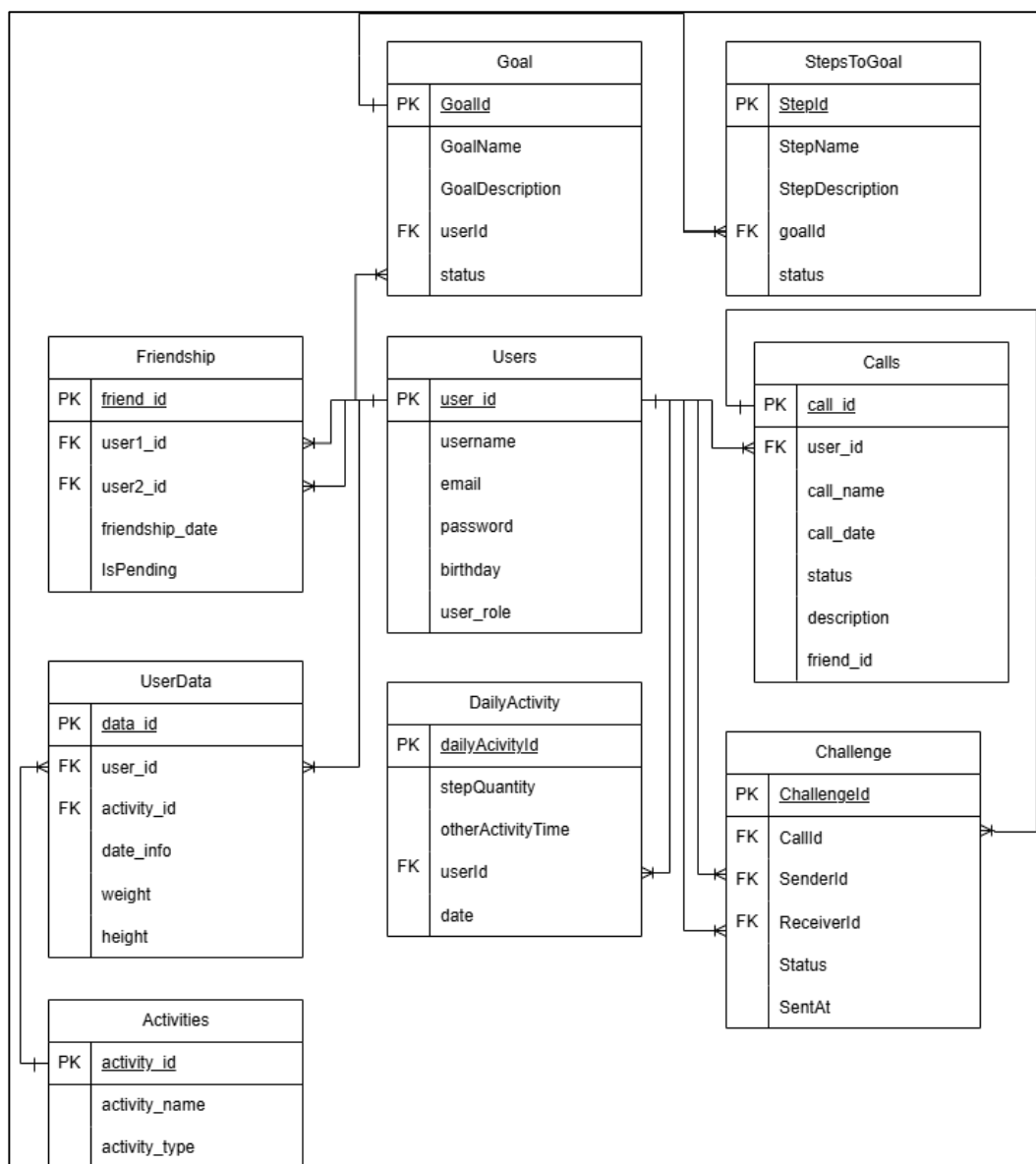


Рисунок 2.2 – Логическая схема базы данных

Спроектирована база данных из 9 таблиц: Users, Calls, Challenges, UserData, Activities, DailyActivity, Friendship, Goal и StepsToGoal. Названия и назначения таблиц базы данных приведены в таблице 2.5.

Таблица 2.5 – Таблицы базы данных

Название таблицы	Назначение таблицы
1	2
Users	Хранение информации о пользователях
Calls	Хранение информации о сгенерированных вызовах
Challenges	Хранение информации о вызовах, полученных от других пользователей
UserData	Хранение данных о пользователе
Activities	Хранение видов активности
DailyActivity	Хранение записей о ежедневной активности

Продолжение таблицы 2.5

1	2
Friendship	Хранение данных о дружбе
Goal	Хранение целей пользователя
StepsToGoal	Хранение шагов достижения цели

Таблица Users содержит информацию о пользователях веб-приложения. Структура данной таблицы приведена в таблице 2.6.

Таблица 2.6 – Структура таблицы Users

Название поля	Тип данных	Описание
User_id	INT	Идентификатор пользователя
username	STIRNG	Имя пользователя
password	STIRNG	Хешированный пароль пользователя
birthday	STIRNG	Дата рождения пользователя
user_role	STIRNG	Роль пользователя

Таблица Calls содержит информацию о вызовах, сгенерированных в веб-приложении. Структура данной коллекции приведена в таблице 2.7.

Таблица 2.7 – Структура таблицы Calls

Название поля	Тип данных	Описание
call_id	INT	Идентификатор вызова
call_name	STIRNG	Название вызова
call_date	STIRNG	Дата создания вызова
status	STIRNG	Статус вызова
description	STIRNG	Описание вызова
user_id	INT	Идентификатор пользователя

Таблица Challenges содержит информацию о вызовах, посланных между пользователями веб-приложения. Структура данной таблицы приведена в таблице 2.8.

Таблица 2.8 – Структура таблицы Challenges

Название поля	Тип данных	Описание
ChallengeId	INT	Идентификатор отправленного вызова
SenderId	INT	Идентификатор отправителя
RecieverId	INT	Идентификатор получателя
CallId	INT	Идентификатор сгенерированного вызова
Status	STIRNG	Статус вызова
SentAt	DateTime	Время отправления вызова

Таблица UserData содержит личные данные о пользователях веб-приложения. Эта таблица играет ключевую роль в поддержке таких аспектов, как отслеживание физической активности, настройка индивидуальных спортивных вызовов и формирование статистики. Структура данной таблицы приведена в таблице 2.9

Таблица 2.9 – Структура таблицы UserData

Название поля	Тип данных	Описание
data_id	INT	Идентификатор записи данных
user_id	INT	Идентификатор пользователя
activity_id	INT	Идентификатор активности
date_info	Date	Дата записи данных
weight	FLOAT	Вес пользователя
height	FLOAT	Рост пользователя

Таблица Activities содержит информацию о видах активностей веб-приложения. Структура данной таблицы приведена в таблице 2.10.

Таблица 2.10 – Структура таблицы Activities

Название поля	Тип данных	Описание
activity_id	INT	Идентификатор активности
activity_type	STRING	Тип активности
activity_name	STRING	Название активности

Таблица DailyActivity содержит информацию о ежедневной активности веб-приложения. Структура данной таблицы приведена в таблице 2.11.

Таблица 2.11 – Структура таблицы DailyActivity

Название поля	Тип данных	Описание
dailyActivityId	INT	Идентификатор ежедневной активности
stepQuantity	INT	Количество шагов
otherActivityTime	FLOAT	Время общей активности
userId	INT	Идентификатор пользователя
date	Date	Дата внесения записи

Таблица Friendship содержит информацию о дружбе между пользователями веб-приложения. Структура данной таблицы приведена в таблице 2.12.

Таблица 2.12 – Структура таблицы Friendship

Название поля	Тип данных	Описание
friend_id	INT	Идентификатор дружбы
user1_id	INT	Идентификатор пользователя, инициирующего дружбу
user2_id	INT	Идентификатор пользователя, принявшего запрос на дружбу
friendship_date	Date	Дата заключения дружбы
isPending	Bool	Статус обработки запроса на дружбу

Таблица Goal хранит информацию целях пользователей, которых они желают достичь. Таблица содержит название и описание цели, а также статус выполнения. Пользователь может создавать любое количество различных целей. Структура таблицы представлена в таблице 2.13.

Таблица 2.13 – Структура таблицы Goal

Название поля	Тип данных	Описание
GoalId	INT	Идентификатор цели
GoalName	STIRNG	Название цели
GoalDescription	STIRNG	Описание цели
userId	INT	Внешний ключ, идентификатор пользователя, который создал цель
status	STIRNG	Статус выполнения цели

Таблица StepsToGoal хранит информацию о действиях, необходимых для достижения цели. Структура таблицы StepsToGoal представлена в таблице 2.14.

Таблица 2.14 – Структура таблицы StepsToGoal

Название поля	Тип данных	Описание
StepId	INT	Идентификатор действия
StepName	STIRNG	Название действия
StepDescription	STIRNG	Описание действия
goalId	INT	Внешний ключ, идентификатор цели, для которой добавляется действие
status	STIRNG	Статус выполнения действия

Таблица 2.15 отображает основные связи между таблицами базы данных.

Таблица 2.15 – Связи между таблицами базы данных

Таблица 1	Поле	Таблица 2	Поле	Тип связи	Описание
Users	user_id	Friendship	User1	Многие ко многим	Дружеские связи между пользователями
Users	user_id	Friendship	User2	Многие ко многим	Дружеские связи между пользователями
Users	user_id	UserData	user_id	Один ко многим	Данные пользователя
Users	user_id	Calls	user_id	Один ко многим	Вызовы, созданные пользователем
Users	user_id	DailyActivity	userId	Один ко многим	Ежедневная активность пользователя
Users	user_id	Goal	userId	Один ко многим	Цели пользователя
Calls	call_id	Challenge	CallId	Один ко многим	Связь вызова с испытанием
Users	user_id	Challenge	SenderId/ReceiverId	Один ко многим	Связь испытания с пользователем
UserData	activity_id	Activities	activity_id	Один ко многим	Связь активности с данными пользователя

Проектирование базы данных для веб-приложения «Спортивный вызов» дает возможность грамотно и эффективно управлять пользователями, вызовами и активностями.

2.3 Архитектура веб-приложения

Архитектура веб-приложения, основанная на клиент-серверной модели, представлена на рисунке 2.3 и в ДП 03.00.ГЧ.

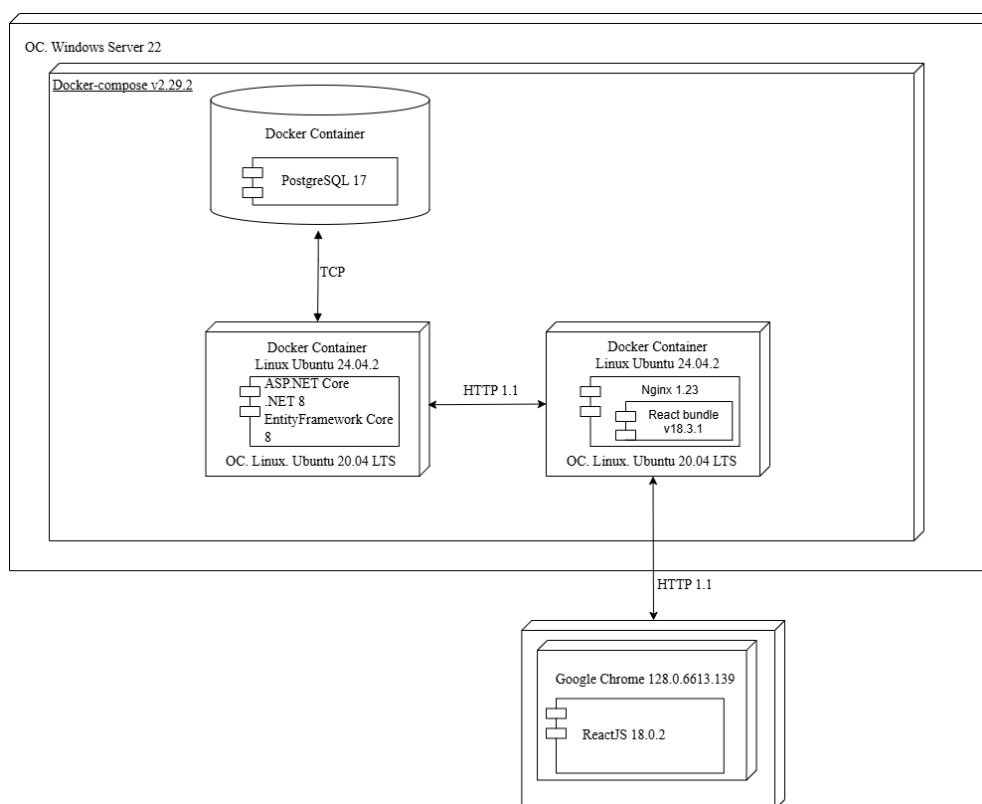


Рисунок 2.3 – Архитектура веб-приложения

В этой модели клиентская часть веб-приложения взаимодействует с сервером через HTTP-запросы с использованием Nginx [7], получая данные для отображения пользователю.

Пояснение назначения каждого элемента веб-приложения представлено в таблице 2.16.

Таблица 2.16 – Назначение элементов архитектурной схемы веб-приложения

Элемент	Назначение
1	2
Frontend Server (React 18)	Обрабатывает HTTP-запросы от клиентов, раздает статические файлы (HTML, CSS, JavaScript) и проксирует API-запросы к Backend Server через HTTPS
Backend Server (ASP.NET Core 8.0)	Реализует бизнес-логику приложения, обрабатывает API-запросы, взаимодействует с базой данных, управляет аутентификацией пользователей и отправляет электронные уведомления через SMTP
Database Server (PostgreSQL 17)	Хранит структурированные данные, включая информацию о пользователях, стажировках, заявках, категориях и навыках
Docker Compose 25 [8]	Оркестрирует контейнеры для всех компонентов приложения, обеспечивая их взаимодействие, автоматическое развертывание и масштабируемость

Продолжение таблицы 2.16

1	2
Entity Framework 8.0 [9]	Используется для работы с базой данных через объектно-ориентированные модели, упрощая создание, обновление и запрос данных

Описание протоколов, используемых при работе веб-приложений, представлено в таблице 2.17.

Таблица 2.17 – Назначение элементов архитектурной схемы веб –приложения

Протокол	Назначение
HTTPS 1.1 [10]	Обеспечивает безопасное шифрованное соединение между клиентами и сервером, защищая данные от перехвата
HTTP 1.1 [11]	Обеспечивает передачу данных между клиентом и сервером
TCP 6 [12]	Обеспечивает надежную передачу данных между клиентом и сервером через стабильное соединение с гарантией доставки и правильного порядка сообщений

Приложение развернуто на Docker, сервер реализован на ASP.NET, разработан с помощью архитектурного паттерна Чистая архитектура C#. Клиент реализован на React.

2.4 Проектирование алгоритма предложения вызова другу

Проектирование алгоритма предложения вызова другу в веб-приложении «Спортивный вызов» начинается с проверки аутентификации пользователя, где система определяет, вошел ли пользователь в систему, и, если нет, предлагает ему авторизоваться, иначе процесс продолжается.

После подтверждения авторизации пользователь переходит к выбору вызова, который он хочет предложить другу, используя доступный список вызовов, таких как ежедневные, еженедельные или месячные, отображаемые в интерфейсе приложения. Затем система открывает список друзей пользователя, загруженный через серверный запрос, и проверяет, есть ли в этом списке доступные контакты, и, если список пуст, пользователю отображается сообщение об отсутствии друзей, а процесс прерывается.

Если друзья найдены, пользователь выбирает одного или нескольких друзей, которым хочет отправить предложение вызова, и подтверждает свой выбор. Далее система формирует данные для отправки на сервер, включая идентификатор вызова, идентификатор пользователя и список выбранных друзей, после чего отправляет запрос на сервер для создания уведомления о предложении вызова. Сервер обрабатывает запрос, регистрируя уведомление для выбранных друзей, и возвращает результат: при успешной отправке пользователю отображается сообщение об успешной передаче предложения, а при ошибке – уведомление о проблеме. После этого система проверяет, были ли уведомления успешно доставлены друзьям, и, если да, процесс завершается,

обновляя интерфейс пользователя с информацией о предложенных вызовах, а если доставка не удалась, отображается сообщение об ошибке.

Блок-схема алгоритма предложения вызова другу представлена на рисунке 2.4 и в ДП 00.04.ГЧ.

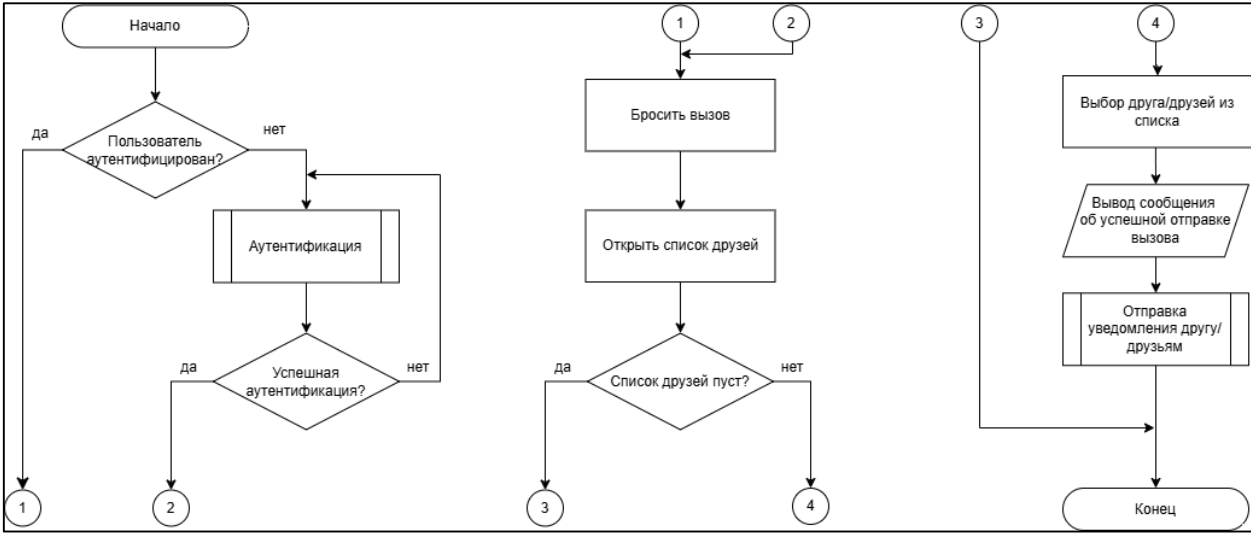


Рисунок 2.4 – Блок-схема алгоритма предложения вызова другу

Разработанный алгоритм предложения вызова другу обеспечивает надежный способ доставки уведомления о новом вызове пользователю либо пользователям.

2.5 Проектирование алгоритма добавления активности

Диаграмма последовательности, демонстрирующая последовательность выполнения алгоритма добавления активности, представлена на рисунке 2.5 и в ДП 00.05.ГЧ.

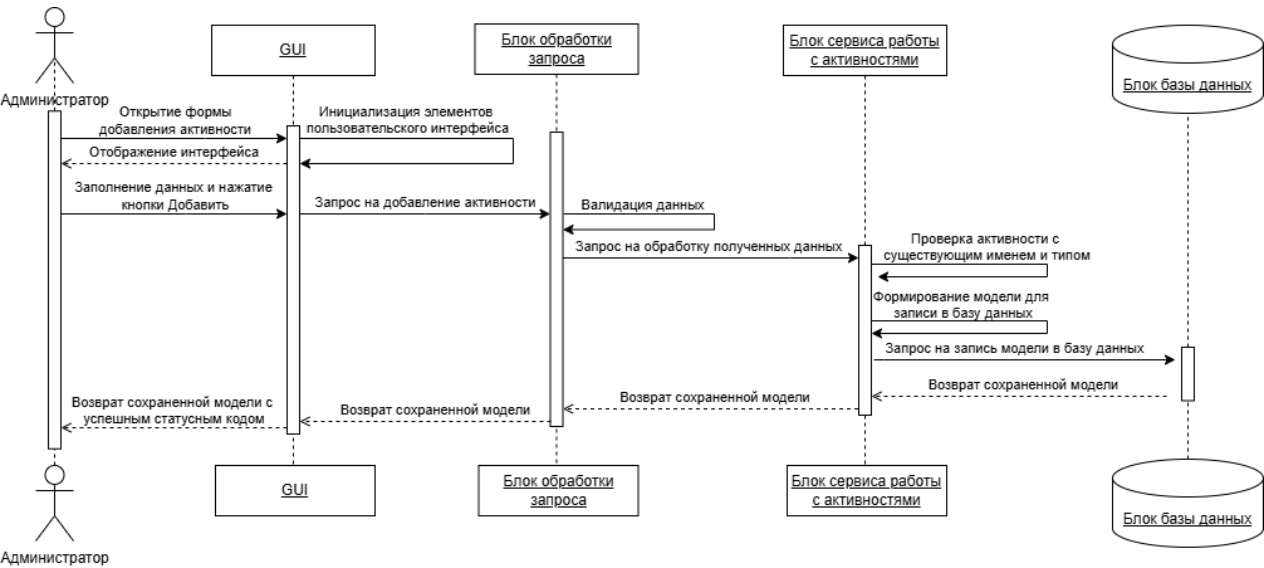


Рисунок 2.5 – Диаграмма последовательности алгоритма добавления активности

Проектирование алгоритма добавления активности в веб-приложении «Спортивный вызов» на основе представленной диаграммы последовательности начинается с взаимодействия администратора через графический интерфейс (GUI), где он инициирует процесс добавления данных активности, что приводит к вызову метода обработки данных в слое бизнес-логики. В этом слое данные проходят проверку на корректность, после чего интерпретатор преобразует их в формат, подходящий для дальнейшей обработки, и передает запрос на добавление активности в слой бизнес-логики, где происходит анализ данных и их валидация.

Если данные корректны, слой бизнес-логики отправляет запрос на сохранение в слой базы данных, который в свою очередь обрабатывает запрос и сохраняет информацию об активности в соответствующей таблице, после чего возвращает подтверждение об успешном сохранении обратно в слой бизнес-логики. Этот слой передает подтверждение через интерпретатор в GUI, где администратору отображается сообщение об успешном добавлении активности.

В случае возникновения ошибки на любом этапе – будь то некорректные данные, проблемы с интерпретацией или сбой в базе данных – система возвращает соответствующее сообщение об ошибке через обратный путь: от слоя базы данных или бизнес-логики через интерпретатор к GUI, где администратор получает уведомление о проблеме.

Аналогичный процесс повторяется для второго администратора, который также взаимодействует с GUI, передает данные через слой бизнес-логики и интерпретатор, получает подтверждение или ошибку от слоя базы данных, обеспечивая синхронную работу с системой и гарантируя, что все изменения активности корректно отражаются в базе данных для последующего использования в приложении.

2.6 Выводы по разделу

В рамках второго раздела была определена функциональность веб-приложения, составлена диаграмма вариантов использования для каждой из ролей, а также подробно разобраны роли и их функциональные возможности.

Была спроектирована логическая схема базы данных, определены таблицы и их назначение, определены первичные и внешние ключи, а также связи между таблицами.

Спроектирована диаграмма развёртывания веб-приложения, описано назначение элементов архитектурной схемы веб-приложения.

Было выполнено проектирование алгоритма предложения вызова другу в виде блок-схемы с полным описанием самого алгоритма.

Был спроектирован алгоритм добавления активности в виде диаграммы последовательности с полным описанием самого алгоритма и схемы диаграммы, демонстрирующей последовательность выполнения алгоритма.

3 Реализация веб-приложения

3.1 Обоснование выбора программной платформы

Для разработки серверной части приложения была выбрана платформа ASP.NET благодаря ее высокой производительности и надежности. ASP.NET обеспечивает эффективную обработку запросов и поддержку масштабируемых решений, что позволяет приложению выдерживать большое количество пользователей и нагрузок. Платформа предлагает встроенные механизмы безопасности, включая аутентификацию и защиту от распространенных веб-угроз, что гарантирует защиту данных пользователей. Кроме того, ASP.NET обладает отличной интеграцией с различными базами данных, включая PostgreSQL, что обеспечивает гибкость в управлении данными и их обработке.

3.2 Система управления базами данных PostgreSQL

Для веб-приложения выбрана PostgreSQL – это реляционная система управления базами данных, которая использует таблицы для организации данных. Данные в PostgreSQL структурированы в виде строк и столбцов, где каждый столбец имеет строго определенный тип данных, что обеспечивает целостность и структурированность информации. PostgreSQL поддерживает сложные связи между таблицами через первичные и внешние ключи, что позволяет эффективно работать с данными и гарантировать их согласованность. Скрипт создания базы данных приведен в Приложении А.

3.3 EntityFramework

В качестве ORM для проекта был выбран EntityFramework версии 8.0. Для генерации моделей в проекте использовался ApplicationDbContext. Таблицы в базе данных создавались с помощью миграций командой Add-Migration. Модели представляются в виде классов C#. Коллекция DbSet описывает набор сущностей.

Модель Users в C# описывает пользователя и включает несколько ключевых полей. Поле user_id представляет уникальный идентификатор пользователя, а username содержит уникальное имя пользователя. Пароль хранится в поле password в зашифрованном виде, а поле birthday предназначено для хранения даты рождения пользователя. Роль пользователя задается в поле user_role, которое по умолчанию имеет значение "User", но может быть изменено на "Admin".

Код, описывающий модель Users, приведен в листинге 3.1

					ДП 03.00.ПЗ				
		ФИО	Подпись	Дата					
Разраб.		Викторович И.С.			3 Реализация веб-приложения	Лит.	Лист	Листов	
Пров.		Гончар Е.А.					1	16	
						БГТУ 1-40 01 01, 2025			
Н. контр.		Гончар Е.А.							
Утв.		Смелов В.В.							

```
public class Users
{
    public int user_id { get; set; }
    public string username { get; set; }
    public string password { get; set; }
    public string birthday { get; set; }
    public string user_role { get; set; } = "User";
}
```

Листинг 3.1 – Модель Users

Код, описывающий модель Calls, приведен в листинге 3.2

```
public class Calls
{
    public int call_id { get; set; }
    public string call_name { get; set; }
    public int? friend_id { get; set; }
    public string call_date { get; set; }
    public string status { get; set; }
    public string description { get; set; }
    public int user_id { get; set; }
}
```

Листинг 3.2 – Модель Calls

Модель Calls в C# описывает звонок и включает несколько ключевых полей. Поле `call_id` представляет уникальный идентификатор звонка, а `call_name` содержит название звонка. Поле `friend_id` указывает идентификатор друга, если он связан с данным звонком, а `call_date` содержит дату звонка. Статус звонка фиксируется в поле `status`, а дополнительная информация о звонке записывается в поле `description`. Поле `user_id` связывает звонок с конкретным пользователем.

Код, описывающий модель Challenges, приведен в листинге 3.3

```
public class Challenge
{
    public int ChallengeId { get; set; }
    public int SenderId { get; set; }
    public int ReceiverId { get; set; }
    public int CallId { get; set; }
    public string Status { get; set; }
    public DateTime SentAt { get; set; }
    public DateTime? RespondedAt { get; set; }
    public Users Sender { get; set; }
    public Users Receiver { get; set; }
    public Calls Call { get; set; }
}
```

Листинг 3.3 – Модель Challenges

Модель Challenge в C# описывает вызов между пользователями и содержит основные поля для управления информацией о вызове. Поле ChallengeId представляет уникальный идентификатор вызова. Поля SenderId и ReceiverId фиксируют идентификаторы отправителя и получателя вызова соответственно, а CallId указывает на связанный с вызовом звонок. Статус вызова хранится в поле Status, дата отправки записывается в поле SentAt, а дата ответа фиксируется в поле RespondedAt, если она имеется.

Код, описывающий модель Activities, приведен в листинге 3.4

```
public class Activities {
    public int activity_id { get; set; }
    public string activity_name { get; set; }
    public string activity_type { get; set; }
    [JsonIgnore]
    public ICollection<UserData> UserData { get; set; }
}
```

Листинг 3.4 – Модель Activities

Модель Activities в C# описывает различные виды активности и содержит ключевые поля для хранения информации о них. Поле activity_id представляет уникальный идентификатор активности, поле activity_name хранит название активности, а activity_type фиксирует тип активности, например, спортивная или социальная.

Код, описывающий модель UserData, приведен в листинге 3.5

```
public class UserData {
    public int data_id { get; set; }
    public int user_id { get; set; }
    public int activity_id { get; set; }
    public DateTime date_info { get; set; }
    public float weight { get; set; }
    public float height { get; set; }
}
```

Листинг 3.5 – Модель UserData

Модель UserData в C# описывает данные пользователя, связанные с его активностями, и включает ключевые поля для хранения этой информации. Поле data_id представляет уникальный идентификатор записи, поле user_id связывает запись с конкретным пользователем, а activity_id указывает на связанную активность. Поле date_info хранит дату, на которую относятся данные, а поля weight и height содержат информацию о весе и росте пользователя на указанную дату.

Модель DailyActivity в C# описывает ежедневную активность пользователя и содержит ключевые поля для хранения информации. Поле dailyActivityId представляет уникальный идентификатор записи активности. Поле stepQuantity хранит количество шагов, выполненных пользователем за

день, а поле `otherActivityTime` (с возможным значением `null`) фиксирует время, потраченное на другие виды активности. Поле `userId` связывает запись с конкретным пользователем, а поле `date` содержит дату, на которую относится эта активность.

Код, описывающий модель `DailyActivity`, приведен в листинге 3.6

```
public class DailyActivity {
    public int dailyActivityId { get; set; }
    public int stepQuantity { get; set; }
    public float? otherActivityTime { get; set; }
    public int userId { get; set; }
    public DateTime date { get; set; }
    [JsonIgnore]
    public Users User { get; set; }
}
```

Листинг 3.6 – Модель `DailyActivity`

Код, описывающий модель `Friendship`, приведен в листинге 3.7

```
public class Friendship
{
    public int friend_id { get; set; }
    public int user1_id { get; set; }
    public int user2_id { get; set; }
    public string friendship_date { get; set; }
    public bool IsPending { get; set; }
}
```

Листинг 3.7 – Модель `Friendship`

Модель `Friendship` в `C#` описывает отношения дружбы между пользователями и содержит ключевые поля для хранения информации. Поле `friend_id` представляет уникальный идентификатор дружбы, а поля `user1_id` и `user2_id` указывают на идентификаторы пользователей, участвующих в дружбе. Поле `friendship_date` хранит дату установления дружбы, а поле `IsPending` определяет, находится ли заявка на дружбу в ожидании подтверждения.

Код, описывающий модель `Goals`, приведен в листинге 3.8

```
public class Goals
{
    public int GoalId { get; set; }
    public string GoalName { get; set; }
    public string GoalDescription { get; set; }
    public int UserId { get; set; }
    public string Status { get; set; }
    public ICollection<StepsToGoal> Steps { get; set; }
}
```

Листинг 3.8 – Модель `Goals`

Модель Goals в C# хранит цели (спортивные задачи) пользователя и содержит ключевые поля для хранения информации. Поле GoalId представляет уникальный идентификатор цели, поле GoalName хранит название цели. Поле GoalDescription хранит описание цели. Поле UserId является внешним ключом для связи с таблицей Users, а поле Status – актуальный статус цели/задачи.

Код, описывающий модель StepsToGoal, приведен в листинге 3.9

```
public class StepsToGoal
{
    public int StepId { get; set; }
    public string StepName { get; set; }
    public string StepDescription { get; set; }
    public int GoalId { get; set; }
    public string Status { get; set; }
    public Goals Goal { get; set; }
}
```

Листинг 3.9 – Модель StepsToGoal

Модель StepsToGoal в C# хранит шаги/действия пользователя для достижения цели. Поле StepId представляет уникальный идентификатор действия/шага, поле StepName хранит название действия. Поле StepDescription хранит описание действия. Поле GoalId является внешним ключом для связи с таблицей Goals, а поле Status – актуальный статус действия.

3.4 Программные библиотеки

В процессе разработки серверной части веб-приложения для обеспечения ее функциональности и повышения эффективности работы системы были использованы программные библиотеки, представленные в таблице 3.1.

Таблица 3.1 – Программные библиотеки серверной части

Библиотека	Версия	Назначение
Npgsql [13]	9.0.2	Открытый провайдер данных для PostgreSQL, разработанный для платформы .NET
Microsoft.AspNetCore.Authentication [14]	8.0.11	Промежуточное программное обеспечение ASP.NET Core, которое позволяет приложению принимать маркер-носитель OpenID Connect.
Microsoft EntityFrameworkCore	9.0.0	Объектно-реляционный маппер (ORM) для платформы .NET. Он поддерживает LINQ-запросы, отслеживание изменений, обновление данных и миграции схемы.
BCrypt.Net-Next [15]	4.0.3	Использует вариант схемы формирования ключей алгоритма шифрования Blowfish и вводит фактор нагрузки (work factor), который позволяет определить, насколько ресурсоемкой будет хеш-функция, обеспечивая "защищенность на будущее".

В процессе разработки клиентской части веб-приложения были задействованы программные библиотеки, представленные в таблице 3.2.

Таблица 3.2 – Программные библиотеки клиентской части

Библиотека	Версия	Назначение
react	19.0.0	Основная библиотека для создания пользовательских интерфейсов на основе компонентов.
react-dom	19.0.0	Библиотека, используемая для рендеринга компонентов React в DOM, обеспечивая взаимодействие между React и веб-браузером.
react-router-dom [16]	7.0.2	Библиотека для маршрутизации в приложениях React.
@mui/material [17]	6.3.0	Библиотека компонентов пользовательского интерфейса от Material-UI.
axios [18]	1.7.9	Библиотека для выполнения HTTP-запросов.
react-toastify [19]	9.1.1	Библиотека для отображения уведомлений (тостов) в React-приложениях.
jwt-decode [20]	3.1.2	Утилита для декодирования JSON Web Token (JWT).

Программные библиотеки позволяют упростить реализацию веб-приложения, а также создать дизайн.

3.5 Структура серверной части

Основные компоненты структуры серверной части включают в себя несколько ключевых элементов, которые обеспечивают эффективную работу приложения:

- Маршрутизаторы – управляют маршрутами и направляют запросы к соответствующим контроллерам.
- Контроллеры – обрабатывают запросы от клиента, выполняют бизнес-логику через сервисы и возвращают ответы.
- Middleware – промежуточные обработчики, используемые для валидации данных и обеспечения безопасности.

Серверная часть представляет из себя приложение, написанное на чистой архитектуре. В таблице 3.3 приведен состав приложения.

Таблица 3.3 – Состав серверной части

Проект	Назначение
Generator.API	Веб-приложение, содержащее контроллеры, middleware, DTOs.
Generator.Domain	Библиотека классов, содержащая модели.
Generator.Infrastructure	Библиотека классов, содержащая миграции, репозитории, интерфейсы репозитория и контекст БД.
Generator.Application	Библиотека классов, содержащая сервисы, интерфейсы сервисов и класс UnitOfWork.

Таблица соответствия маршрутов контроллерам и функциям в исходном коде представлена в таблице 3.4.

Таблица 3.4 – Контроллеры и функции маршрутов

Метод	Маршрут	Контроллер	Метод контроллера
POST	/account/register	AccountController	Register
POST	/account/login	AccountController	Login
POST	/account/logout	AccountController	Logout
POST	/activity/	ActivityController	AddActivity
DELETE	/activity/{id}	ActivityController	DeleteActivity
GET	/activity/activities	ActivityController	GetAllActivities
POST	/activity/add-daily-activity	ActivityController	AddDailyActivity
GET	/activity/daily-activity	ActivityController	GetDailyActivityById
PUT	/activity/update-daily-activity/{id}	ActivityController	UpdateDailyActivity
POST	/call/generate/daily	CallController	GenerateDailyCall
POST	/call/generate/weekly	CallController	GenerateWeeklyCall
POST	/call/generate/monthly	CallController	GenerateMonthlyCall
POST	/call/update-status	CallController	UpdateCallStatus
GET	/call/user-calls	CallController	GetUserCalls
POST	/challenge/send	ChallengeController	SendChallenge
GET	/challenge/received	ChallengeController	GetReceivedChallenges
POST	/friendship/respond	FriendshipController	AddFriend
POST	/userdata/	UserDataController	AddUserData
GET	/stats/admin	StatsController	GetGlobalStats
GET	/stats/user	StatsController	GetUserStats

В данной таблице представлены основные маршруты для взаимодействия серверной части с клиентской.

3.6 Реализация функций для роли «Гость»

3.6.1 Регистрация

Для гостя доступна регистрация, которая позволяет ему создать учетную запись в системе. Эта возможность реализована в контроллере AccountController.

В данном методе контроллера выполняется процесс регистрации нового пользователя, начиная с проверки наличия пользователя с указанным именем в базе данных, чтобы избежать дублирования учетных записей. Если пользователь с таким именем уже существует, метод возвращает сообщение об ошибке, информирующее о необходимости выбрать другое имя. Если же пользователь отсутствует, метод извлекает пароль из входящих данных и хеширует его с использованием алгоритма BCrypt для обеспечения безопасности хранения. После этого создается новый объект пользователя, содержащий указанное имя, хешированный пароль и дату рождения, которые передаются из запроса. Этот объект добавляется в базу данных через паттерн единицы работы (Unit of Work), который обеспечивает согласованность операций, после чего изменения сохраняются в базе данных. В завершение метод возвращает подтверждение успешной регистрации, уведомляя клиента о том, что новый пользователь был создан.

Реализация этого метода приведена в листинге 3.10.

```
[HttpPost("register")]
public IActionResult Register([FromBody] RegisterDto registerDto)
{
    if (_unitOfWork.Users.UserExists(registerDto.username))
    {
        return BadRequest("User already exists.");
    }
    var hashedPassword = BCrypt.Net.BCrypt.HashPassword(registerDto.password);

    var user = new Users
    {
        username = registerDto.username,
        password = hashedPassword,
        birthday = registerDto.birthday
    };
    _unitOfWork.Users.Add(user);
    _unitOfWork.Commit();

    return Ok("User registered successfully.");
}
```

Листинг 3.10 – Реализация HTTP-метода для регистрации

В случае, если пользователь уже зарегистрирован в системе, ему следует пройти авторизацию.

3.6.2 Авторизация

Для гостя доступна авторизация, которая позволяет ему войти в свою учетную запись в системе, обеспечивая безопасный доступ к функционалу приложения. Эта возможность реализована в контроллере AccountController, где применяется паттерн Unit of Work [21] для управления транзакциями и взаимодействия с базой данных. В данном методе реализован процесс аутентификации пользователя, начинающийся с проверки наличия пользователя с указанным именем в базе данных через соответствующий репозиторий.

Если пользователь не найден или введенный пароль не совпадает с сохраненным хешем, метод возвращает сообщение о несанкционированном доступе, информируя о некорректных учетных данных.

При успешной проверке для пользователя генерируется токен с использованием специализированной службы токенов, которая обеспечивает создание уникального и безопасного идентификатора сессии. Программное обеспечение для проверки токена приведено в приложении В.

Для повышения безопасности токен имеет ограниченный срок действия, после которого пользователю потребуется повторная аутентификация. Время жизни токена составляет 30 минут.

Реализация этого метода приведена в листинге 3.11.

```
[HttpPost("login")]
public IActionResult Login([FromBody] LoginDto loginDto)
{
    var user = _unitOfWork.Users.Get-
    ByUsername(loginDto.username);

    if (user == null || !BCrypt.Net.BCrypt.Ver-
    ify(loginDto.password, user.password))
    {
        return Unauthorized("Invalid username or password.");
    }

    var token = _tokenService.GenerateToken(user);

    return Ok(new { Token = token });
}
```

Листинг 3.11 – Реализация HTTP-метода для авторизации

При успешной авторизации в зависимости от роли, пользователь получает доступ к основному ресурсу.

3.7 Реализация функций для роли «Пользователь»

3.7.1 Ввод данных о себе

Пользователи веб-приложения «Спортивный вызов» имеют возможность заполнить форму со своими данными, на основании которых генерируется специализированный вызов, адаптированный к их уровню подготовки и предпочтениям.

Реализация метода добавления данных приведена в приложении Б.

В данном методе контроллера реализуется добавление данных пользователя, причём метод защищён политикой авторизации "UserPolicy". Сначала проверяется, что переданные данные пользователя не равны null, а затем проводится проверка авторизации, извлекая информацию из JWT-токена и проверяя наличие соответствующих утверждений (claims).

После извлечения имени пользователя из токена осуществляется поиск пользователя в базе данных, и, если пользователь или указанный тип активности не найдены, возвращаются соответствующие сообщения об ошибке.

Если данные корректны, создаётся новый объект UserData, который связывает пользователя с активностью, а также сохраняет вес и рост из переданных данных. Попытка добавления нового объекта UserData осуществляется в блоке try-catch, что обеспечивает обработку возможных исключений: при успешном добавлении данных возвращается статус 201 (Created) с информацией о добавленных данных, а в случае ошибки — сообщение об ошибке сервера с кодом 500. Для повышения надёжности метод также логирует все операции, включая успешные добавления и ошибки, что упрощает отладку и мониторинг системы.

3.7.2 Получение уведомлений

Пользователь может получить уведомление, если ему был отправлен запрос в друзья либо если ему брошен вызов. В полученном уведомлении есть выбор действия: либо принять запрос в друзья/вызов, либо отклонить.

```
[HttpGet("notifications")]
public async Task<IActionResult> GetNotifications([FromQuery]
int userId)
{
    var notifications = _context.Friendships
        .Where(f => f.user2_id == userId && f.IsPending)
        .Include(f => f.User1)
        .Select(f => new
        {
            f.friend_id,
            SenderName = f.User1.username,
            SenderId = f.user1_id,
            RecieverName = f.User2.username,
            RecieverId = f.user2_id
        })
        .ToList();

    if (!notifications.Any())
        return Ok(new List<object>());

    return Ok(notifications);
}
```

Листинг 3.12 – Реализация HTTP-метода для уведомления

В полученном уведомлении содержится информация о названии вызова, а также имя друга, отправившего вызов.

3.7.3 Получение вызова

Пользователи веб-приложения «Генератор спортивных вызовов» могут получить ежедневный, еженедельный и ежемесячный вызовы.

В данном методе контроллера реализуется генерация ежедневного вызова для пользователя. Метод защищен атрибутом [Authorize], что требует авторизации для его вызова. Сначала проверяется, что имя пользователя передано и не пустое. Затем в базе данных осуществляется поиск пользователя с указанным именем, включая связанные данные через навигационное свойство UserData. Если пользователь не найден, возвращается сообщение об ошибке.

После успешного поиска пользователя используется сервис ChallengeGeneratorService для генерации вызова, который может включать идентификатор друга, если он передан. Сгенерированный вызов добавляется в базу данных через контекст Calls, и изменения сохраняются. В случае успеха метод возвращает созданный вызов в ответе.

Реализация метода генерации ежедневного вызова приведена в листинге 3.13.

```
[HttpPost("generate/daily")]
public async Task<IActionResult> GenerateDailyCall([FromQuery]
string username, [FromQuery] int? friendId = null) {
    if (string.IsNullOrEmpty(username))
        return BadRequest("Имя пользователя отсутствует.");
    var user = await _context.Users.Include(u => u.UserData)
        .FirstOrDefaultAsync(u => u.username == username);
    var call = _challengeGeneratorService.Generate-
DailyCall(user, friendId);
    _context.Calls.Add(call);
    await _context.SaveChangesAsync();
    return Ok(call);
}
```

Листинг 3.13 – Реализация HTTP-метода для генерации ежедневного вызова

Аналогично выполнены методы для генерации еженедельного и ежемесячного вызовов.

3.7.4 Ввод ежедневной активности

Этот метод добавляет информацию о ежедневной активности пользователя.

```
[HttpPost("add-daily-activity")]
public async Task<IActionResult> AddDailyActivity([FromBody]
DailyActivityDto dailyActivityDto) {
    var existingActivity = await _context.DailyActivities
        .FirstOrDefaultAsync(a => a.userId == dailyActivityDto.UserId
        && a.date.Date == DateTime.UtcNow.Date);
    var user = await _context.Users.FirstOrDefaultAsync(u
=> u.user_id == dailyActivityDto.UserId);
    var dailyActivity = new DailyActivity {
        stepQuantity = dailyActivityDto.StepQuantity,
        otherActivityTime = dailyActivityDto.OtherActivity-
Time.Value,
        userId = dailyActivityDto.UserId,
        date = DateTime.UtcNow,
        User = user
    };
    _context.DailyActivities.Add(dailyActivity);
    await _context.SaveChangesAsync();
    return Ok("Активность успешно добавлена.");
}
```

Листинг 3.14 – Реализация HTTP-метода для добавления ежедневной активности

Он сначала проверяет, существует ли уже запись активности для указанного пользователя на текущую дату.

Если такая запись есть, возвращается ошибка с уведомлением о конфликте. Если активности нет, создается новая запись с указанными данными (например, количество шагов и время другой активности) и сохраняется в базе данных.

Если операция успешна, возвращается сообщение о том, что активность успешно добавлена, в противном случае – сообщение об ошибке.

3.7.5 Возможность бросить вызов другу

Метод `SendChallenge` обрабатывает отправку спортивного вызова от одного пользователя другому. Его реализация представлена в листинге 3.15

```
[HttpPost("send")]
public async Task<IActionResult> SendChallenge([FromBody] SendChallengeDto dto)
{
    if (dto == null)
        return BadRequest("Данные вызова отсутствуют.");

    var challenge = new Challenge
    {
        SenderId = dto.SenderId,
        ReceiverId = dto.ReceiverId,
        CallId = dto.CallId,
        Status = "Pending",
        SentAt = DateTime.UtcNow
    };

    _context.Challenges.Add(challenge);

    await _context.SaveChangesAsync();

    return Ok("Вызов успешно отправлен.");
}
```

Листинг 3.15 – Реализация HTTP-метода для вызова другу

При получении запроса он сначала проверяет, были ли переданы корректные данные. Если данные отсутствуют, возвращается сообщение об ошибке, указывающее на невозможность создания вызова.

Если переданная информация корректна, создается новый вызов, в котором указываются идентификаторы отправителя, получателя и связанного вызова. В случае успешного сохранения данных метод возвращает подтверждение, сообщающее, что вызов был успешно отправлен. После успешного выполнения данной операции вызов отображается в списке вызовов пользователя со статусом «Принято».

Таким образом, этот метод обеспечивает корректное создание и регистрацию новых вызовов в системе, позволяя пользователям бросать друг другу вызовы и фиксировать их в базе данных.

3.7.6 Добавление пользователей в друзья

Метод `AddFriend` отвечает за отправку запроса на добавление дружбы между двумя пользователями. Его реализация представлена в листинге 3.16

```
[HttpPost("add")]
public async Task<IActionResult> AddFriend([FromBody] FriendshipDto dto) {
    var user1 = await _context.Users.FindAsync(dto.user1_id);
    var user2 = await _context.Users.FindAsync(dto.user2_id);

    var exists = await _context.Friendships.AnyAsync(f =>
        (f.user1_id == dto.user1_id && f.user2_id ==
dto.user2_id) ||
        (f.user1_id == dto.user2_id && f.user2_id ==
dto.user1_id));

    var friendship = new Friendship {
        user1_id = dto.user1_id,
        user2_id = dto.user2_id,
        friendship_date = DateTime.UtcNow.ToString(),
        IsPending = true
    };
    _context.Friendships.Add(friendship);

    try {
        await _context.SaveChangesAsync();
    }
    catch (DbUpdateException ex) {
        return StatusCode(500, $"Database error: {ex.Message}");
    }

    return Ok("Friend request sent.");
}
```

Листинг 3.16 – Реализация HTTP-метода для добавления в друзья

При вызове метода сначала осуществляется логирование входных данных, после чего проверяется их наличие. Если переданные данные отсутствуют, возвращается сообщение об ошибке.

Далее происходит поиск пользователей в базе данных по их идентификаторам. Если хотя бы один из пользователей не найден, метод сообщает об этом, завершая выполнение. Затем выполняется проверка на существование дружбы между указанными пользователями. Если дружба уже установлена или запрос на добавление находится в ожидании, пользователь получает уведомление о невозможности повторной отправки. Если все условия соблюдены, создается новый запрос на добавление в друзья, в котором фиксируются идентификаторы пользователей, дата создания запроса и его статус, который указывает, что запрос пока не одобрен. Этот запрос добавляется в базу данных.

3.8 Реализация функций для роли «Администратор»

3.8.1 Добавление и удаление активностей

Администратор системы имеет возможность создавать и удалять виды активности, а также просматривать список созданных активностей с их типами. Реализация метода контроллера создания активностей представлена на листинге 3.17

```
[Authorize(Roles = "Admin")]
[HttpPost]
public IActionResult AddActivity([FromBody] ActivityDto activityDto)
{
    if (activityDto == null)
    {
        return BadRequest("Activity data is required.");
    }

    var activity = new Activities
    {
        activity_name = activityDto.activity_name,
        activity_type = activityDto.activity_type
    };

    _activityService.AddActivity(activity);

    return CreatedAtAction(nameof(GetActivityById), new { id = activity.activity_id }, activity);
}
```

Листинг 3.17 – Создание активности

Если данные корректны, создается новый объект `Activities`, в котором указываются параметры, полученные из `ActivityDto`, включая название активности и ее тип. После успешного добавления активности метод возвращает ответ с кодом 201 Created, который указывает на успешное создание ресурса.

3.8.2 Ведение статистики

Администратор веб-приложения «Спортивный вызов» обладает возможностью просматривать как глобальную статистику, отражающую общую активность всех пользователей, так и детализированную статистику каждого пользователя в отдельности.

Код для функции получения статистики представлен в листинге Б.

С помощью данного метода администратор может получить статистику по вызовам конкретного пользователя за определённый период, включая количество завершённых вызовов за текущий месяц и год, а также разбивку по категориям активности.

3.9 Структура клиентской части

Клиентская часть приложения разработана с использованием компонентного подхода, что обеспечивает удобную модульную структуру кода и упрощает поддержку и расширение функциональности. Вся основная логика и элементы пользовательского интерфейса расположены в директории `src`, которая содержит различные поддиректории, необходимые для разделения кода на логические блоки.

В данной директории хранятся файлы компонентов, отвечающие за отображение интерфейса, управление состоянием приложения, обработку пользовательских событий и взаимодействие с сервером. Компоненты построены с учетом повторного использования, что позволяет легко изменять и расширять приложение.

Для организации структуры проекта используются поддиректории, каждая из которых выполняет свою роль в архитектуре клиентской части. Их состав и предназначение подробно описаны в таблице 3.5.

Таблица 3.5 – Основные директории проекта в папке `src` и их назначение

Директория	Назначение
<code>components</code>	Включает React-компоненты, предназначенные для создания элементов пользовательского интерфейса веб-приложения.
<code>context</code>	Глобальное состояние для управления данными между компонентами React.
<code>routes</code>	Хранит маршруты веб-приложения для навигации между страницами.
<code>services</code>	Хранит сервисы регистрации и авторизации, а также путь к серверу.

Таблица 3.6 показывает основные компоненты. В назначении описаны основные функции, реализованные в рамках этих компонентов.

Таблица 3.6 – Основные компоненты и их назначение

Компонент	Назначение
<code>AdminDashboard</code>	Представляет из себя страницу администратора. Содержит реализацию всех функций пользователя «Администратора».
<code>UserDashboard</code>	Представляет из себя страницу пользователя. Содержит реализацию функций получения вызова, просмотра списка друзей, поиска пользователей и других функций пользователя.
<code>CallList</code>	Представляет из себя список вызовов пользователя, реализует функции смены статуса вызова. Возможность бросить вызов другу реализована в модальном окне в выборе друга через выпадающее меню.
<code>ChallengeComponent</code>	Компонент для отображения уведомления пользователю с предложением о вызове от друга.
<code>UserActivityInput</code>	Компонент реализует функцию ввода, обновления и просмотра активности пользователя.

Функции работы с токеном на клиенте представлены в приложении Г.

3.10 Выводы по разделу

Таким образом, было реализовано веб-приложение «Спортивный вызов» со следующими особенностями:

Использована программная платформа ASP.NET Core, обеспечивающая высокую производительность и масштабируемость серверной части приложения, а для хранения данных применялась реляционная СУБД PostgreSQL, известная своей надёжностью и поддержкой сложных запросов.

Для упрощения взаимодействия с базой данных использовался ApplicationDbContext из Entity Framework Core, который автоматизировал создание коллекций на основе моделей, что позволило сократить время разработки и минимизировать ошибки при работе с данными.

Разработана структура веб-приложения, основанная на модульном подходе с применением современных библиотек, таких как React для клиентской части, обеспечивающий динамичный и отзывчивый интерфейс, и ASP.NET Core для серверной части, что гарантирует гибкость и удобство поддержки.

Разработаны и внедрены все необходимые функции для различных ролей – Гостя, Пользователя и Администратора, включая такие возможности, как авторизация, персонализированный подбор спортивных вызовов, детализированная статистика с визуальными графиками, взаимодействие с друзьями через совместные активности и управление контентом, что обеспечивает универсальность приложения и создаёт комфортное, интуитивно понятное окружение для всех категорий пользователей, удовлетворяя их разнообразные потребности.

Общее количество ключевых функций достигло 10, что позволяет охватить все основные сценарии использования, обеспечивая пользователям полноценный, удобный и интуитивно понятный опыт взаимодействия с приложением, отвечающий их потребностям и ожиданиям.

4 Тестирование веб-приложения

4.1 Функциональное тестирование

Функциональное тестирование направлено на проверку корректности работы функций веб-приложения в соответствии с установленными требованиями. Для проверки корректности работы всех функций разработанного веб-приложения было проведено ручное тестирование, а последовательность действий и полученные результаты приведены в таблице 4.1. Нумерация тестируемых функций соответствует идентификаторам вариантов использования, отображенных на диаграмме, представленной в документе ДП 01.00.ГЧ.

Таблица 4.1 – Результаты тестирования функций веб-приложения

№	Шаги	Ожидаемый результат
1	2	3
Регистрация		
1	1. Нажать «Зарегистрироваться». 2. Ввести пароль, имя и дату рождения. 3. Нажать «Зарегистрироваться».	Создается новая учетная запись
Авторизация		
2	1. Ввести имя и пароль своего аккаунта. 2. Нажать «Войти».	Переход на главную страницу
Получение вызова		
3	1. Нажать на кнопку «Получить вызов». 2. Ознакомиться с предложенным вызовом. 3. Нажать кнопку «Принять»	Новый вызов отображается в списке вызовов
Просмотр списка вызовов		
4	1. Зайти в свой аккаунт. 2. Перейти в раздел вызовов.	Отображается список принятых вызовов
Смена статуса вызовов		
5	1. Перейти в раздел вызовов. 2. Нажать на кнопку «Выполнено» или «Не выполнено».	Статус сменился на противоположный
Возможность бросить вызов другу		
6	1. Перейти в раздел вызовов. 2. Нажать кнопку «Бросить вызов». 3. Из списка выбрать одного или нескольких друзей. 4. Нажать на кнопку подтверждения.	Друзьям пришло уведомление с предложением о вызове
Ввод ежедневной активности		
7	1. Нажать «Ввести ежедневную активность»	Данные сохранены

					ДП 04.00.ПЗ		
		ФИО	Подпись	Дата			
Разраб.	Викторович И.С.				4 Тестирование веб-приложения		
Пров.	Гончар Е.А.						
Н. контр.	Гончар Е.А.						
Утв.	Смелов В.В.						
					Лит.	Лист	Листов
						1	4
					БГТУ 1-40 01 01, 2025		

Продолжение таблицы 4.1

1	2	3
	2. Ввести количество шагов и общее время активности. 3. Нажать «Сохранить».	
Просмотр ежедневной активности		
8	1. Нажать «Просмотреть свою активность». 2. Ознакомиться с отображаемыми данными.	Отображаются данные об активности пользователя
Обновление ежедневной активности		
9	1. Нажать «Просмотреть свою активность». 2. Нажать «Обновить». 3. Ввести новые значения. 4. Нажать «Сохранить».	Данные обновлены и сохранены
Ввод данных о себе		
10	1. Перейти в раздел о себе. 2. Заполнить предложенные поля. 3. Нажать «Сохранить».	Данные успешно сохранены
Просмотр данных о себе		
11	1. Перейти в раздел о себе. 2. Ознакомиться с отображаемыми данными.	Отображается список всех записей, заполненных пользователем.
Добавление пользователей в друзья		
12	1. Перейти в раздел поиска пользователей. 2. Ввести в поисковое поле имя пользователя. 3. Среди предложенных вариантов выбрать нужный вариант. 4. Нажать «Добавить в друзья».	Пользователю отправлена заявка на добавление в друзья
Поиск пользователей		
13	1. Перейти в раздел поиска пользователей. 2. Ввести в поисковое поле имя пользователя. 3. Среди предложенных вариантов выбрать нужный вариант.	По результатам запроса найдены пользователи
Получение уведомлений		
14	1. Перейти в раздел поиска пользователей. 2. Ввести в поисковое поле имя пользователя. 3. Среди предложенных вариантов выбрать нужный вариант. 4. Нажать «Добавить в друзья».	Пользователю отправлена заявка на добавление в друзья
Запрос на дружбу		
15	1. Отправить заявку в друзья пользователю. 2. В разделе уведомлений должна отобразиться заявка на добавление. 3. Нажать на кнопку «Принять» в выбранном уведомлении.	Отображается соответствующее уведомление. Пользователь добавлен в друзья
Предложение вызова		
16	1. Отправить предложение о вызове другу. 2. В разделе уведомлений должна отобразиться заявка на новый вызов.	Отображается соответствующее уведомление. Вызов добавлен в список вызовов

Продолжение таблицы 4.1

1	2	3
	3. Нажать на кнопку «Принять» в выбранном уведомлении.	
Ведение статистики		
17	1. Зайти в систему как администратор. 2. Ознакомиться с данными.	Данные о вызовах пользователей отображаются
Просмотр глобальной статистики		
18	1. Зайти в систему. 2. Ознакомиться с данными.	Глобальная статистика отображается
Просмотр статистики конкретного пользователя		
19	1. Зайти в систему. 2. Нажать «Статистика». 3. Ознакомиться со статистикой пользователя.	Отображается статистика пользователя
Просмотр активностей		
21	1. Зайти в систему как администратор. 2. В разделе активностей ознакомиться со списком.	Отображается список активностей
Добавление/удаление активностей		
22	1. Зайти в систему как администратор. 2. В разделе активностей заполнить предложенные поля. 3. Нажать «Добавить».	Новая активность добавлена и отображается в списке активностей
Поиск пользователей		
23	1. Зайти в систему как администратор. 2. Перейти в раздел пользователей. 3. В поисковой строке ввести имя пользователя.	Найдены пользователи по запросу
Добавление своей цели		
24	1. Перейти в раздел целей. 2. Нажать «Добавить цель». 3. Заполнить предложенную форму. 4. Нажать «Добавить».	Цель добавлена
Добавление шагов достижения цели		
25	1. Перейти в раздел целей. 2. Выбрать цель. 3. Нажать «Добавить шаг». 4. Заполнить предложенную форму. 5. Нажать «Добавить»	Шаг к цели добавлен

Таким образом, были протестированы ключевые функции веб-приложения, включая основные аспекты его работы. В ходе тестирования проверялась корректность обработки запросов на регистрацию, авторизацию и управление пользовательскими данными. Особое внимание было уделено функциональности, связанной с вызовами, включая их создание, обновление статусов и получение статистики. Также были протестированы механизмы взаимодействия между пользователями, такие как отправка и обработка запросов на дружбу.

Тестирование охватило как позитивные сценарии, когда действия выполнялись в рамках ожидаемого поведения, так и негативные сценарии,

направленные на проверку обработки ошибок и исключительных ситуаций. Это позволило убедиться в надежности и устойчивости приложения к некорректным данным и нештатным ситуациям.

Результаты тестирования подтверждают, что приложение корректно обрабатывает запросы, возвращает ожидаемые ответы и успешно сохраняет изменения в базе данных. Проведенное тестирование обеспечивает уверенность в стабильной работе ключевых функций веб-приложения и его готовности к использованию пользователями.

4.2 Выводы по разделу

Проведено тщательное ручное тестирование всех ключевых функций веб-приложения, что позволило детально оценить их работоспособность, стабильность и удобство использования, обеспечивая высокое качество пользовательского опыта.

Функциональность системы проверена на соответствие ожидаемым результатам, что подтверждает ее корректную работу.

Количество выполненных тестов составило 25, что обеспечивает покрытие тестами на уровне 100%.

5 Руководство пользователя

5.1 Руководство пользователя для роли «Гость»

5.1.1 Авторизация

При запуске веб-приложения открывается страница авторизации, представленная на рисунке 5.1.

Зарегистрироваться'."/>

Рисунок 5.1 – Страница с формой авторизации

Если пользователь зарегистрирован, он может ввести свои данные и зайти в систему.

5.1.2 Регистрация

Если у пользователя ещё нет аккаунта в веб-приложении «Спортивный вызов», он имеет возможность создать его, перейдя на страницу регистрации. Для этого в интерфейсе предусмотрена ссылка «Зарегистрироваться», которая расположена на странице входа и отображается в виде интерактивного текстового элемента, выполненного в акцентированном стиле для привлечения внимания пользователя. При нажатии на данную ссылку происходит плавный переход на страницу регистрации, реализованный через клиентскую маршрутизацию с использованием React Router, что обеспечивает быстрое и бесшовное переключение без полной перезагрузки страницы. Страница регистрации, представленная на рисунке 5.2, содержит форму с интуитивно понятным дизайном, включающую поля для ввода имени пользователя, пароля с глазком для просмотра введенных символов и даты рождения.

					ДП 05.00.ПЗ						
		ФИО	Подпись	Дата							
Разраб.	Викторович И.С.				5 Руководство пользователя			Лит.	Лист	Листов	
Пров.	Гончар Е.А.									1	12
								БГТУ 1-40 01 01, 2025			
Н. контр.	Гончар Е.А.										
Утв.	Смелов В.В.										

The registration form is titled "Регистрация". It contains three input fields: "Имя пользователя *" with the value "Maria", "День рождения *" with the value "17.03.2003" and a calendar icon, and "Пароль *" with masked characters ".....". Below the fields is a large orange button labeled "ЗАРЕГИСТРИРОВАТЬСЯ". At the bottom, there is a link "Уже есть аккаунт? Войти".

Рисунок 5.2 – Страница регистрации

После успешной регистрации, происходит переадресация на страницу авторизации, где гость может ввести свои данные и войти в систему.

5.2 Руководство пользователя для роли «Пользователь»

5.2.1 Ввод данных о себе

После успешной авторизации, пользователь переходит на главную страницу. Воспользовавшись меню, пользователь переходит в раздел данных, где ему предлагается заполнить форму, которая представлена на рисунке 5.3.

The form is titled "Добавить данные пользователя" and is part of a user profile page. The page header includes a greeting "Добро пожаловать, Viktor!" and a navigation menu with items: "ВЫЗОВЫ", "ЦЕЛИ", "ДАННЫЕ", "СТАТИСТИКА", "ДРУЗЬЯ", "УВЕДОМЛЕНИЯ", and "ВЫЙТИ". The form itself has a tabbed interface with "1" selected. It contains two buttons at the top: "ВВЕСТИ АКТИВНОСТЬ ЗА СЕГОДНЯ" and "ПРОСМОТРЕТЬ СВОЮ АКТИВНОСТЬ". Below these are four input fields: "Тип активности" (dropdown menu with "Командный спорт" selected), "Название активности" (dropdown menu with "Хоккей" selected), "Вес (кг)", and "Рост (см)". At the bottom of the form is a button labeled "ДОБАВИТЬ".

Рисунок 5.3 – Добавление данных пользователя

Пользователь может ввести какие-либо данные о себе в специальную форму. При успешном заполнении, данные будут выведены на странице

пользователя. Данные отображаются в таблице и используются для подбора персонального задания. Данные пользователя представлены на рисунке 5.4.

Ваши данные				
Дата	Тип активности	Название активности	Вес (кг)	Рост (см)
28.05.2025	Командный спорт	Хоккей	77	183

[НАЗАД](#)
[Страница 1 из 1](#)
[ВПЕРЕД](#)

Рисунок 5.4 – Добавление данных пользователя

Для создания персонального вызова используются самые новые данные, предоставленные пользователем.

5.2.2 Получение уведомлений

Пользователь может получать два типа уведомлений: заявка в друзья и предложение вызова. Пользователь переходит в раздел уведомления и выбирает действие с сообщением. Пример заявки представлен на рисунке 5.5

Уведомления			
Имя пользователя	Тип уведомления	Описание	Действие
Ivan	Запрос на дружбу	Не указано	ПРИНЯТЬ ОТКЛОНИТЬ

Рисунок 5.5 – Получение уведомления

В уведомлении содержится информация о типе уведомления, его описание и имя отправителя, а также две кнопки действия.

5.2.3 Получение вызова

Основной функцией приложения является генерация спортивных вызовов. При наличии записей данных о пользователе, генерация становится более специализированной, так как сервис использует предоставленные данные для подбора задания, которое с наибольшей вероятностью подойдет конкретному пользователю в зависимости от его индекса массы тела, возраста и предпочтений. Чтобы сгенерировать вызов, нужно нажать кнопку «Получить вызов» с соответствующим окном.

В зависимости от того, какой тип вызова выбран: ежедневный, еженедельный или ежемесячный, из расчёта на такой срок и выдается задание, то есть, для ежедневного задания пользователю не будет предложен месячный объем и наоборот.

Пример предложения задания представлен на рисунке 5.6.

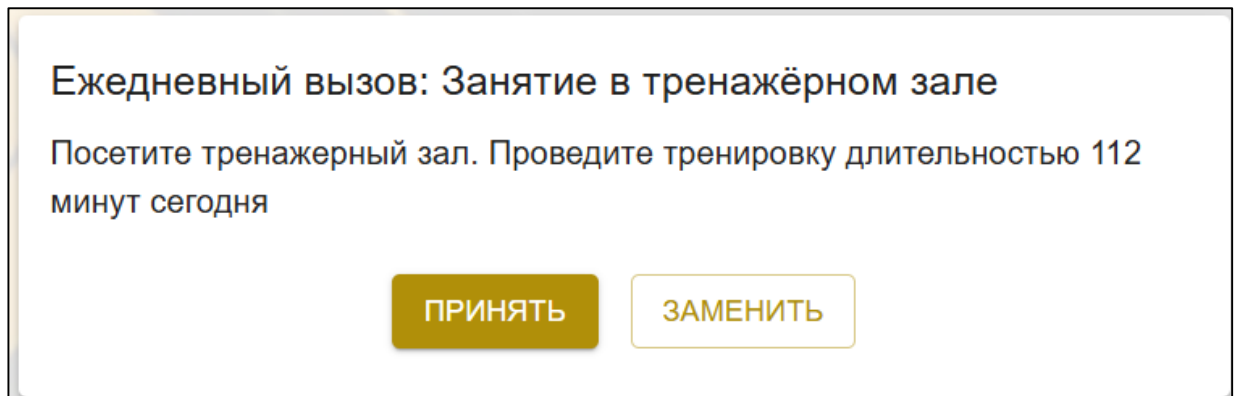


Рисунок 5.6 – Сгенерированный вызов

Вызов можно либо принять, либо отклонить. Принятый вызов заносится в список вызовов пользователя, где можно поменять статус или бросить вызов другу. Список вызовов пользователя представлен на рисунке 5.7.

Еженедельный вызов: Бассейн	Посетите бассейн. Плавайте 238 минут на этой неделе	2025-02-28	Выполнено	<div>ВЫПОЛНЕНО</div> <div>НЕ ВЫПОЛНЕНО</div> <div>БРОСИТЬ ВЫЗОВ</div>
Ежедневный вызов: Занятие в тренажёрном зале	Посетите тренажерный зал. Проведите тренировку длительностью 112 минут сегодня	2025-05-28	Принято	<div>ВЫПОЛНЕНО</div> <div>НЕ ВЫПОЛНЕНО</div> <div>БРОСИТЬ ВЫЗОВ</div>

Рисунок 5.7 – Список вызовов пользователя

Пользователь может менять статус вызова, а также предложить этот вызов другу.

5.2.4 Ввод ежедневной активности

Также пользователь может ввести свою активность за сегодня, нажав на кнопку «Ввести активность за сегодня». Появится модальное окно, где будет предложено ввести количество пройденных за сегодня шагов и время прочей активности. Пользователю не следует беспокоиться об ошибке в введенных данных, так как веб-приложение предусматривает такой вариант, что будет описан далее. Благодаря этому, ввести активность можно в любое время суток, а после обновить результат без необходимости его удаления. Под понятием «Время прочей активности» подразумевается любая спортивная деятельность пользователя помимо ходьбы.

Модальное окно активности будет представлено на рисунке 5.8.

Ввести активность за сегодня

Количество шагов

Время прочей активности (в минутах)

ОТМЕНА **СОХРАНИТЬ**

Рисунок 5.8 – Модальное окно активности

В случае, если пользователь ошибся при вводе данных в модальном окне, можно нажать кнопку «Просмотреть свою активность», где данные, внесенные пользователем, можно отредактировать. Модальное окно редактирования и просмотра активностей представлено на рисунке 5.9.

Обновить активность

Количество шагов

Время прочей активности (в минутах)

ОТМЕНА **СОХРАНИТЬ**

Моя активность

Количество шагов	Прочая ак
8796	34

Действия

ОБНОВИТЬ

ЗАКРЫТЬ

Рисунок 5.9 – Модальное окно редактирования и просмотра активностей

Данные об ежедневной активности можно ввести только один раз в сутки, но можно обновить в любой момент.

5.2.5 Возможность бросить вызов другу

Чтобы предложить пользователю из списка друзей вызов, нужно нажать на кнопку «Бросить вызов». Далее появится модальное окно, где вы можете выбрать друга из своего списка друзей и отправить ему уведомление. Модальное окно подготовки приглашения представлено на рисунке 5.10.

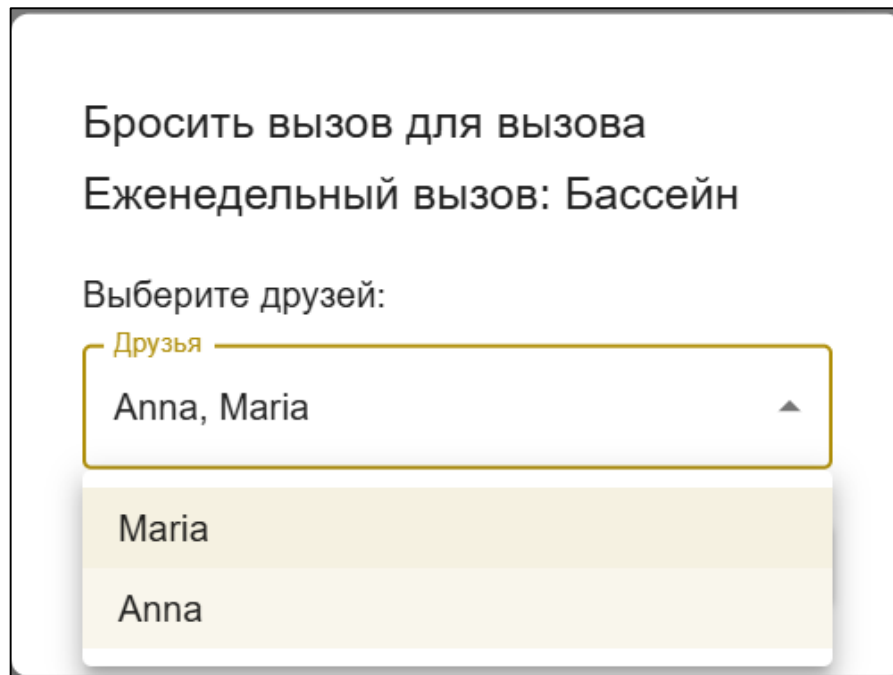


Рисунок 5.10 – Модальное окно для вызова

Можно выбрать как одного, так и нескольких друзей из списка для отправки им предложения вызова.

5.2.6 Добавление пользователей в друзья

Чтобы найти друга, можно воспользоваться поиском. Далее возле каждого пользователя есть кнопка «Добавить в друзья». При нажатии на кнопку пользователю отправляется запрос на дружбу. Как только заявка будет одобрена, пользователь будет добавлен в друзья. Поиск друзей представлен на рисунке 5.11.

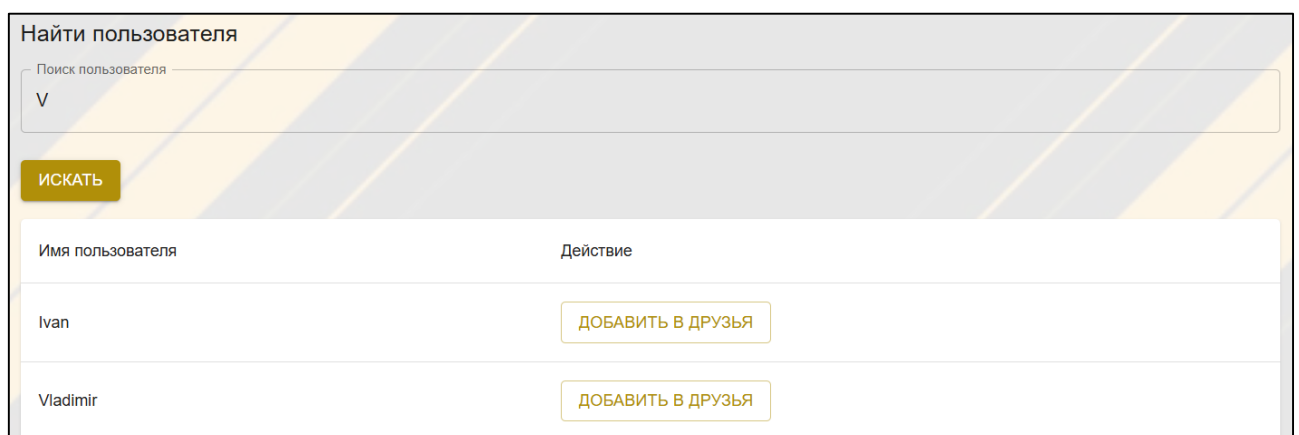


Рисунок 5.11 – Поиск друзей

На рисунке 5.12 представлено уведомление заявки пользователю на дружбу, которое появляется у получателя при нажатии на кнопку «Добавить в друзья». Предлагается на выбор два действия.

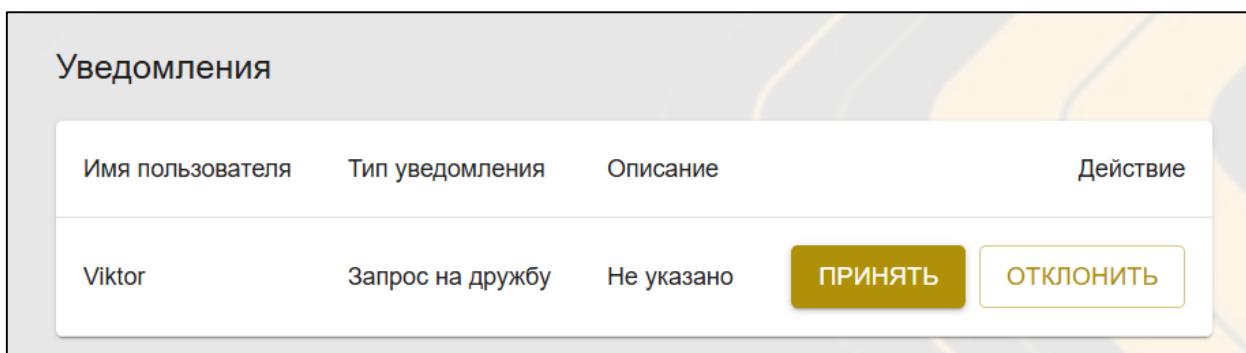


Рисунок 5.12 – Заявка на добавление в друзья

Теперь, когда пользователь добавлен в друзья, как видно из списка друзей, что изображен на рисунке 5.13, можно бросать другу вызов.



Рисунок 5.13 – Список друзей

После добавления в друзья у пользователей есть возможность предлагать друг другу вызовы.

5.2.7 Добавление своей цели

Пользователь может создать свою цель, которой желает достичь, что позволяет ему установить личные ориентиры для мотивации и отслеживания прогресса в рамках веб-приложения «Спортивный вызов». Для этого пользователь переходит в специальный раздел «Цели», доступный через главное меню интерфейса, где ему предоставляется интуитивно понятный доступ к функциям управления целями. В этом разделе он выбирает опцию «Добавить цель», активируя процесс создания новой задачи, после чего открывается форма. Добавление цели, сопровождаемое визуальной обратной связкой, показано на рисунке 5.14, где отображается как заполненная форма, так и результат её успешного сохранения в виде уведомления об успехе, что делает процесс интерактивным и удобным для пользователя.

Рисунок 5.14 – Добавление цели

После создание цель добавляется в список целей пользователя. Задачу можно удалить, обновить ее статус, а также добавить шаги для достижения этой цели. Карточка цели представлена на рисунке 5.15.

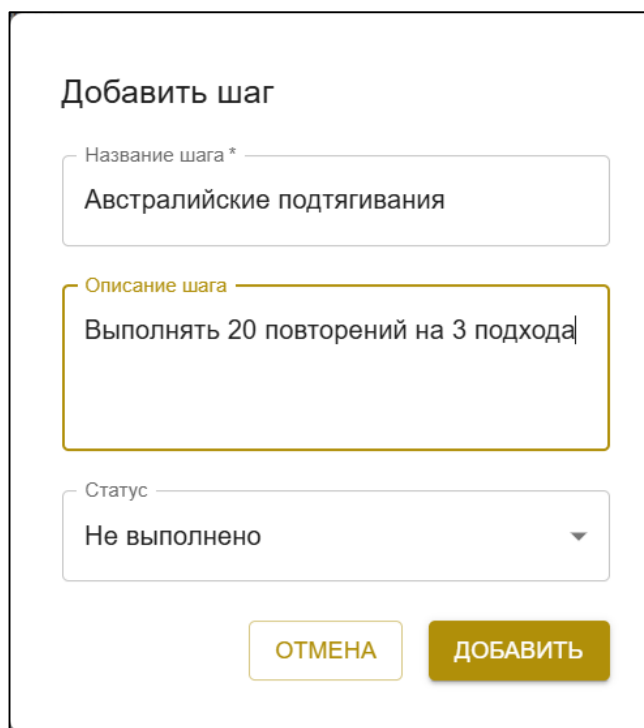
Название шага	Описание	Статус	Действия
Подтягивание с помощью резинок	Делать 10 подтягиваний с помощью резинок	Не выполнено	

Рисунок 5.15 – Карточка цели

Пользователь может ставить множество целей для себя, а также определять необходимое количество шагов, необходимых ему для достижения цели.

5.2.8 Добавление шагов к выполнению цели

Как было описано в предыдущем пункте, пользователь может создавать шаги, они же действия, для достижения определенной цели. Форма создания цели представлена на рисунке 5.16



Добавить шаг

Название шага *

Австралийские подтягивания

Описание шага

Выполнять 20 повторений на 3 подхода

Статус

Не выполнено

ОТМЕНА ДОБАВИТЬ

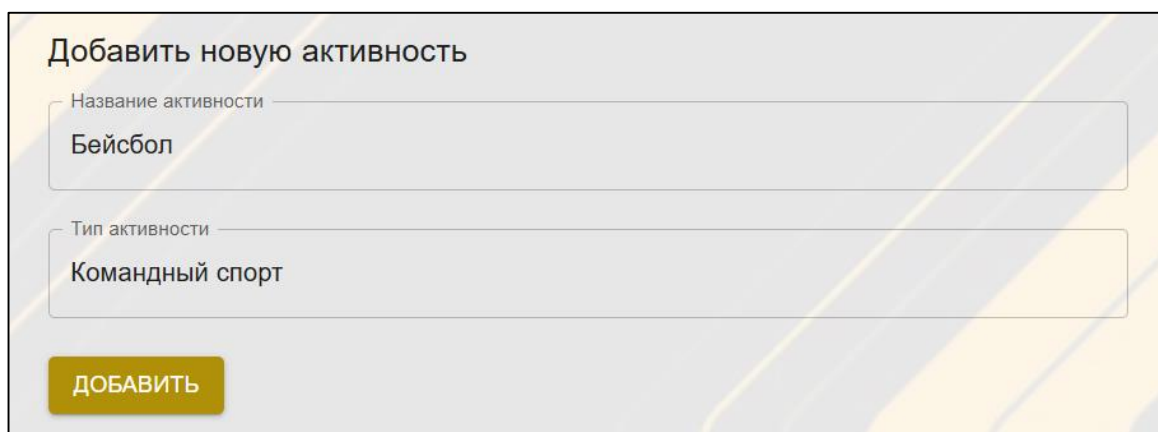
Рисунок 5.16 – Добавления действия

Созданное действие пользователь может редактировать: менять название, описание либо обновлять статус.

5.3 Руководство пользователя для роли «Администратор»

5.3.1 Добавление и удаление активностей

У администратора есть возможность добавлять новый вид активности. Форма добавления представлена на рисунке 5.17.



Добавить новую активность

Название активности

Бейсбол

Тип активности

Командный спорт

ДОБАВИТЬ

Рисунок 5.17 – Добавление активности

Добавленная активность отображается в списке активностей. При заполнении данных о себе пользователь выбирает активность из предложенного списка. Администратор может удалить позицию из списка, представленного на рисунке 5.18.

Название активности	Тип активности	
Футбол	Командный спорт	
Волейбол	Командный спорт	
Баскетбол	Командный спорт	
Бег	Лёгкая атлетика	
Скандинавская ходьба	Лёгкая атлетика	

< 1 2 >

Рисунок 5.18 – Список активностей

Так как пользовательские данные связаны с активностью, то при удалении активности также удаляется пользовательская запись с данной активностью.

5.3.2 Поиск пользователей

Администратор может просматривать список пользователей, а также использовать поисковую строку для поиска и фильтрации пользователей. Список пользователей представлен на рисунке 5.19

Имя пользователя	Действия
string	СТАТИСТИКА
admin	СТАТИСТИКА
Ivan	СТАТИСТИКА
Ilya	СТАТИСТИКА
Maria	СТАТИСТИКА

Рисунок 5.19 – Список пользователей

Также добавлена пагинация для визуального улучшения просмотра списка пользователей.

5.3.3 Ведение статистики

Администратор имеет возможность просмотреть статистику каждого пользователя лично, нажав на кнопку «Статистика». Модальное окно со статистикой отдельного пользователя показано на рисунке 5.20.



Рисунок 5.20 – Модальное окно личной статистики

Также администратор имеет возможность просмотреть глобальную статистику (рисунок 5.21).

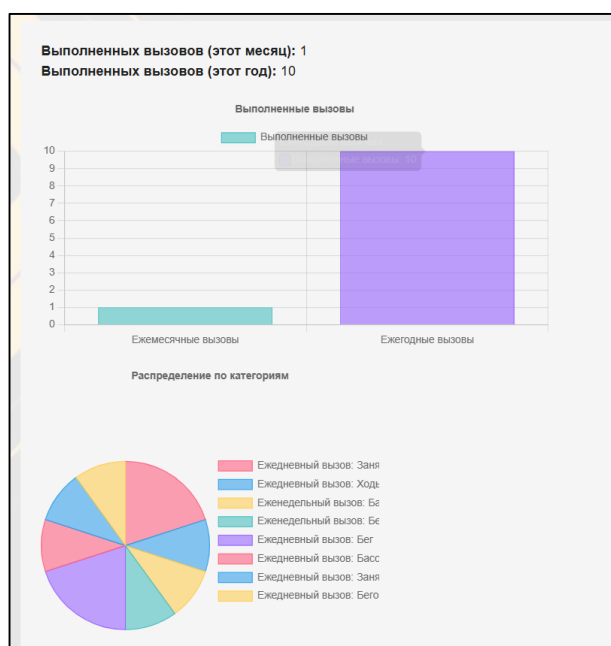


Рисунок 5.21 – Глобальная статистика

Статистические данные играют ключевую роль в улучшении работы приложения «Спортивный вызов», предоставляя разработчикам ценные инсайты для оптимизации функционала и устранения возможных недостатков. Они собираются на основе активности пользователей, таких как выполнение вызовов, добавление данных и взаимодействие с друзьями, что позволяет анализировать популярность различных функций и адаптировать приложение под текущие потребности аудитории. Также статистические данные доступны и обычным пользователям.

5.4 Выводы по разделу

Разработано руководство, описывающее действия пользователей системы: гостя, зарегистрированного пользователя и администратора, с учетом уникальных функций каждой роли, описанных в диаграмме вариантов использования.

Гости могут зарегистрироваться и авторизоваться. Зарегистрированные пользователи могут добавлять данные о себе, получать вызовы, принимать или отклонять их, добавлять в друзья других пользователей, бросать им вызовы, добавлять свои цели и шаги к их выполнению. Администраторы могут просматривать глобальную статистику и статистику определенного пользователя, просматривать список пользователей, а также добавлять или удалять активности, необходимые пользователям для заполнения данных.

Реализована система уведомлений, позволяющая пользователям получать своевременные оповещения о новых вызовах, а также уведомления о действиях друзей, таких как принятие вызова или добавление в друзья, что повышает вовлечённость и поддерживает социальное взаимодействие.

6 Технико-экономическое обоснование проекта

6.1 Общая характеристика разрабатываемого программного средства

При выполнении данного проекта было разработано веб-приложение «Спортивный вызов» для создания и генерации спортивных вызовов и целей, записи своей активности, соревнований в выполнении спортивных заданий со своими друзьями, идеей которого является мотивация пользователей вести здоровый образ жизни и отслеживать свой прогресс.

Целью дипломного проекта было создание веб-приложения для генерации вызовов пользователю на основе его физических показателей, таких как возраст и индекс массы тела и предпочтений, а также предоставить возможность пользователю фиксировать и просматривать свои результаты для отслеживания прогресса и создавать свои собственные цели и шаги для их достижения.

Пользователи приложения могут вводить данные о себе, получать уведомления о новых предложениях вызовов, получать вызовы, вводить ежедневную активность, бросать вызов другу, добавить пользователя в друзья, добавить свою цель и шаги достижения этой цели. Административная часть системы позволяет управлять активностями, пользователями, вести статистику по пользователям и вызовам, выполненным за определенный период.

Во время разработки дипломного проекта были использованы следующие технологии: библиотека React.js для реализации клиентской части, серверная платформа ASP.NET на C#, система управления базами данных PostgreSQL, аутентификация через JSON Web Token, а также адаптивная верстка для поддержки различных устройств. Разработанное программное решение имеет следующие преимущества перед аналогичными образцами, рассмотренными в первой главе работы:

- интуитивно понятный интерфейс с панелью навигации по странице;
- сервис генерации вызовов, основанный обработке данных, полученных от пользователя.

Стратегия монетизации предполагает продажу продукта с передачей прав заказчику.

6.2 Исходные данные для проведения расчетов и маркетинговый анализ

Источниками исходных данных для данных расчетов выступают действующие нормативные правовые акты. Исходные данные для расчета приведены в таблице 6.1.

					ДП 06.00.ПЗ		
		ФИО	Подпись	Дата			
Разраб.	Викторович И.С.				6 Технико-экономическое обоснование проекта		
Пров.	Гончар Е.А.						
Консульт.	Познякова Л.С.						
Н. контр.	Гончар Е.А.						
Утв.	Смелов В.В.						
					Лит.	Лист	Листов
					У	1	10
					БГТУ 1-40 01 01, 2025		

Таблица 6.1 – Исходные данные для расчета

Наименование показателя	Условные обозначения	Норматив
Норматив дополнительной заработной платы, %	$H_{дз}$	10
Ставка отчислений в Фонд социальной защиты населения, %	$H_{фсзн}$	34
Ставка отчислений по обязательному страхованию в БРУСП «Белгосстрах», %	$H_{бгс}$	0,6
Норматив прочих прямых затрат, %	$H_{пз}$	20
Норматив накладных расходов, %	$H_{обп, обх}$	55
Норматив расходов на реализацию, %	$H_{рр}$	10
Ставка НДС, %	$H_{ндс}$	20
Налог на прибыль, %	$H_{п}$	20

Для того чтобы проанализировать стоимость разработки программного средства, в первую очередь необходимо рассмотреть стоимость создания приложений-аналогов. Современный рынок веб-приложений для ведения спортивной активности представлен рядом коммерческих решений, отличающихся по функциональности, дизайну и способу взаимодействия с пользователями. В рамках маркетингового анализа была определена ориентировочная стоимость разработки аналогичных веб-приложений. Результаты анализа представлены в таблице 6.2.

Таблица 6.2 – Анализ стоимости разработки

Продукты-аналоги	Источник	Стоимость, руб	Примечание
Приложение «Спортивный вызов»	https://play.google.com/store/apps/details?id=ru.stayfitt.sportchallenge&hl=ru	41 000	Удобная навигация, соревновательная система
Приложение «Life+»	https://play.google.com/store/apps/details?id=ru.stayfitt.lifeplus&hl=ru	61 000	Развитая соревновательная система, возможность делиться достижениями
Приложение «SPOBI»	https://apkpure.net/ru/sportbi-sports-being-intelligent/com.spobi.android	23 000	Возможность загрузить фото

Средняя стоимость разработки подобных решений варьируется от 23 000 BYN (для простых сайтов с базовым функционалом) до 61 000 BYN (для полноценных соревновательных спортивных платформ). Учитывая функциональные возможности разработанного веб-приложения «Спортивный вызов», включая визуальный генератор спортивных вызовов, авторизацию пользователей, администраторскую панель, а также адаптивный интерфейс, рыночная стоимость проекта оценивается в 42 000 BYN.

Средняя стоимость разработки аналогичного продукта составляет 42 000 рублей.

6.3 Обоснование цены программного средства

6.3.1 Расчет затрат рабочего времени на разработку программного средства

Необходимо посчитать время, которое было затрачено на разработку данного программного средства. В таблице 6.3 представлены затраты рабочего времени на разработку программного средства.

Таблица 6.3 – Затраты рабочего времени на разработку ПС

Содержание работ	Исполнитель	Затраты рабочего времени, часов
Анализ предметной области и сбор требований	Бизнес-аналитик	40
Анализ аналогов и функциональное проектирование	Бизнес-аналитик	30
Утверждение требований и ТЗ	Проектный менеджер	10
Проектирование архитектуры приложения и структуры базы данных	Backend -разработчик	55
Утверждение архитектуры	Проектный менеджер	5
Проектирование UI/UX и создание макетов	Дизайнер	65
Утверждение дизайна	Проектный менеджер	5
Backend-разработка	Backend -разработчик	150
Frontend-разработка	Frontend-разработчик	165
Тестирование приложения	Тестировщик	35
Устранения дефектов	Backend-разработчик	30
	Frontend-разработчик	30
Деплой и настройка <i>production</i> -окружения	Backend-разработчик	15
Управление проектом	Проектный менеджер	60
Разработка технической документации	Frontend-разработчик	15
	Backend-разработчик	15
Всего		725

Таким образом, занятость на проекте бизнес-аналитика составит 70 часов, проектного менеджера – 80 часов, дизайнера – 65 часов, *backend*-разработчика – 265 часа, *frontend*-разработчика – 210 часа, тестировщика – 35 часов, а в сумме 725 часов.

6.3.2 Расчет основной заработной платы

Для определения величины основной заработной платы, было проведено исследование величин заработных плат для специалистов в сфере разработки и определение их часовых ставок. Источником данных служили открытые веб-порталы, официальная отчетность, а также общий средний уровень заработка в сфере информационных технологий в Республике Беларусь.

После определения часовых ставок и трудозатрат исполнителей определяются заработные платы всех исполнителей, а также основная заработная

плата, которая является суммой всех заработных плат исполнителей. Заработная плата отдельного специалиста рассчитывается по формуле 6.1. Результаты подсчетов основных заработных плат специалистов представлены в таблице 6.4.

$$C_{\text{оз}} = T_{\text{раз}} \cdot C_{\text{зп}}, \quad (6.1)$$

где $C_{\text{оз}}$ – основная заработная плата, руб.;

$T_{\text{раз}}$ – трудоемкость, чел./час.;

$C_{\text{зп}}$ – средняя часовая ставка, руб./час.

Таблица 6.4 – Расчет основной заработной платы специалистов

Исполнитель	Затраты рабочего времени, часов	Средняя часовая ставка, руб./час	Основная заработная плата, руб.
Бизнес-аналитик	70	14	980
Проектный менеджер	80	20	1600
Дизайнер	65	12	780
<i>Backend</i> -разработчик	265	17	4 505
<i>Frontend</i> -разработчик	210	15	3 150
Тестировщик	35	11	385
Всего	725		11 400

Таким образом, при разработке программного средства основная заработная плата бизнес-аналитика составит 980 руб., проектного менеджера – 1 680 руб., дизайнера – 960 руб., *backend*-разработчика – 4 335 руб., *frontend*-разработчика – 3 225 руб., тестировщика – 385 руб. Суммарная основная заработная плата всех специалистов веб-приложения составит 11 565 руб.

6.3.3 Расчет дополнительной заработной платы

Дополнительная заработная плата на конкретное программное средство включает выплаты, предусмотренные законодательством о труде, и определяется по нормативу в процентах к основной заработной плате по формуле 6.2.

$$C_{\text{дз}} = \frac{C_{\text{оз}} \cdot N_{\text{дз}}}{100}, \quad (6.2)$$

где $C_{\text{оз}}$ – основная заработная плата, руб.;

$N_{\text{дз}}$ – норматив дополнительной заработной платы, %.

$$C_{\text{дз}} = 11\,400 \cdot 10 / 100 = 1\,140 \text{ руб.}$$

Таким образом дополнительная заработная плата составила 1 140 руб.

6.3.4 Расчет отчислений в Фонд социальной защиты населения и по обязательному страхованию

Отчисления в Фонд социальной защиты населения (ФСЗН) и по обязательному страхованию от несчастных случаев на производстве, и профессиональных заболеваний в БРУСП «Белгосстрах» определяются в соответствии с действующими законодательными актами по нормативу в процентном отношении к фонду основной и дополнительной заработной платы исполнителей и вычисляются по формуле 6.3.

$$C_{\text{фсзн}} = \frac{(C_{\text{оз}} + C_{\text{дз}}) \cdot N_{\text{фсзн}}}{100}, \quad (6.3)$$

где $C_{\text{оз}}$ – основная заработная плата, руб.;

$C_{\text{дз}}$ – дополнительная заработная плата на конкретное ПС, руб.;

$N_{\text{фсзн}}$ – норматив отчислений в Фонд социальной защиты, %.

Отчисления в БРУСП «Белгосстрах» вычисляются по формуле 6.4.

$$C_{\text{бгс}} = \frac{(C_{\text{оз}} + C_{\text{дз}}) \cdot N_{\text{бгс}}}{100}, \quad (6.4)$$

$$C_{\text{фсзн}} = \frac{(11\,400 + 1\,140) \cdot 34}{100} = 4\,263,6 \text{ руб.}$$

$$C_{\text{бгс}} = \frac{(11\,400 + 1\,140) \cdot 0,6}{100} = 75,24 \text{ руб.}$$

Таким образом, общие отчисления в БРУСП «Белгосстрах» составили 75,24 руб., а в фонд социальной защиты населения – 4 263,6 руб.

6.3.5 Расчет суммы прочих прямых затрат

Расходы на конкретное программное средство $C_{\text{пз}}$ включают расходы на приобретение и подготовку специальной технической информации, платных сервисов тестирования и прочие операционные издержки, прямо относимые на проект и рассчитываются по формуле 6.5.

$$C_{\text{пз}} = \frac{C_{\text{оз}} \cdot N_{\text{пз}}}{100}. \quad (6.5)$$

где $N_{\text{пз}}$ – норматив прочих затрат в целом по организации, %.

$$C_{\text{пз}} = 11\,400 \cdot 20 / 100 = 2\,280 \text{ руб.}$$

Таким образом, сумма прочих прямых затрат при разработке веб-приложения «Спортивный вызов» составила 2 280 рублей.

6.3.6 Расчет суммы накладных расходов

Сумма накладных расходов $C_{\text{обп,обх}}$ – произведение основной заработной платы исполнителей на конкретное программное средство $C_{\text{оз}}$ на норматив накладных расходов в целом по организации $H_{\text{обп,обх}}$, по формуле 6.6.

$$C_{\text{обп,обх}} = \frac{C_{\text{оз}} \cdot H_{\text{обп,обх}}}{100}. \quad (6.6)$$

Сумма накладных расходов составит:

$$C_{\text{обп,обх}} = 11\,400 \cdot 55 / 100 = 6\,270 \text{ руб.}$$

Таким образом, сумма накладных расходов составила 6 270 руб.

6.3.7 Сумма расходов на разработку программного средства

Сумма расходов на разработку программного средства C_p определяется как сумма основной и дополнительной заработной платы исполнителей на конкретное программное средство, отчислений на социальные нужды, суммы прочих затрат и суммы накладных расходов, по формуле 6.7.

$$C_p = C_{\text{оз}} + C_{\text{дз}} + C_{\text{фсзн}} + C_{\text{бгс}} + C_{\text{пз}} + C_{\text{обп,обх}}. \quad (6.7)$$

Все данные необходимые для вычисления есть, поэтому можно определить сумму расходов на разработку программного средства.

$$C_p = 11\,400 + 1\,140 + 4\,263,6 + 75,24 + 2\,280 + 6\,270 = 25\,428,84 \text{ руб.}$$

Сумма расходов на разработку программного средства была вычислена на основе данных, рассчитанных ранее, и составила 25 428,84 рублей.

6.3.8 Расходы на сопровождение и адаптацию

Сумма расходов на реализацию программного средства $C_{\text{рп}}$ определяется как произведение суммы расходов на разработки на норматив расходов на сопровождение и адаптацию $H_{\text{рса}}$, по формуле 6.8.

$$C_{\text{рса}} = \frac{C_p \cdot H_{\text{рса}}}{100}, \quad (6.8)$$

где $C_{\text{рса}}$ – сумма расходов на сопровождение и адаптацию ПС, руб.;

C_p – общая сумма расходов на разработку ПС, руб.;

H_{pca} – норматив расходов на сопровождение и адаптацию, %.

Основываясь на исходные данные, расположенные в таблице 6.1 и формулу 6.8, норматив расходов на сопровождение и адаптацию H_{pp} равен 10%. Сумма расходов на реализацию ПС составляет.

$$C_{pca} = 25\,428,84 \cdot 10 / 100 = 2\,542,88 \text{ руб.}$$

Сумма расходов на сопровождение и адаптацию программного средства, определенная по формуле 6.8, составляет 2 542,88 рубля.

6.3.9 Расчет полной себестоимости

Полная себестоимость $C_{п}$ определяется как сумма двух элементов.

Суммы расходов на разработку C_p и суммы расходов на сопровождение и адаптацию программного средства C_{pca} по формуле 6.9.

$$C_{п} = C_p + C_{pca}, \quad (6.9)$$

$$C_{п} = 25\,428,84 + 2\,542,88 = 27\,971,72 \text{ руб.}$$

Получим, что полная себестоимость приложения равна 27 971,72 рубля.

6.3.10 Определение цены

Приложение разрабатывается на заказ и передается заказчику. Его цена определяется желаемой рентабельности 20%. Прибыль от реализации программного средства вычисляется по формуле 6.10.

$$\Pi_{пс} = \frac{C_{п} \cdot Y_{\text{рент}}}{100} \quad (6.10)$$

где $Y_{\text{рент}}$ – уровень рентабельности, %;

$C_{п}$ – полная себестоимость программного средства, руб.

Цена разработки программного средства без налогов находится по формуле 6.11:

$$\Pi_p = C_{п} + \Pi_{пс} \quad (6.11)$$

Сумма налога на добавленную стоимость рассчитывается из формулы 6.12:

$$\text{НДС} = \frac{\Pi_p \cdot H_{\text{ндс}}}{100} \quad (6.12)$$

где Π_p – цена разработки программного средства, руб.;

$N_{\text{ндс}}$ – ставка НДС, %.

Планируемая отпускная цена с НДС вычисляется по формуле 6.13:

$$C_{\text{с ндс}} = C_{\text{р}} + \text{НДС} \quad (6.13)$$

Организация не является резидентом ПВТ, рассчитываем чистую прибыль по формуле 6.14

$$\Pi_{\text{ч}} = \Pi_{\text{пс}} \cdot \left(1 - \frac{N_{\text{п}}}{100}\right) \quad (6.14)$$

где $N_{\text{п}}$ – ставка налога на прибыль, %.

Рассчитаем рентабельность проекта $R_{\text{а}}$, как отношение чистой прибыли к себестоимости по формуле 6.15.

$$R_{\text{а}} = \Pi_{\text{ч}} / C_{\text{п}} \cdot 100\% \quad (6.15)$$

Исходя из вышеперечисленных данных рассчитаем прибыль от реализации программного средства, цену разработки без налогов, сумму налогов на добавленную стоимость, планируемую отпускную цену с НДС, чистую прибыль, а также рентабельность проекта.

$$\begin{aligned} \Pi_{\text{пс}} &= 27\,971,72 \cdot 20 / 100 = 5\,594,34 \text{ руб.} \\ C_{\text{р}} &= 27\,971,72 + 5\,594,34 = 33\,566,06 \text{ руб.} \\ \text{НДС} &= 33\,566,06 \cdot 20 / 100 = 6\,713,21 \text{ руб.} \\ C_{\text{с ндс}} &= 33\,566,06 + 6\,713,21 = 40\,279,27 \text{ руб.} \\ \Pi_{\text{ч}} &= 5\,594,34 \cdot (1 - 20 / 100) = 4\,475,47 \text{ руб.} \\ R_{\text{а}} &= 4\,475,47 / 27\,971,72 \cdot 100\% = 16\% \end{aligned}$$

Планируемая отпускная цена продукта с учетом НДС составляет 40 279,27 руб. Хотя стоимость разработки превышает затраты на использование готовых шаблонных решений, индивидуальное веб-приложение «Спортивный вызов» обеспечивает более высокую гибкость, безопасность и соответствие конкретным требованиям приложения для спорта.

6.4 Выводы по разделу

В рамках данного раздела были проведены экономические расчеты и определение преимуществ разрабатываемого проекта по сравнению с аналогичными решениями, на основе которых была определена себестоимость разрабатываемого программного средства, а также прогнозируемая отпускная цена всего продукта. Анализ такого вида позволяет определить целесообразность разработки веб-приложения.

В таблице 6.5 представлены результаты расчетов для основных показателей данной главы в краткой форме.

Таблица 6.4 – Результаты расчетов

Наименование показателя	Значение
Время разработки, ч.	725
Основная заработная плата, руб.	11 400
Дополнительная заработная плата, руб.	1 140
Отчисления в Фонд социальной защиты населения, руб.	4 263,6
Отчисления в БРУСП «Белгосстрах», руб.	75,24
Прочие прямые затраты, руб.	2 280
Накладные расходы, руб.	6 270
Себестоимость разработки программного средства, руб.	25 428,84
Расходы на реализацию, руб.	2 542,88
Полная себестоимость, руб.	27 971,72
Прибыль от реализации программного средства, руб.	5 594,34
Планируемая отпускная цена с НДС, руб.	40 279,27
Рентабельность разработки, %	16
Чистая прибыль, руб.	4 475,47

Проведённые экономические расчёты подтвердили целесообразность разработки программного средства, продемонстрировав его финансовую эффективность и перспективы. По итогам вычислений полная себестоимость проекта составила 27 971,72 руб. при сроке разработки 725 часов, что отражает оптимальное распределение ресурсов на создание приложения. Планируемая отпускная цена в размере 40 279,27 руб. с учётом НДС обеспечивает рентабельность на уровне 16% и чистую прибыль в размере 4 475,47 руб., что свидетельствует о финансовой устойчивости проекта и его способности окупить затраты в краткосрочной перспективе. Кроме того, внедрение приложения позволяет снизить издержки на маркетинг и поддержку пользователей за счёт автоматизации процессов, таких как уведомления и аналитика, что дополнительно повышает экономическую привлекательность проекта.

Необходимость разработки обусловлена потребностью пользователей в удобном инструменте для наблюдения за своим спортивным прогрессом, повышения спортивной мотивации и участия в соревновательном процессе. Приложение «Спортивный вызов» заменяет разрозненные инструменты, такие как устаревшие сайты, офлайн-продажи и ручной учёт, единой цифровой платформой, что позволяет существенно сократить операционные затраты и повысить эффективность управления спортивной активностью. Платформа предоставляет пользователям возможность централизованного доступа к данным, что упрощает контроль за тренировками и делает процесс более прозрачным и управляемым. Это особенно актуально для людей, которые стремятся к систематическому подходу в занятиях спортом, но сталкиваются с отсутствием современных решений в этой области.

Социальный эффект применения разработанного веб-приложения «Спортивный вызов» проявляется в нескольких ключевых аспектах.

Во-первых, приложение повышает доступность спортивного досуга и здорового образа жизни для широких слоёв населения за счёт удобного онлайн-доступа к персонализированным спортивным вызовам, статистике активности и планированию тренировок. Это особенно важно для жителей небольших городов, где доступ к цифровым спортивным сервисам и фитнес-программам ранее был ограничен, а также для начинающих спортсменов, которым необходимы простые и понятные инструменты для старта.

Во-вторых, решение способствует цифровизации сферы физической активности, формируя у пользователей привычку к самостоятельному планированию и отслеживанию спортивных целей через онлайн-платформу, что снижает зависимость от оффлайн-тренеров и повышает мотивацию. Это особенно актуально в постпандемийной реальности, где предпочтение отдаётся бесконтактным и дистанционным форматам взаимодействия, минимизирующим риски и обеспечивающим безопасность.

В-третьих, приложение стимулирует интерес к регулярным занятиям спортом за счёт интеграции современных механизмов вовлечения – личных кабинетов, системы уведомлений о новых вызовах, напоминаний о тренировках и персонализированных рекомендаций, адаптированных под индивидуальные потребности пользователя. Это повышает вовлечённость пользователей и делает занятия спортом более систематичными, увлекательными и социально ориентированными.

Кроме того, приложение способствует формированию культуры здорового образа жизни среди молодёжи. Реализация функций приложения позволяет «Спортивному вызову» стать универсальным решением, способным удовлетворить запросы пользователей с разным уровнем подготовки и интересами, а также укрепить социальные связи через спортивное взаимодействие.

Заключение

Целью дипломного проекта было разработать веб-приложение «Спортивный вызов». Цель достигнута полностью, соответствует заявленным требованиям. Веб-приложение обеспечивает удобный и функциональный интерфейс. В дипломном проекте:

1. Реализованы три ключевые роли: «Гость», «Пользователь» и «Администратор». Каждая роль обладает уникальным набором возможностей, адаптированным под их потребности. Это позволяет гибко взаимодействовать с веб-приложением в зависимости от уровня доступа.

2. Веб-приложение построено на основе клиент-серверной архитектуры. Серверная часть реализована с использованием фреймворка ASP.NET Core. Клиентская часть разработана на React.js. Это гарантирует динамичный и отзывчивый интерфейс. Для управления доступом к системе и генерации вызовов разработаны собственные сервисы. Аутентификация и авторизация реализованы через JWT, что обеспечивает высокий уровень безопасности и защиты данных.

3. Веб-приложение включает 25 ключевых функций. Они охватывают весь необходимый функционал: создание, обновление и удаление спортивных вызовов, управление дружескими запросами, просмотр персональной статистики с визуализацией прогресса, а также глобальной статистики для администраторов и пользователей, что делает приложение универсальным и полезным для всех категорий пользователей.

4. Для хранения данных была спроектирована и создана реляционная база данных на PostgreSQL. База данных включает 9 таблиц.

5. Общий объем программного кода веб-приложения составил более 8000 строк авторского кода.

6. Общее количество тестов – 25. Общее покрытие кода составляет 100%.

7. Количество маршрутов, реализованных в приложении – 25.

					ДП 00.00.ПЗ		
		ФИО	Подпись	Дата	Заключение		
Разраб.		Викторович И.С.					
Пров.		Гончар Е.А.					
Н. контр.		Гончар Е.А.					
Утв.		Смелов В.В.			БГТУ 1-40 01 01, 2025		
					Лит.	Лист	Листов
						1	1

Список используемых источников

- 1 ASP.NET [Электронный ресурс]. – Режим доступа: <https://dotnet.microsoft.com/apps/aspnet> – Дата доступа: 03.03.2025.
- 2 React [Электронный ресурс]. – Режим доступа: <https://react.dev/> – Дата доступа: 03.03.2025.
- 3 PostgreSQL [Электронный ресурс]. – Режим доступа: <https://www.postgresql.org/> – Дата доступа: 03.03.2025.
- 4 Приложение «Спортивный вызов» [Электронный ресурс]. – Режим доступа: <https://play.google.com/store/apps/details?id=ru.stayfitt.sportchallenge&hl=ru> – Дата доступа: 13.02.2025.
- 5 Приложение «Life+» [Электронный ресурс]. – Режим доступа: <https://play.google.com/store/apps/details?id=ru.stayfitt.lifeplus&hl=ru> – Дата доступа: 13.02.2025.
- 6 Приложение «SPOBI» [Электронный ресурс]. – Режим доступа: <https://apkpure.net/ru/spobi-sports-being-intelligent/com.spobi.android> – Дата доступа: 13.02.2025.
- 7 Nginx [Электронный ресурс]. – Режим доступа: <https://nginx.org/> – Дата доступа: 20.03.2025.
- 8 Docker Compose [Электронный ресурс]. – Режим доступа: <https://docs.docker.com/compose/> – Дата доступа: 20.03.2025.
- 9 Entity Framework Core [Электронный ресурс]. – Режим доступа: <https://learn.microsoft.com/en-us/ef/core/> – Дата доступа: 01.04.2025.
- 10 HTTPS [Электронный ресурс]. – Режим доступа: <https://www.myrasecurity.com/de/knowledge-hub/https/> – Дата доступа: 01.04.2025.
- 11 HTTP [Электронный ресурс]. – Режим доступа: <https://developer.mozilla.org/ru/docs/Web/HTTP/> – Дата доступа: 01.04.2025.
- 12 TCP [Электронный ресурс]. – Режим доступа: <https://www.computerweekly.com/de/definition/TCP-Transmission-Control-Protocol> – Дата доступа: 01.04.2025.
- 13 Npgsql (PostgreSQL для .NET) [Электронный ресурс]. – Режим доступа: <https://www.npgsql.org/> – Дата доступа: 04.04.2025.
- 14 Microsoft.AspNetCore.Authentication [Электронный ресурс]. – Режим доступа: <https://learn.microsoft.com/en-us/aspnet/core/security/authentication/?view=aspnetcore-9.0> – Дата доступа: 04.04.2025.

					ДП 00.00.ПЗ			
		ФИО	Подпись	Дата				
Разраб.	Викторович И.С.				Список использованных источников	Лит.	Лист	Листов
Пров.	Гончар Е.А.						1	2
						БГТУ 1-40 01 01, 2025		
Н. контр.	Гончар Е.А.							
Утв.	Смелов В.В.							

15 BCrypt.NET [Электронный ресурс]. – Режим доступа: <https://www.nuget.org/packages/BCrypt.Net-Next/> – Дата доступа: 03.01.2025.

16 React Router [Электронный ресурс]. – Режим доступа: <https://reactrouter.com/> – Дата доступа: 17.04.2025.

17 MUI/material [Электронный ресурс]. – Режим доступа: <https://mui.com/> – Дата доступа: 17.04.2025.

18 Axios [Электронный ресурс]. – Режим доступа: <https://axios-http.com/docs/intro> – Дата доступа: 19.04.2025.

19 React Toastify [Электронный ресурс]. – Режим доступа: <https://fkhadra.github.io/react-toastify/> – Дата доступа: 23.04.2025.

20 JWT (Json Web Tokens) [Электронный ресурс]. – Режим доступа: <https://jwt.io/> – Дата доступа: 23.04.2025.

21 Unit Of Work [Электронный ресурс]. – Режим доступа: https://en.wikipedia.org/wiki/Unit_of_work – Дата доступа: 23.04.2025.

Диаграмма вариантов использования ДП 01.00.ГЧ

Логическая схема базы данных ДП 02.00.ГЧ

Диаграмма развертывания ДП 03.00.ГЧ

Блок-схема алгоритма предложения вызова другу ДП 04.00.ГЧ

Диаграмма последовательности алгоритма добавления активности ДП 05.00.ГЧ

Скриншот работы приложения ДП 06.00.ГЧ

Приложение А

```

create table Users(
    user_id INT GENERATED ALWAYS AS IDENTITY PRIMARY KEY,
    username VARCHAR(30) NOT NULL,
    password_hash VARCHAR(300) NOT NULL,
    birthday DATE NOT NULL,
    user_role VARCHAR(15) CHECK (user_role IN ('User', 'Admin')) DEFAULT 'User'
);

create table Activities(
    activity_id INT GENERATED ALWAYS AS IDENTITY PRIMARY KEY,
    activity_name VARCHAR(35) NOT NULL,
    activity_type VARCHAR(35) NOT NULL
);

create table User_data(
    data_id INT GENERATED ALWAYS AS IDENTITY PRIMARY KEY,
    user_id INT REFERENCES Users(user_id),
    activity_id INT REFERENCES Activities(activity_id),
    date_info DATE DEFAULT CURRENT_DATE,
    weight NUMERIC,
    height numeric
);

CREATE TABLE Friendship (
    friend_id INT GENERATED ALWAYS AS IDENTITY PRIMARY KEY,
    user1_id INT REFERENCES Users(user_id),
    user2_id INT REFERENCES Users(user_id),
    friendship_date DATE DEFAULT CURRENT_DATE
);

create table Calls (
    call_id INT GENERATED ALWAYS AS IDENTITY PRIMARY KEY,
    call_name VARCHAR(30) NOT NULL,
    friend_id INT REFERENCES Friendship(friend_id),
    call_date DATE DEFAULT CURRENT_DATE,
    status VARCHAR(15) CHECK (status IN ('active', 'completed', 'pending'))
);

CREATE TABLE Challenge (
    ChallengeId SERIAL PRIMARY KEY,
    SenderId INT NOT NULL,
    ReceiverId INT NOT NULL,
    CallId INT NOT NULL,
    Status VARCHAR(50) NOT NULL,
    SentAt TIMESTAMP NOT NULL DEFAULT NOW(),
    RespondedAt TIMESTAMP NULL,
    FOREIGN KEY (SenderId) REFERENCES Users(UserId) ON DELETE CASCADE,

```

```

        FOREIGN KEY (ReceiverId) REFERENCES Users(UserId) ON DE-
LETE CASCADE,
        FOREIGN KEY (CallId) REFERENCES Calls(CallId) ON DELETE
CASCADE );
CREATE TABLE DailyActivity (
    DailyActivityId SERIAL PRIMARY KEY,
    StepQuantity INT NOT NULL, OtherActivityTime FLOAT NULL,
    UserId INT NOT NULL,
    Date TIMESTAMP NOT NULL DEFAULT NOW(),
    FOREIGN KEY (UserId) REFERENCES Users(UserId) ON DELETE
CASCADE );

create table Goals (
    GoalId INT GENERATED ALWAYS AS IDENTITY PRIMARY KEY,
    GoalName VARCHAR(30) NOT NULL,
    GoalDescription VARCHAR(100) NOT NULL,
    UserId INT REFERENCES Users(user_id),
    status VARCHAR(15) CHECK (status IN ('complited', 'uncom-
plited'))
);

create table StepsToGoal (
    StepId INT GENERATED ALWAYS AS IDENTITY PRIMARY KEY,
    SepName VARCHAR(30) NOT NULL,
    StepDescription VARCHAR(100) NOT NULL,
    GoalId INT REFERENCES Goals(GoalId),
    status VARCHAR(15) CHECK (status IN ('complited', 'uncom-
plited'))
);

```

Листинг – Скрипт создания базы данных

Приложение Б

```
[Authorize(Policy = "UserPolicy")]
[HttpPost]
public async Task<IActionResult> AddUserData([FromBody]
UserDataDto userDataDto) {
    if (userDataDto == null)
        return BadRequest("User data cannot be null.");

    var token = Request.Headers["Authorization"].ToString();
    var userNameClaim = User.Claims.FirstOrDefault();
    if (userNameClaim == null)
        { return Unauthorized("User is not authorized."); }

    string userName = userNameClaim.Value;
    var user = await _context.Users
        .FirstOrDefaultAsync(u => u.username == userName);
    if (user == null)
        return BadRequest($"User with name '{userName}' not
found.");

    var activity = await _context.Activities
        .FirstOrDefaultAsync(a => a.activity_name ==
userDataDto.activity_name);
    if (activity == null)
        return BadRequest($"Activity with name
'{userDataDto.activity_name}' not found.");

    var userData = new UserData {
        user_id = user.user_id,
        activity_id = activity.activity_id,
weight = userDataDto.weight,
        height = userDataDto.height
    };

    try {
        _context.UserData.Add(userData);
        await _context.SaveChangesAsync();
        return CreatedAtAction(nameof(GetUserData), new { id =
userData.data_id }, userData);
    }
    catch (Exception ex){
        return StatusCode(500, $"Internal server error:
{ex.Message}");
    }
}
```

Листинг 1 – Реализация метода AddUserData

```

[HttpGet("user")]
    public async Task<IActionResult> GetUserStats([FromQuery]
string username){
    var user = await _context.Users.FirstOrDefaultAsync(u =>
u.username == username);
    var now = DateTime.UtcNow;
    var currentYear = now.Year;
    var currentMonth = now.Month;

    var userCompletedCalls = await _context.Calls
        .Where(c => c.user_id == user.user_id && c.status ==
"completed").ToListAsync();

    int monthlyCompleted = userCompletedCalls.Count(c =>
        DateTime.TryParse(c.call_date, out DateTime callDate)
        && callDate.Year == currentYear && callDate.Month == currentMonth);

    int yearlyCompleted = userCompletedCalls.Count(c =>
        DateTime.TryParse(c.call_date, out DateTime callDate)
        && callDate.Year == currentYear);

    var categoriesStats = userCompletedCalls
        .Where(c => DateTime.TryParse(c.call_date, out
DateTime callDate) &&
            callDate.Year == currentYear)
        .GroupBy(c => c.call_name)
        .Select(g => new {
            category = g.Key,
            monthlyCompleted = g.Count(c => DateTime.Try-
Parse(c.call_date, out DateTime callDate) && callDate.Month == cur-
rentMonth), yearlyCompleted = g.Count() }).ToList();

    var result = new {
        username = user.username,
        monthlyCompleted,
        yearlyCompleted,
        categoriesStats
    };

    return Ok(result);
}
}

```

Листинг 2 – Создание активности

Приложение В

```

using Generator.Application.Services;
using Microsoft.IdentityModel.Tokens;
using System.IdentityModel.Tokens.Jwt;
using System.Text;

namespace Generator.API.Middleware;

public class JwtMiddleware
{
    private readonly RequestDelegate _next;
    private readonly TokenService _tokenService;

    public JwtMiddleware(RequestDelegate next, TokenService tokenService)
    {
        _next = next;
        _tokenService = tokenService;
    }

    public async Task InvokeAsync(HttpContext context)
    {
        var token = context.Request.Headers["Authorization"].ToString().Replace("Bearer ", string.Empty);

        if (!string.IsNullOrEmpty(token))
        {
            if (_tokenService.IsTokenBlacklisted(token))
            {
                context.Response.StatusCode = 401;
                await context.Response.WriteAsync("Token is blacklisted.");
                return;
            }

            try
            {
                var tokenHandler = new JwtSecurityTokenHandler();
                var key = Encoding.UTF8.GetBytes(_tokenService.GetKey());

                var validationParameters = new TokenValidationParameters
                {
                    ValidateIssuer = true,
                    ValidateAudience = true,
                    ValidIssuer = _tokenService.GetIssuer(),
                    ValidAudience = _tokenService.GetAudience(),

```

```

        IssuerSigningKey = new SymmetricSecurityKey(key),
        ClockSkew = TimeSpan.Zero
    };

    var principal = tokenHandler.ValidateToken(token, validationParameters, out var validatedToken);

    if (validatedToken is JwtSecurityToken jwtToken
        && jwtToken.Header.Alg.Equals(SecurityAlgorithms.HmacSha256,
            StringComparison.InvariantCultureIgnoreCase))
    {
        context.User = principal;
    }
    catch (Exception)
    {
        context.Response.StatusCode = 401;
        await context.Response.WriteAsync("Invalid token.");
        return;
    }

    await _next(context);
}

```

Листинг - Middleware для проверки токена

Приложение Г

```
import React, { createContext, useState, useEffect } from 'react';
import jwt_decode from 'jwt-decode';

export const AuthContext = createContext();

export const AuthProvider = ({ children }) => {
  const [authData, setAuthData] = useState(() => {
    const token = localStorage.getItem('token');
    console.log('Токен из localStorage:', token);
    if (token) {
      try {
        const decodedUser = jwt_decode(token);
        const now = Math.floor(Date.now() / 1000);
        console.log('Декодированный токен:', decodedUser);

        if (decodedUser.exp < now) {
          console.warn('Токен истёк.');
```

```
          localStorage.removeItem('token');
          localStorage.removeItem('currentUser');
          localStorage.removeItem('currentUserRole');
          return { token: null, user: null };
        }

        const role = decodedUser['http://schemas.microsoft.com/ws/2008/06/identity/claims/role'];
        if (!role) {
          console.error('Роль отсутствует в токене.');
```

```
          localStorage.removeItem('token');
          localStorage.removeItem('currentUser');
          localStorage.removeItem('currentUserRole');
          return { token: null, user: null };
        }

        localStorage.setItem('currentUser', decodedUser.sub);
        localStorage.setItem('currentUserRole', role);
        return { token, user: { ...decodedUser, role } };
      } catch (err) {
        console.error('Ошибка декодирования токена:', err);
        localStorage.removeItem('token');
        localStorage.removeItem('currentUser');
        localStorage.removeItem('currentUserRole');
        return { token: null, user: null };
      }
    }
    return { token: null, user: null };
  });

  useEffect(() => {
    if (authData.token) {
```

```

        console.log('Сохранение токена в localStorage.');
```

`localStorage.setItem('token', authData.token);
 } else {
 console.log('Удаление токена и данных пользователя из
localStorage.');`
`localStorage.removeItem('token');`
`localStorage.removeItem('currentUser');`
`localStorage.removeItem('currentUserRole');`
`}
}, [authData]);

const logout = () => {
 console.log('Выход пользователя.');`
`setAuthData({ token: null, user: null });
 localStorage.removeItem('token');`
`localStorage.removeItem('currentUser');`
`localStorage.removeItem('currentUserRole');`
`};

return (
 <AuthContext.Provider value={{ ...authData, setAuthData,
logout }}>
 {children}
 </AuthContext.Provider>
);
};`

Листинг – Реализация AuthContext и AuthProvider