TOMSK POLYTECHNIC UNIVERSITY ТОМСКИЙ ПОЛИТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ

Министерство науки и высшего образования Российской Федерации
федеральное государственное автономное
образовательное учреждение высшего образования
«Национальный исследовательский Томский политехнический университет» (ТПУ)

Школа – Инженерная школа информационных технологий и робототехники
Направление подготовки – 09.04.02 «Информационные системы и технологии»
Отделение школы (НОЦ) – Отделение информационных технологий

МАГИСТЕРСКАЯ ДИССЕРТАЦИЯ

Тема работы
Предметно ориентированное проектирование на примере разработки информационной системы для клуба управленческой борьбы

УДК 004.451:004.434:005

Студент

Группа	ФИО	Подпись	Дата
8ИМ02	Мухтар Д.Р.		

Руководитель

Должность	ФИО	Учёная степень, звание	Подпись	Дата
Доцент ОИТ	Мирошниченко Е.А.	к.т.н.		

КОНСУЛЬТАНТЫ ПО РАЗДЕЛАМ:

По разделу «Финансовый менеджмент»

Должность	ФИО	Учёная степень,звание	Подпись	Дата
Доцент ОСГН ШБИП	Былкова Т.В.	к.э.н.		

По разделу «Социальная ответственность»

Должность	ФИО	Учёная степень,звание	Подпись	Дата
Профессор ООД ШБИП	Федоренко О.Ю.	д.м.н.		

По разделу на английском языке

Должность	ФИО	Учёная степень,звание	Подпись	Дата
Доцент	Сидоренко Т.В.	к.п.н.		

ДОПУСТИТЬ К ЗАЩИТЕ:

Руководитель ООП	ФИО	Ученая степень,звание	Подпись	Дата
Доцент ОИТ	Савельев А.О.	к.т.н.		

Томск – 2022

ПЛАНИРУЕМЫЕ РЕЗУЛЬТАТЫ ОБУЧЕНИЯ

Код компетенции	Наименование компетенции
	Универсальные компетенции
УК(У)-1	Способен осуществлять критический анализ проблемных ситуаций на основе системного подхода, вырабатывать стратегию действий
УК(У)-2	Способен управлять проектом на всех этапах его жизненного цикла
УК(У)-3	Способен организовывать и руководить работой команды, вырабатывая командную стратегию для достижения поставленной цели
УК(У)-4	Способен применять современные коммуникативные технологии, в том числе на иностранном(-ых) языке(-ах), для академического и профессионального взаимодействия
УК(У)-5	Способен анализировать и учитывать разнообразие культур в процессе межкультурного взаимодействия
УК(У)-6	Способен определять и реализовывать приоритеты собственной деятельности и способы ее совершенствования на основе самооценки
	Общепрофессиональные компетенции
ОПК(У)-1	Способен самостоятельно приобретать, развивать и применять математические, естественно-научные, социально-экономические и профессиональные знания для решения нестандартных задач, в том числе в новой или незнакомой среде и в междисциплинарном контексте
ОПК(У)-2	Способен разрабатывать оригинальные алгоритмы и программные средства, в том числе с использованием современных интеллектуальных технологий, для решения профессиональных задач
ОПК(У)-3	Способен анализировать профессиональную информацию, выделять в ней главное, структурировать, оформлять и представлять в виде аналитических обзоров с обоснованными выводами и рекомендациями
ОПК(У)-4	Способен применять на практике новые научные принципы и методы исследований
ОПК(У)-5	Способен разрабатывать и модернизировать программное и аппаратное обеспечение информационных и автоматизированных систем
ОПК(У)-6	Способен использовать методы и средства системной инженерии в области получения, передачи, хранения, переработки и представления информации посредством информационных технологий
ОПК(У)-7	Способен разрабатывать и применять математические модели процессов и объектов при решении задач анализа и синтеза распределенных информационных систем и систем поддержки принятия решений
ОПК(У)-8	Способен осуществлять эффективное управление разработкой программных средств и проектов
	Профессиональные компетенции
ПК(У)-1	Способен управлять программно-техническими, технологическими и человеческими ресурсами
ПК(У)-2	Способен управлять развитием баз данных
ПК(У)-3	Способен управлять работами по сопровождению и проектами создания (модификации) информационных систем, автоматизирующих задачи организационного управления и бизнес-процессы.
ПК(У)-4	Способен проектировать и организовывать учебный процесс по образовательным программам с использованием современных образовательных технологий.
ПК(У)-5	Способен осуществлять руководство разработкой комплексных проектов на всех стадиях и этапах выполнения работ.

TOMSK POLYTECHNIC UNIVERSITY ТОМСКИЙ ПОЛИТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ

Министерство науки и высшего образования Российской Федерации
федеральное государственное автономное
образовательное учреждение высшего образования
«Национальный исследовательский Томский политехнический университет» (ТПУ)

Школа – Инженерная школа информационных технологий и робототехники
Направление подготовки – 09.04.02 «Информационные системы и технологии»
Отделение школы (НОЦ) – Отделение информационных технологий

УТВЕРЖДАЮ:
Руководитель ООП

_____ _____ _____
(Подпись) (Дата) (Ф.И.О.)

ЗАДАНИЕ
на выполнение выпускной квалификационной работы

В форме

Магистерской диссертации

(бакалаврской работы, дипломного проекта/работы, магистерской диссертации)

Студенту:

Группа	ФИО
8ИМ02	Мухтар Дархану Русланулы

Тема работы:

Предметно ориентированное проектирование на примере разработки информационной системы для клуба управленческой борьбы	
Утверждена приказом директора (дата, номер)	

Срок сдачи студентом выполненной работы:	

ТЕХНИЧЕСКОЕ ЗАДАНИЕ:

Исходные данные к работе	Предмет исследования: применение предметно-ориентированного проектирование при разработке информационной системы для Клуба управленческой борьбы

Перечень подлежащих исследованию, проектированию и разработке вопросов	– Аналитический обзор предметной области; – Разработка системы с применением предметно-ориентированного проектирования; – Исследование результатов применения предметно-ориентированного проектирования; – Социальная ответственность; – Финансовый менеджмент, ресурсоэффективность и ресурсосбережение.
Перечень графического материала	

Консультанты по разделам выпускной квалификационной работы

Раздел	Консультант
Основная часть	Доцент ОИТ ИШИТР, к.т.н., Мирошниченко Е.А.
Финансовый менеджмент, Ресурсоэффективность и ресурсосбережение	Доцент ОСГН ШБИП к.э.н., Былкова Т.В.
Социальная ответственность	Профессор ООД ШБИП д.м.н., Федоренко О.Ю
Английский язык	Доцент к.п.н, Сидоренко Т.В

Названия разделов, которые должны быть написаны на русском и иностранном языках:

Аналитический обзор предметной области
Разработка системы с применением предметно-ориентированного проектирования
Исследование результатов применения предметно-ориентированного проектирования
Обзор литературы

Дата выдачи задания на выполнение выпускной квалификационной работы по линейному графику	

Задание выдал руководитель / консультант (при наличии):

Должность	ФИО	Ученая степень, звание	Подпись	Дата
Доцент	Мирошниченко Е.А.	к.т.н.		

Задание принял к исполнению студент:

Группа	ФИО	Подпись	Дата
8ИМ02	Мухтар Д.Р.		

ЗАДАНИЕ ДЛЯ РАЗДЕЛА
«ФИНАНСОВЫЙ МЕНЕДЖМЕНТ, РЕСУРСОЭФФЕКТИВНОСТЬ И РЕСУРСОСБЕРЕЖЕНИЕ»

Студенту:

Группа	ФИО
8ИМ02	Мухтар Дархан Русланулы

Школа	Инженерная школа информационных технологий и робототехники	**Отделение школы (НОЦ)**	Информационных технологий
Уровень образования	Магистратура	**Направление/специальность**	09.04.02 «Информационные системы и технологии»

Исходные данные к разделу «Финансовый менеджмент, ресурсоэффективность и ресурсосбережение»:	
1. Стоимость ресурсов научного исследования (НИ): материально-технических, энергетических, финансовых, информационных и человеческих	Оплата труда (руководитель – 37700, инженер-исследователь – 23800).
2. Нормы и нормативы расходования ресурсов	Тариф на электроэнергию – 4,04 руб. за 1 кВт·ч.Районный коэффициент 30%; Коэффициент дополнительной заработной платы 12%; Накладные расходы 16%.
3. Используемая система налогообложения, ставки налогов, отчислений, дисконтирования и кредитования	Коэффициент отчислений во внебюджетные фонды – 1.3.
Перечень вопросов, подлежащих исследованию, проектированию и разработке:	
1. Оценка коммерческого и инновационного потенциала НТИ	Провести оценку готовности проекта к коммерциализации
2. Разработка устава научно-технического проекта	Определить: 1. Цели и результаты проекта 2. Организационную структуру проекта
3. Планирование процесса управления НТИ: структура и график проведения, бюджет, риски и организация закупок	Разработать: 1. План проекта 2. Бюджет научного исследования
4. Определение ресурсной, финансовой, экономической эффективности	Оценить сравнительную эффективность исследования
Перечень графического материала *(с точным указанием обязательных чертежей):*	
Карта сегментирования по разработке системы. Результаты оценки технических решений конкурентов. Интерактивная матрица проекта. SWOT-анализ. Форма готовности проекта к коммерциализации. Заинтересованные стороны проекта. Цели и результат проекта. Рабочая группа проекта. Перечень работ и исполнителей при разработке. Временные показатели проведенных работ. Календарный план-график проведения НИОКР по теме. Специальное оборудование для первого исполнения. Специальное оборудование для второго исполнения. Материальные затраты. Баланс времени. Расчет основной и дополнительной заработной платы. Отчисления во внебюджетные фонды. Величина затрат научно-исследовательской работы. Оценка характеристик вариантов исполнения проекта. Сравнительная эффективность разработки	

Дата выдачи задания для раздела по линейному графику	

Задание выдал консультант:

Должность	ФИО	Ученая степень, звание	Подпись	Дата
Доцент ОСГН ШБИП	Былкова Т. В.	к.э.н		

Задание принял к исполнению студент:

Группа	ФИО	Подпись	Дата
8ИМ02	Мухтар Д.Р.		

ЗАДАНИЕ ДЛЯ РАЗДЕЛА «СОЦИАЛЬНАЯ ОТВЕТСТВЕННОСТЬ»

Студенту:

Группа	ФИО		
8ИМ02	Мухтар Дархан Русланулы		
Школа		**Отделение (НОЦ)**	
Уровень образования	Магистратура	**Направление/ специальность**	09.04.02 «Информационные системы и технологии»

Тема ВКР:

Предметно ориентированное проектирование на примере разработки информационной системы для клуба управленческой борьбы	
Исходные данные к разделу «Социальная ответственность»:	
Введение – Характеристика объекта исследования (вещество, материал, прибор, алгоритм, методика) и области его применения. – Описание рабочей зоны (рабочего места) при разработке проектного решения/при эксплуатации	Объект исследования: разработка информационной системы с применением предметно-ориентированного проектирования. Область применения: проведения мероприятий для Клуба управленческой борьбы. Рабочее место: рабочее местооператора персонального компьютера в помещении офисного типа площадью 12 м^2, оснащённым системой отопления и кондиционирования воздуха, искусственными источниками света;
Перечень вопросов, подлежащих исследованию, проектированию и разработке:	
1. Правовые и организационные вопросы обеспечения безопасности <u>при разработке проектного решения:</u> – специальные (характерные при эксплуатации объекта исследования, проектируемой рабочей зоны) правовые нормы трудового законодательства; – организационные мероприятия при компоновке рабочей зоны.	– Трудовой кодекс Российской Федерации от 30.12.2001 N 197- ФЗ (ред. От 30.12.2015). – ГОСТ Р 50923-96. Дисплеи. Рабочее место оператора. Общие эргономические требования и требования к производственной среде. Методы измерения. – ТОИ Р-45-084-01 Типовая инструкция по охране труда при работе на персональном компьютере. – ГОСТ 12.2.032-78 Система стандартов безопасности труда (ССБТ). Рабочее место при выполнении работ сидя. Общие эргономические требования. – ГОСТ 12.0.003-2015 Система стандартов безопасности труда (ССБТ). Опасные и вредные производственные факторы – Шум. Общие требования безопасности, СН 2.2.4/2.1.8.562–96; – СанПиН 2.2.3670-20 «Санитарно- эпидемиологические требования к условиям труда» – СанПиН 2.2.1/2.1.1.1278-03 «Гигиенические требования к микроклимату производственных помещений» – СанПиН 1.2.3685-21 «Гигиенические нормативы и требования к обеспечению безопасности и (или) безвредности для человека факторов среды обитания» – СанПиН 2.2.2.542-96 «Гигиенические требо-

	вания к видео-дисплейным терминалам, персональным электронно-вычислительным машинам и организации работ» – СН 2.2.4/2.1.8.562-96. Шум на рабочих местах, в помещениях жилых, общественных зданий и на территории жилой застройки; – СанПиН 1.2.3685-21 «Гигиенические нормативы и требования к обеспечению безопасности и (или) безвредности для человека факторов среды обитания» – ГОСТ 12.1.019-2017 «Электробезопасность. Общие требования и номенклатура видов защиты» – ГОСТ Р 53692-2009. Ресурсосбережение. Обращение с отходами. Этапы технологического цикла отходов. – М.: ИПК Издательство стандартов, 2001 год. ГОСТ 12.1.004-91. ССБТ Пожарная безопасность. Общие требования. – М.: ИПК Издательство стандартов, 2001.
2. Производственная безопасность <u>при разработке проектного решения:</u> – Анализ выявленных вредных и опасных производственных факторов – Расчет уровня опасного или вредного производственного фактора	Вредные факторы: – отклонение показателей микроклимата; – превышение уровня шума на рабочем месте; – недостаточная освещенность рабочей зоны; – психофизиологические факторы (монотонность труда, нервно-психические перегрузки, перенапряжение зрительных анализаторов). Опасные факторы: – нарушение предельно допустимых значений напряжений прикосновения и токов; – короткое замыкание; – статическое электричество.
3. Экологическая безопасность <u>при разработке проектного решения</u>	Негативное влияние на селитебную зону, атмосферу, гидросферу не выявлено. Негативное влияние на литосферу происходит при утилизации компьютера и периферийных устройств (принтеры, МФУ, веб-камеры, наушники, колонки, телефоны); люминесцентных ламп; макулатуры.
4. Безопасность в чрезвычайных ситуациях <u>при разработке проектного решения</u>	Возможные ЧС: ураганы, ливни, оползни, пожары. Наиболее типичная ЧС: пожар.
Дата выдачи задания для раздела по линейному графику	

Задание выдал консультант:

Должность	ФИО	Ученая степень, звание	Подпись	Дата
Профессор, ООД ШБИП	Федоренко Ольга Юрьевна	Доктор медицинских наук		

Задание принял к исполнению студент:

Группа	ФИО	Подпись	Дата
8ИМ02	Мухтар Дархан Русланулы		

РЕФЕРАТ

Выпускная квалификационная работа содержит 107 страниц, 31 рисунков, 29 таблиц, 24 источников, 2 приложения.

Ключевые слова: клиентская часть веб-сайтов, серверная часть веб-сайтов, C#, .NET, ORM, DDD, CQRS, HTML, SPA, JavaScript, Vue, SQL, HTTP, CSS, JSON, API.

Объектом исследования данной работы является предметно-ориентированное проектирование.

Целью работы является применение предметно-ориентированного проектирования при разработке программного обеспечения для Клуба управленческой борьбы. В ходе исследования были проанализированы существующие актуальные инструменты и технологии применяемый в разработки информационных веб-систем как для клиентской, так и для серверной части. Также были сформированы функциональные требования к системе совместно с Заказчиком.

В результате работы была спроектирована и разработана система для Клуба управленческой борьбы с применением предметно-ориентированного проектирования. Преимуществами применения DDD является структурированная и понятная архитектура кодовой базы, а также расширяемость системы.

Область применения: система предназначена для регистрации тренингов и их участников для Клуба управленческой борьбы и администрирования данными системы.

Экономическая эффективность работы обусловлена малым количеством систем, удовлетворяющих требования Заказчика.

СПИСОК ТЕРМИНОВ, СОКРАЩЕНИЙ И УСЛОВНЫХ ОБОЗНАЧЕНИЙ

C# — мульти-парадигменный язык программирования Microsoft.

.NET — это технология, которая поддерживает создание и выполнение веб-служб и приложений Windows.

ORM (англ. Object relational mapping) — технология программирования, которая связывает базы данных с концепциями объектно-ориентированных языков программирования

DDD (англ. Domain driven development) —набор принципов направленных на оптимальных систем объектов.

CQRS (англ. Command and Query Responsibility Segregation) — архитектурны стиль, предписывающий разделение чтение и запись данный в программном коде.

HTML (англ. HyperText Markup Language) — язык гипертекстовой разметки страниц в браузере.

SPA (англ. Single-Page application) — веб-сайт использующий один HTML-документ в качестве оболочки для веб-страниц.

JavaScript — мульти-парадигменный язык программирования, широко применяемый в веб-разработке.

Vue — фреймворк для разработки веб-клиентов на JavaScript.

SQL (англ. Structured query language) — структурированный язык запросов для работы с реляционными базами данных.

HTTP (англ. HyperText Transfer Protocol) — протокол прикладного уровня передачи данных.

CSS (англ. Cascading Style Sheets) — каскадные таблицы стилей, применяемые для описания внешнего вида HTML страниц.

JSON (англ. JavaScript Object Notation)— формат обмена данными, основанный на JavaScript.

API (англ. Application programming interface) — протокол для взаимодействия между программами.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Информационная система представляет из себя среду, состоящей из людей, компьютеров, компьютерных сетей, программных продуктов и баз данных, а также различного рода технологических и программные средств. Реализация функций информационной системы требует изучения и понимания ориентированной на нее информационной технологии. [1].

Ключевыми функциями информационной системы является хранение, сбор, обработка, поиск и выдача данных, необходимых в процессе решений задач в любой области [2].

В настоящее время имеется большое количество технологий, позволяющих создавать информационные системы. Так как информационные системы имеют разные задачи, инструменты, применяемые при их создании, могут отличаться. Поэтому необходимо применять технологии при создании информационной системы с учетом поставленных к ней задач.

Актуальность рассмотрения темы научно-исследовательской работы магистранта, состоит в том, что у общественной организации «Федерация управленческой борьбы» отсутствует программный продукт, который позволит регистрировать турниры и их участников для проведения соревнований по управленческой борьбе. Применение предметно-ориентированного проектирования в разработке позволит построить программные абстракции, которые основываются на бизнес-логике и не зависят от реализаций сторонних библиотек, что приведет к лучшей расширяемости системы.

Соревнования по управленческой борьбе проводятся на регулярно. В их основу заложена авторская технология В. К. Тарасова «Управленческий поединок», которая является интеллектуальной собственностью Таллиннской школы менеджеров (ТШМ) [3].

Целью научно-исследовательской работы магистранта является проектирование программного обеспечения для проведения соревнований по управленческой борьбе с применением предметно-ориентированного проектирования.

Из цели научно-исследовательской работы магистранта можно выделить следующие задачи:

- проанализировать принципы предметно-ориентированного проектирования;
- спроектировать архитектуру программного обеспечения для хранения и управления данными необходимыми для проведения соревнований по управленческой борьбе;
- проанализировать актуальные инструменты, применяемые при разработке программного обеспечения и подобрать подходящие средства для реализации программного комплекса согласно тематике темы НИР магистранта;
- реализовать комплекс программного обеспечения для процесса проведения соревнования по управленческой борьбе применяя предметно-ориентированное проектирование;
- проанализировать качество кодовой базы.

1. Анализ предметной области исследования

1.1 Описание предметной области Заказчика

Управленческие поединки представляет из себя социальную технологию, целью которой является развить навыки ведения переговоров у руководителей и персонала. Суть поединка заключается в моделировании какой-либо управленческой ситуации, которая может иметь конфликтный характер, и поместить участников позволив им назначить друг-другу соответствующие роли. Особенностью таких ситуаций является отсутствие однозначного верного пути разрешения конфликта. Перед началом поединка оглашается описание ситуации и цели действующих в ней лиц. Задачей каждого из участников является достижение этих целей наиболее корректным путём.

Турниры по управленческой борьбе подразделяются на: практикум, товарищеский, именной, отборочный на финал года и финал года. Турнир может иметь следующие варианты сеток: по командам, навылет, каждый с каждым и комбинированный. Поединки в турнирах подразделяются на классические управленческие поединки, в которых участвуют девять судей, и экспресс-поединки с пятью судьями. Турнир может иметь статус официального. Тогда его результаты учитываются в определении квалификации игроков и судей. Все участники, регистрируемые в турнирах, подразделяются на зрителей, игроков, секундантов, судей, арбитров, секретарей и тренеров. Кроме того, каждый участник турнира может иметь принадлежность к определённой школе по управленческой борьбе [4].

Заказчиком системы является Томское региональное отделение Федерации управленческой борьбы. В ходе анализа предметной области и встреч Заказчиком было определено, что Заказчику необходима система, для регистрации турниров и участников в них, ведения учета данных участников и администрирования всех предстоящих турниров. Заказчик также рассматривает возможность расширения системы и добавление в нее нового функционала в будущем, например, возможность регистрироваться пользователям самостоятельно. Поэтому было важно спроектировать систему таким образом, чтобы при

смене разработчика система могла быть расширена в кратчайшие сроки и была поддерживаемой.

1.2 Основы предметно-ориентированного проектирования

Предметно-ориентированное проектирование (Domain-driven design, DDD) — это подход в проектировании программного обеспечения, используемый для проецирования бизнес-моделей и бизнес-логики в программных абстракциях, называемыми моделями предметных областей. Данный подход позволяет сфокусироваться на требованиях предметной области и оптимально реализовать бизнес-логику в программном коде [5].

Часто разработчик не знаком с предметной областью Заказчика, что может вызывать сложности при разработки программного продукта. Ключевой задачей Domain-Driven Design является упрощение сложных бизнес-процессов, а также их автоматизация и реализация в программном коде. «Domain» переводится с английского как «предметная область», и именно на предметной области основывается разработка ПО и проектирование в рамках данного подхода.

В традиционном жизненном цикле разработки программного обеспечения детали предметной области Заказчика «переводятся» аналитиком в удобную для инженера форму, в виде технического задания. При передаче требовании системы программистам в таком виде, возможна некоторая потеря информации необходимой для полного понимания бизнес-процессов. Для того чтобы облегчить коммуникацию разработчиков и экспертов предметной области, и упростить обмен полезными знаниями о предметной области, подход DDD предлагает использовать общий набор терминов, понятий и фраз в общении между членами команды. Этот набор позже отразится в исходном коде разработанной программы [5].

Бизнес-логика является важнейшей частью любого программного обеспечения, однако существуют и другие не менее важные модули информационной системы. Чтобы реализовать функциональные и нефункциональные требования, в кодовой базе должны присутствовать и другие блоки выполняющие задачи связанные с инфраструктурой системы. Они должны взаимодействовать

с пользователями для сбора данных и предоставления выходных данных, использовать различные механизмы хранения для сохранения состояния объектов и интеграции с внешними системами.

В следствие большого количества задач, с которые должна решать кодовая база, существует риск рассеивания кода, связанного с бизнес-логикой между различными компонентами. Например, дублирование некоторой логики в модуле, связанном с пользовательским интерфейсом и модуле работы с данными. Отсутствие строгой организации кодовой базы влекут к проблемам с реализацией нового функционала и затрудняет изменение имеющегося кода. В случаях, когда бизнес-логика должна измениться или дополниться, могут возникнуть трудности с определением частей кода подлежащих изменениям. При внесении изменении в неструктурированную кодовую базу также есть вероятность появления логических багов, в, казалось бы, несвязанных частях системы [6].

Применение архитектурных шаблонов структурирует кодовую базу и устанавливает четкие границы между ними попутно определяя правила взаимодействия между компонентами.

Правильный выбор подходящего способа организации кодовой базы имеет решающее значение для корректной реализации бизнеса-логики в краткосрочной перспективе и облегчения поддержки системы в долгосрочной перспективе.

В ходе исследования система была написана в традиционной трёхслойной архитектуре и в чистой архитектуре

1.3 Применяемые архитектурные шаблоны в разработке

1.3.1 Обзор чистой архитектуры и ее преимуществ

Многоуровневая архитектура является одной из наиболее распространенных архитектурных шаблонов [7]. Такая архитектура организует кодовую базу в горизонтальные слои, где каждый слой отвечает за взаимодействие с пользователями, реализацию бизнес-логики и хранение данных соответственно.

В своей классической форме многоуровневая архитектура состоит из трех слоев:

- слой представления;
- слой бизнес-логики;
- слой доступа к данным.

Изначально слой представления, реализовывал интерфейс пользователя программы, например такой как веб-интерфейс или десктопное приложение. Однако в современных системах уровень представления имеет более широкий охват и отвечает за любые виды взаимодействии пользователя с программой как в синхронном, так и в асинхронном ключе, например посредством:

- Графического пользовательского интерфейса (GUI);
- Интерфейса командной строки (CLI);
- API для программной интеграции с другими системами;
- Подписки на события в брокере сообщений.

Вышеперечисленные методы взаимодействия позволяют системе получать запросы от внешней среды и возвращать результаты запроса. Строго говоря, уровень представления — это общедоступный интерфейс программы.

После слоя представления следует слой бизнес-логики, отвечающий за реализацию и инкапсуляцию бизнес-логики программы [5]. На этом уровне реализуется вся логика, связанная с бизнес-процессами.

Последним уровнем является уровень доступа к данным, который обеспечивает доступ к данным и реализует все операции, связанные с хранением данных в базе данных, файлах и других источниках. Этот слой

также может отвечать за интеграцию с различными внешними ресурсами, необходимыми для реализации функционала программы, например API, предоставляемые внешними системами, или облачные сервисы.

Уровни построены в коммуникационную модель, с зависимостью, идущей «сверху вниз», то есть каждый слой может иметь зависимости только от слоя, находящегося непосредственно под ним, как показано на рисунке 1.

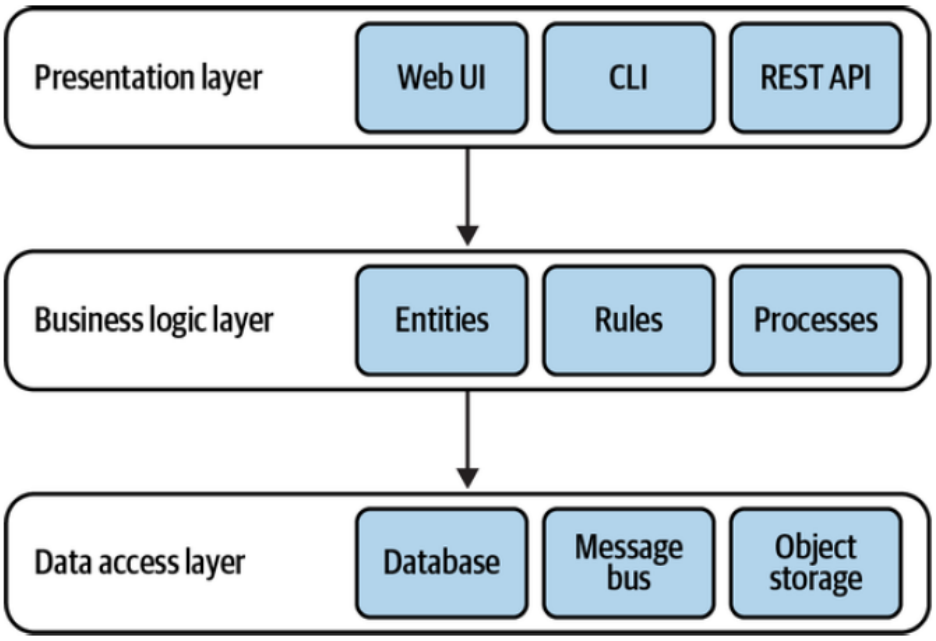

Рисунок 1. Классическая трехуровневая архитектура

Зависимость между бизнес-логикой и уровнями доступа к данным делает этот архитектурный шаблон подходящим для систем в которых бизнес-процессы реализуются в виде транзакции или в виде шаблона активной записи.

Однако такой подход не подходит для DDD так как в DDD бизнес логика не должна зависеть от технических деталей. Поэтому для реализации серверной системы была выбрана иная архитектура, а именно чистая архитектура. Чистая архитектура предписывает разделение кода, связанного с бизнес-логикой (абстракцией более высокого уровня) от изменчивых технических деталей (детали более низкого уровня), определяя при этом четкие границы. Основным принципом такой архитектуры является принцип инверсии зависимостей, согласно которому, модули, существующие в программном коде, должны зависеть от абстракции более высокого уровня. Пример классической чистой архитектуры приведен в рисунке 2.

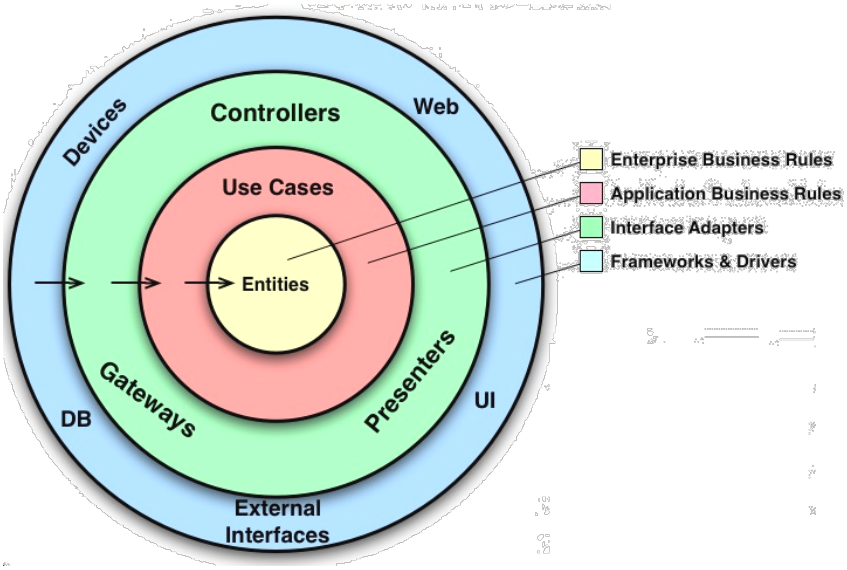

Рисунок 2. Диаграмма классической чистой архитектуры

1.3.2 Паттерн «Репозиторий»

Репозитории — это классы или компоненты, которые инкапсулируют логику, необходимую для доступа к источникам данных. Они централизуют общие функции доступа к данным, обеспечивая лучшую сопровождаемость и отделяя инфраструктурный слой, используемый для доступа к базе данных, от слоя моделей предметной области [8].

Для каждого агрегата создается один экземпляр класса «репозиторий». В системе, построенной на шаблонах Domain-Driven Design (DDD), единственным объектом, через который происходят операции по работе с данными в базе данных, должен быть репозитории. Это обусловлено тем, что репозиторий имеет отношение один к одному с корневым агрегатом, который управляет инвариантами и отвечает за согласованность данных и транзакций. Запрашивать данные из базы данных можно и другими путями, например следуя подходу CQRS, так как запросы все равно не изменяют состояние данных в базы. Однако все изменения данных и транзакций всегда должны контролироваться репозиториями и корневыми агрегатами.

Репозиторий предоставляет возможность позволяет вам заполнять оперативную память данными, которые поступают из базы данных в виде сущностей предметной области. Как только объекты попадают в память, их можно изменить, а затем вернуть обратно в базу данных посредством транзакций.

При использовании архитектурного шаблона CQS/CQRS, запросы, связанные с чтением данных, выполняются с помощью прямых SQL запросов посредством библиотек ADO.NET или Dapper [9]. Этот подход гораздо более гибкий, чем репозитории, потому что можно запрашивать и присоединять любые нужные таблицы, без ограничений накладываемыми агрегатами. Эти данные напрямую поступают в уровень представления или в клиентское приложение.

Для проведения транзакции используется паттерн Unit Of Work который синхронизирует операции с базой данных среди различных репозиториев. Благодаря этому паттерну несколько операций добавления, обновления или удаления данных объединяются одной транзакцией. Проще говоря, это означает, что для конкретного действия пользователя, такого как регистрация на веб-сайте, все операции работы с данными обрабатываются в одной транзакции.

1.3.3 Обзор CQRS и Event Sourcing

При разработке архитектуры серверных систем часто используется одна и та же модель данных для запроса и обновления данных. Такой подход является распространенным и хорошо работает при разработке простых систем. Однако в более сложных приложениях этот подход может стать громоздким так как при чтении данных может использоваться одна модель со всеми необходимыми полями, которые будут отображаться в интерфейсе, либо несколько моделей для определенных случаев. Отображение объектов может стать сложным. Кроме того, необходимо помнить, что на стороне записи модель может реализовывать сложную проверку и бизнес-логику. В результате вы можете получить слишком сложную модель, которая делает слишком много. В добавок к этим трудностям возникает проблема скорости выполнения сложных запросов при применении ORM библиотек. Для решения данной проблемы были проанализированы различные шаблоны проектирования, такие как MVC, MVVM и MVP. Однако данные шаблоны применяются для разделения логики приложения от ее представления и не решают задачу более гибкой работы с данными. Данную задачу решает шаблон CQRS. Помимо данной проблемы, в системе необходимо

вести учет всех манипуляций с данными для ведения аудита. Для этого был применен шаблон Event Sourcing.

CQRS — это архитектурный шаблон, который разделяет модели для чтения и записи данных [10]. Базовая идея состоит в том, что операции системы делятся на две четкие категории:

– Запросы. Возвращают результат или данные не изменяя состояние системы.

– Команды. Изменяют состояние системы.

Согласно CQS, что методы в одном и том же объекте должны быть либо запросами, либо командами. Каждый метод возвращает состояние или изменяет состояние, но не делает и то, и другое одновременно. Объект шаблона даже одного репозитория может соответствовать CQS.

Принцип разделения ответственности команд и запросов (CQRS) был введен Грегом Янгом и активно распространяется Уди Даханом и другими [9]. Он основан на принципе CQS, но более детален. Его можно рассматривать как шаблон, основанный на командах и событиях с возможностью использовать асинхронные сообщения. Во многих случаях CQRS относится к более сложным сценариям, например, использование отдельных баз данных для операций чтения (запросов) и для операций записи (обновлений). Кроме того, в более сложной системе CQRS можно реализовать использование событий в качестве источников (Event Sourcing) и хранить в модели предметной области только события вместо данных о текущем состоянии [11].

1.4 Обзор применяемых инструментов в разработке

1.4.1 Средства работы с базой данных

Для хранения данных об участниках и предстоящих турнирах Клуба управленческой борьбы была выбрана СУБД компании Майкрософт MS SQL Server. СУБД MS SQL Server представляет собой набор объектов, позволяющих хранить данные и управлять ими. Теоретически SQL Сервер поддерживает 32 767 баз данных на один экземпляр, хотя обычно используется только несколько баз данных. Количество баз данных, которые может обрабатывать SQL Server, зависит от нагрузки и аппаратного обеспечения [12].

При создании базы данных был применена технология программирования, которая позволяет преобразовывать несовместимые типы моделей в ООП, Entity Framework Core. Данная ORM, была выпущена Microsoft в 2016 году и является кросс-платформенной так как может работать на Windows, Linux и MacOS [13].

Entity Framework Core связывает элементы реляционной базы данных с сущностями объектно-ориентированных языков программирования, в данном случае языка C#. Например, столбцы таблицы в базе данных представляются как поля класса в языке C#. В таблице 1 приведены подобные соответствия.

Таблица 1. Соответствия реляционной базы данных и EFCore

Реляционная база данных	**Язык C#**
Таблица	Классы
Столбцы таблицы	Поля класса
Запись	Объект в коллекции
SQL запросы	Запросы LINQ

1.4.2. Средства разработки серверной части

Для реализации серверной части была выбран серверный веб фреймворк от Майкрософт, ASP.NET Core. Программные среды, которые упрощающие создание, поддержку и масштабирование веб-приложений являются серверными веб-фреймворками. Они предоставляют библиотеки и инструменты, упрощаю-

щие общие задачи веб-разработки, включая маршрутизацию URL-адресов для соответствующих обработчиков, поддержку сеансов и авторизацию пользователей, взаимодействие с базами данных, форматирование вывода (например, HTML, JSON, XML) и улучшение защиты от веб-атак [14].

ASP.NET Core — это один из популярной веб-фреймворков. ASP.NET Core был выпущен в июне 2016 г. В последних версиях ASP.NET появилось множество дополнительных обновлений, в которых было уделено особое внимание высокой производительности фреймворка и удобству разработки. Помимо этого, была проделана определенная работа над обратной совместимостью.

ASP.NET Core построен на .NET Core, который представляет собой кроссплатформенную версию .NET Framework без специфических для Windows интерфейсов (API) [15]. Windows по-прежнему является доминирующей операционной системой, однако большинство веб-приложении размещается в небольших контейнерах на облачных платформах. Поэтому Microsoft расширила возможности .NET Framework и сделала возможным развертывать приложения, написанные с применение ASP.NET Core в более широком наборе хостинговых сред и разработчики могут создавать веб-приложения ASP.NET Core для Linux и MacOS. В целом, ASP.NET Core имеет следующие преимущества:

– кросс-платформенность;
– модульную архитектура упрощающая поддержку систем;
– открытый исходный код;
– возможность развертывания в облачных средах

Для серверной части было написано API, веб-служба благодаря которой клиент может сохранять, получать, удалять или изменять данные в базе данных по HTTP протоколу. HTTP — это веб-протокол передачи данных, позволяющий получать различные ресурсы, например HTML-документы. Данный протокол является основой обмена данными в Интернете. HTTP является протоколом клиент-серверного взаимодействия. Это значит что получатель сам инициирует запросы к серверу, обычно посредством веб-браузера (web-browser). Данные передаются от клиента серверу и наоборот в формате JSON или XML [14].

1.4.3. Средства разработки клиентской части

Для разработки клиента был необходим веб фреймворк, позволяющий в кратчайшие сроки разработать прототип и в тоже время будет достаточно простым для будущего расширения. В качестве такого фреймворка был выбран Vue.js. Vue.js это фреймворк написанный на JavaScript позволяющий создавать производительные одностраничные веб-сайты. Данный фреймворк имеет более низкий порог вхождения по сравнению с такими фреймворками как React или Angular и имеет подробную документацию с множеством примеров и вариантов использования. Vue.js приложения можно создавать с помощью Vue CLI или включать его на свою страницу через тег скрипта, что упрощает его использование в проектах без систем сборки [16].

Подобно другим современным веб-фреймворка, Vue.js позволяет создавать автономные и управляемые компоненты, в которых можно написать собственную логику представления данных. Это позволяет масштабировать систему по мере ее увеличения, потому что любой изменения могут быть инкапсулированы отдельно друг от друга.

При масштабировании веб-приложения управление состоянием данных в переменных является одной из основных проблем. Vue.js решает эту проблему с помощью библиотеки Vuex.

Vue.js позволяет внедрять интерактивное поведение и дополнительные возможности в любой контекст, в котором выполняется JavaScript. Vue.js можно использовать как на отдельных страницах, решая простые задачи, так и в качестве фундамента для полноценных промышленных приложений. [17].

Когда JavaScript-фреймворки начали поддерживать асинхронные методы программирования, исчезла необходимость в обновлении всей веб-страницы при каждом запросе. Частичное обновление представления позволяет сайтам и приложениям быстрее реагировать на изменения, но в определенной степени приводит к дублированию кода.

Данная библиотека позволяет создавать реактивные приложения, то есть приложения выполняющие следующие функции:

– отслеживать изменения в переменных
– уведомлять свои компоненты при изменениях
– отображать представления в ответ на возникшие изменения
– своевременно реагировать на действия пользователя

Помимо Vue.js, была использована дизайн-система Ant Design.

2. Проектирование и программная реализация информационной системы

2.1 Проектирование серверной части системы

2.1.1 Проектирование схемы базы данных

База данных состоит из справочных и основных (заполняемые данными об участниках и турнирах) таблиц: справочные таблицы включают в себя таблицы “Роли”, “Формат турнира”, “Тип турнира”, “Сетка турнира”, остальные же таблицы являются основными. Ниже приведена схемы базы данных.

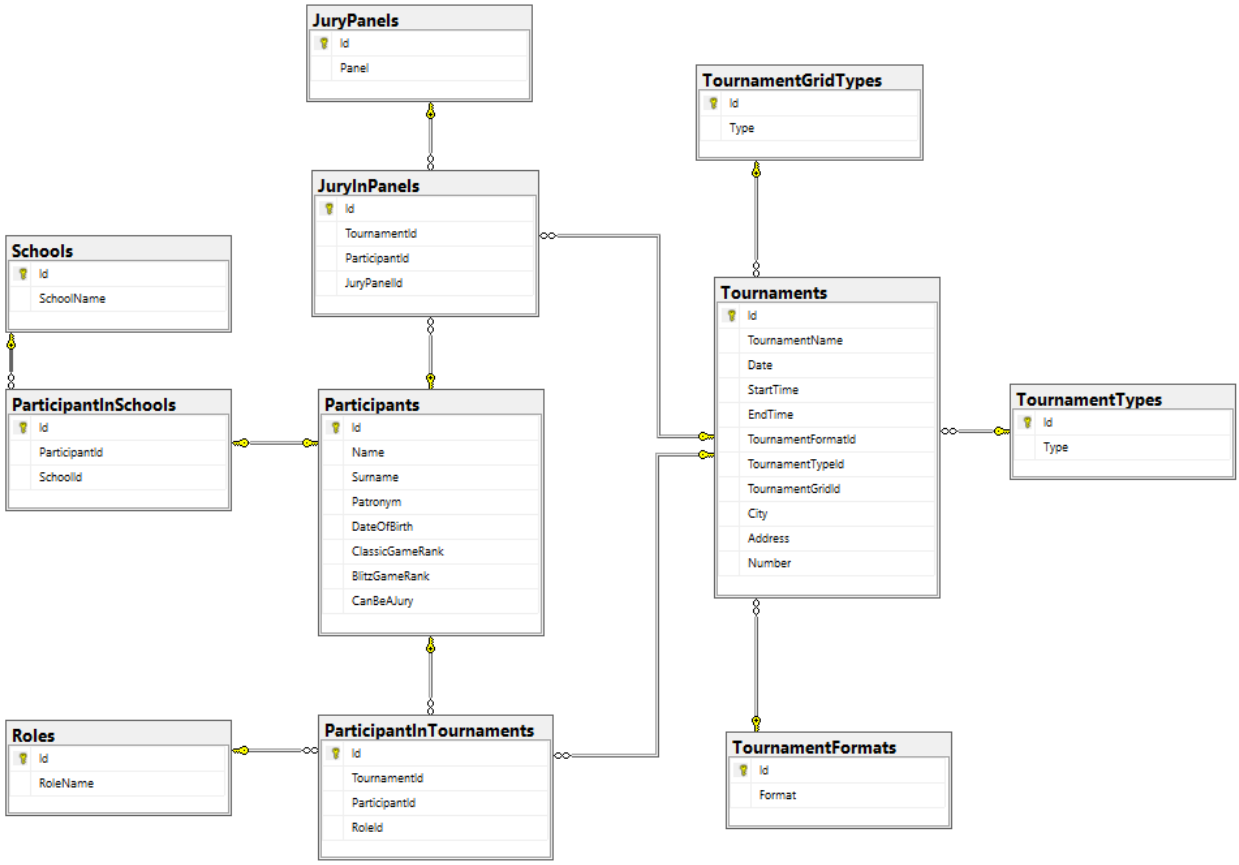

Рисунок 3. Схема базы данных

Схема базы данных подразделена на два агрегатных кластера [5]. Согласно DDD, агрегаты, это кластер объектов предметной области, который можно рассматривать как единое целое. Задачей агрегатов является сохранение согласованности данных путем обозначения границ, в рамках которых происходит модификация данных во избежание возникновений нарушении в бизнес логике. В случае спроектированной схемы данных, мы имеем два основных кластера агрегатов:

– кластер таблиц “Участник в школах”, “Школы”, “Участник в турнире”, “Роли”, “Судьи”, “Судейские коллегии”, “Судьи в коллегиях”;

– кластер таблиц “Турниры”, “Типы сеток турнира”, “Типы турнира”, “Форматы турнира”.

Агрегатный кластер должен иметь корневой объект (aggregate root), через который происходит доступ ко всем остальным объектам кластера и выполняются команды изменения данных [7]. Любая работа с объектами агрегатного кластера должна производиться только через корневой агрегат. Таким образом, корень может обеспечить согласованность данных своих агрегатов. Таким образом агрегат является основным элементом переноса данных, а транзакции не должны пересекать границы другого кластера агрегатов. Ниже приведены корневые объекты двух агрегатных кластеров. Данные объекты будут содержать в себе методы, связанные с бизнес-логикой.

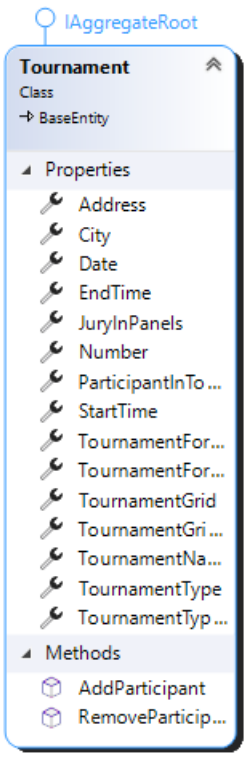

Рисунок 4. Основные корневые объекты

2.1.2 Построение системы согласно чистой архитектуре

В качестве основной архитектуры приложения была выбрана чистая архитектура. Исходный код состоит веб проекта Web, который зависит от трех библиотек, SharedKernel, Infrastructure и Core. Библиотека Infrastructure отвечает за реализацию работы с базой данных и зависит от библиотеки Core, так как в этой библиотеке хранятся классы доменных сущностей и их связи согласно которым будет создаваться таблицы в базе данных. Библиотека SharedKernel включает в объекты передачи данных (DTO) и другие модели необходимые для всех слоев. Для наглядности приведена следующая иллюстрация.

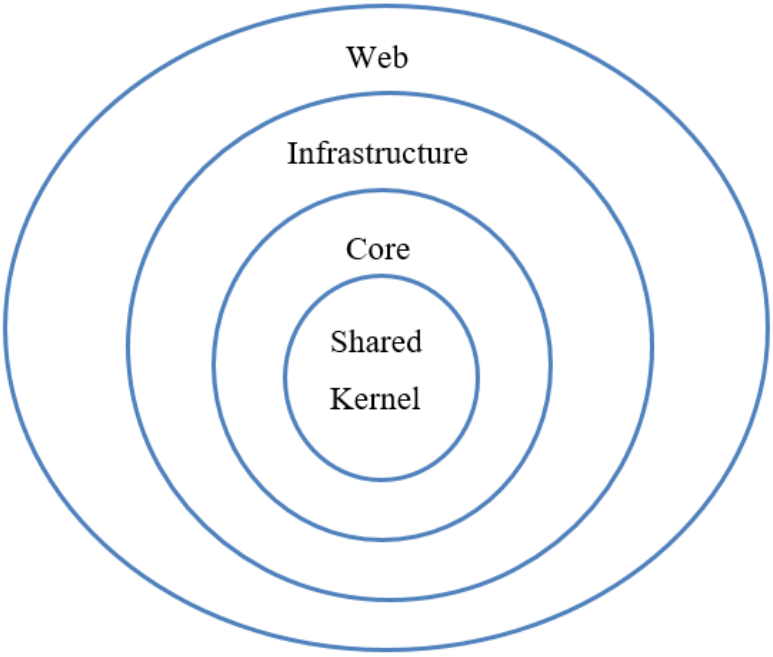

Рисунок 5. Архитектура ПО

Библиотека Core содержит классы, в которых реализуется бизнес логика системы. Например, для регистрации участника в турнире, необходимые данные запрашиваются на уровне абстракции Infrastructure, а затем передаются на нижний уровень где происходит валидация и другие операции, связанные с бизнес-логикой. Таким образом код реализующий бизнес логику не зависит от используемой библиотеки ORM и других технических деталей.

Для реализации данной архитектуры в кодовой базе были созданы стандартные библиотеки классов на языке C# для каждого слоя и были добавлены зависимости согласно принципу самой архитектуры, то есть проект Web зави-

сит от библиотеки Infrastructue и Core, а библиотека Infrastructure в свою очередь зависит от библиотеки Core. При это все библиотеки имеют доступ к библиотеке SharedKernel. Ниже приведено дерево проекта в интерфейсе среды разработки [18].

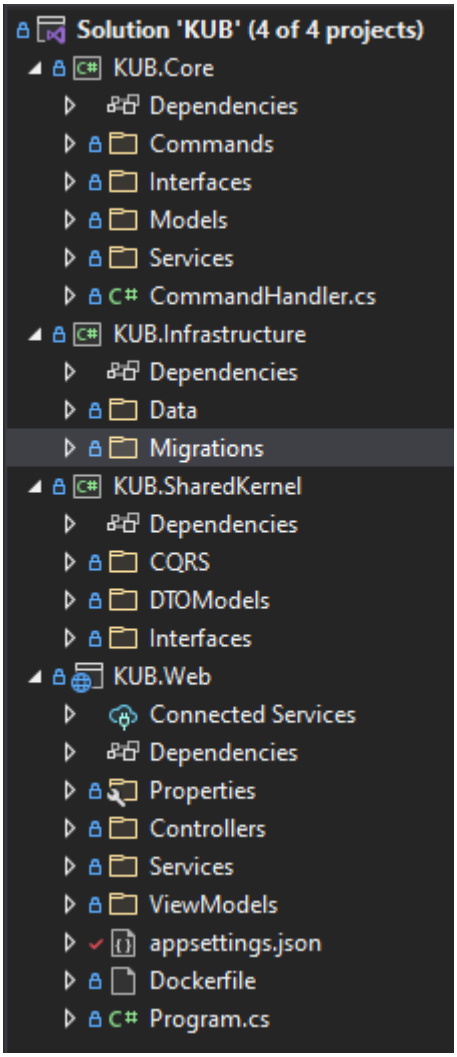

Рисунок 6. Дерево проекта в IDE

2.1.3 Реализация CQRS

Для реализации данного шаблона в первую очередь были написаны интерфейсы репозиториев и сервисов в библиотеке Core. Классы реализующие интерфейсы репозиториев были добавлены в библиотеку Infrastructure так как они не являются абстрактными и могут измениться в будущем. В то же время, класс реализующий интерфейс IService был создан корневой библиотеке Core так как он не будет меняться. Ниже приведена таблица самими классами и их описанием.

Таблица 2. Описание классов

№	Интерфейс	Реализующий класс	Функция
1.	IBaseCommandHandler	CommandHandler	Отвечает за все операции, связанные с добавлением, изменением и удалением данных. Инкапсулирует в себе классы реализующие интерфейсы IEventRepository и IBaseWriteRepository
2.	IEventRepository	EventRepository	Отвечает за логирование событий, связанных добавлением, изменением и удалением данных
3.	IService	Service	Выполняет методы работы с данными и передает команды в класс CommandHandler и логирует события через класс EventRepository.
4.	IUnitOfWork	UnitOfWork	Распределяют транзакции в репозиториях при операциях связанными с редактированием данных. Инкапсулирует класс BaseWriteRepository, EventRepository.
5.	IReadRepository	SchoolReadRepository, ParticipantReadRepository, TournamentReadRepository	Классы, содержащие в себе SQL запросы в виде строк. Запрашивают данные из базы данных
6.	IWriteRepository	BaseWriteRepository	Класс обновляющий, добавляющий и удаляющий данные

Алгоритм обработки HTTP запроса выглядит следующим образом:

– HTTP запросы обрабатываются веб-сервером Kestrel и оборачиваются в класс HttpContext.

– Затем создаются объекты класса Controller в котором вызывается соответствующий метод обработки запроса в зависимости от URL пути. Всего

было написано четыре контроллера, SchoolsController, TournamentsController и ParticipantsController отвечающие за работу с сущностями «Школы», «Турниры» и «Участники» соответственно, и AuthController ответственный за аутентификацию и авторизацию пользователя

– Если это запрос на получение данных, контроллер напрямую обращается к репозиторию для выполнения операции чтения

– Если это запрос на добавление, изменение или удаление данных, контроллер вызывает соответствующий метод в классе Service, который затем передает команду обработчику команд CommandHandler. Обработчик вызывает метод в объекте WriteRepository и сохраняет событие в репозитории EventRepository.

Для наглядности приведена схема алгоритма на рисунке 7.

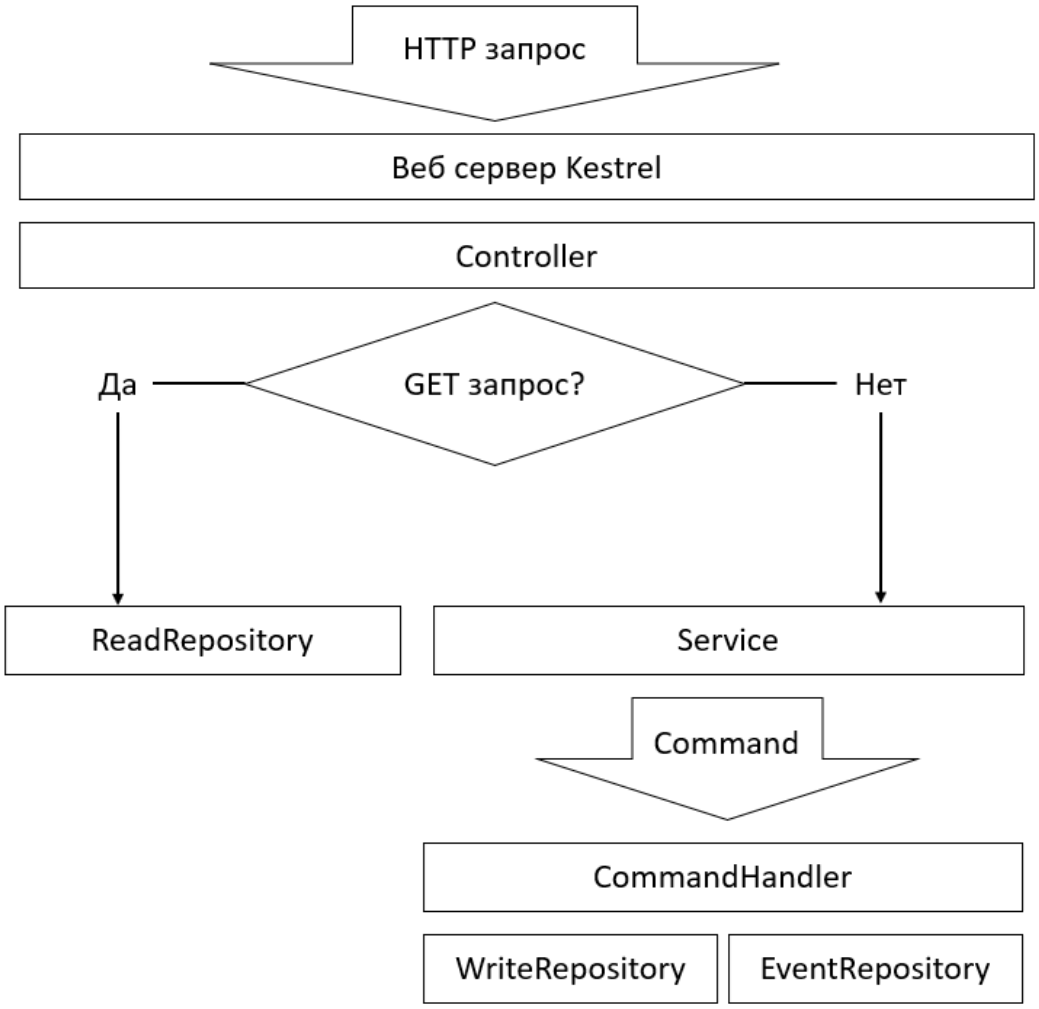

Рисунок 7. Построенный алгоритм работы с серверной системой

2.2 Разработка клиентской части системы

2.2.1 Добавление компонентов Vue и Ant Design

Компоненты являются основными элементами программного кода позволяющие подразделять приложение на логические блоки. В них определяется HTML каркас пользовательского интерфейса и прописывается код на JavaScript, отвечающий за любые логические операции. В папке проекта веб-клиента были созданы компоненты и подразделены по директориям согласно их функциональным задачам и расположения на веб-страницах. Ниже приведен рисунок, на котором изображены эти компоненты в интерфейсе среды разработки.

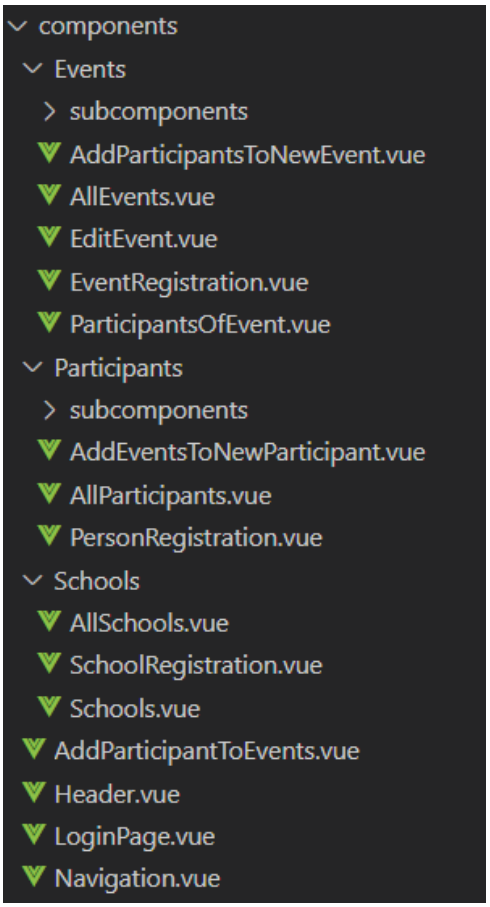

Рисунок 8. Разделение компонентов

Все компоненты можно повторно использовать в любой части приложения веб-клиента. Для того, чтобы использовать созданные компонент, необходимо присвоить ему HTML тэг и разместить присвоенный тэг в другом компоненте либо HTML элементе.

Ниже приведен пример создания компонента Header.

```
<template>
  <a-layout-header class="header">
    <div class="logo">
      <p style="color: white">Клуб управленческой борьбы</p>
    </div>
    <div class="logout" @click="logout">
      <a-icon type="logout" />
    </div>
  </a-layout-header>
</template>
```

Рисунок 9. Компонент Header

Для HTML форм и других элементов была использована дизайн-система And Design. Ant Design это библиотека содержащая в себе готовые Vue компоненты, такие как кнопки, иконки, формы заполнения и другие элементы интерфейса [19]. Например, в библиотеке есть компонент «a-layout-header», который был добавлен в созданный компонент Header, который в свою очередь был добавлен в компонент App посредством импортирования языком JavaScript из директории "./components/Header.vue". Импортирование и экспортирование компонентов происходит в специальном участке кода, обозначенном тэгом script. Помимо импортирования и экспортирования компонентов, в данном блоке добавляются любые методы и объекты языка JavaScript необходимые для любых манипуляций данными в HTML элементах компонента. В каждом файле компонента также присутствует блок style, в котором размещаются CSS классы, устанавливающие стиль для элементов компонента. Ниже приведен пример кода, в котором также можно увидеть блоки, описанные выше.

```
<template>
  <div id="app">
    <a-layout style="height: 100vh">
      <Header />
    </a-layout>
  </div>
</template>
<script>
import Header from "./components/Header.vue";
export default {
  name: "App",
  components: {
    Header,
  },
};
</script>
<style>
#app {
  font-family: Avenir, Helvetica, Arial, sans-serif;
  -webkit-font-smoothing: antialiased;
  -moz-osx-font-smoothing: grayscale;
  text-align: center;
  color: #2c3e50;
  margin-top: 0;
}
</style>
```

Рисунок 10. Компонент App

2.2.2 Добавление общего хранилища Vuex

Одним из вызовов в разработке клиентских веб-приложении является изоляция состояния переменны JavaScript от других компонентов. В отличии от традиционных веб-приложении, SPA приложения позволяют сохранять и передавать данные в переменных между компонентами. В фреймворке Vue за это отвечает система управления состоянием Vuex. Данная система была разработана под влиянием проекта Flux, разработанной компанией Facebook, Vuex позволяет хранить состояние приложения в виде централизованного хранилища, а также предоставлять общий доступ к изменению данных хранилища посред-

ством публичных методов в этом хранилище. При разработке веб клиента такое хранилище было использовано буфера для временного хранения данных регистрируемых новых участников, турниров и школ, списков участников и школ с целью заполнения HTML форм при переходе на новую страницу или редактирования данных в пользовательском интерфейсе. Кроме того, в хранилище имеются переменные, которые идентифицируют об успешной отправке HTTP запросов и в зависимости от их состояния отображается окно оповещения. Все компоненты имеют доступ к данному хранилищу, а именно к ее переменным и методам и при необходимости взаимодействуют с ним. Ниже приведена иллюстрация, описывающая эту взаимосвязь [17].

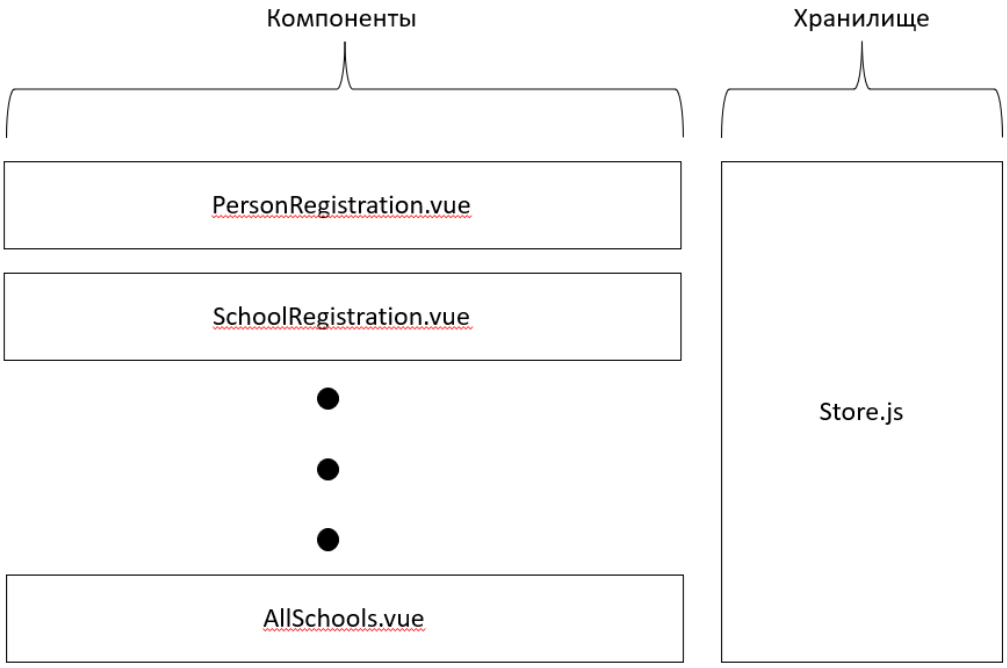

Рисунок 11. Компоненты и хранилище

2.2.3 Реализация обмена данными с серверным приложением

Одной из основных задача любого клиентского веб-фреймворка является поддержка обмена данными с сервером посредством HTTP протокола. Для данной задачи в проект была добавлена и использована библиотека axios основанной на асинхронности языка JavaScript [20]. Все методы созданные для HTTP запросов и использующие axios были подразделены на две группы: методы отправляющий запрос GET и методы отправляющие запросы POST, PUT и DELETE. Методы первой группы были вынесены в отдельный файл

apiMethods.js, в то время как методы второй группы были добавлены в хранилище store.js. Причиной их добавления в хранилище является необходимость доступа к данным, подлежащим отправке на сервер, такие как Id сущностей и данные из заполняемых форм для регистрации сущностей. Все методы вызываются из компонентов напрямую. Ниже приведен пример кода отправляющий запрос GET.

```
export async function getSchools() {
  var data;
  var token = JSON.parse(localStorage.getItem("user")).token;
  await axios.get(process.env.VUE_APP_ROOT_API + "schools", {
     headers: {
       Authorization: `Bearer ${token}`,
     },
   }).then((res) => {
     data = res.data;
   });
  return data;}
```

Рисунок 12. Пример отправки запроса GET

2.2.4 Разработка пользовательского интерфейса

Для разработки пользовательского интерфейса были использованы готовые компоненты библиотеки Ant Design. На странице имеется три основных области: заголовок с кнопкой выхода из системы, вертикальное навигационное меню слева и основное окно справа, в котором будут представлены все формы заполнения данных и таблицы. В навигационном меню присутствуют три основные навигационные вкладки, содержащие в себе две кнопки для регистрации и управления данными о турнирах, участниках и школах соответственно. Навигационное меню с раскрытыми навигационными вкладками приведено на рисунке 13.

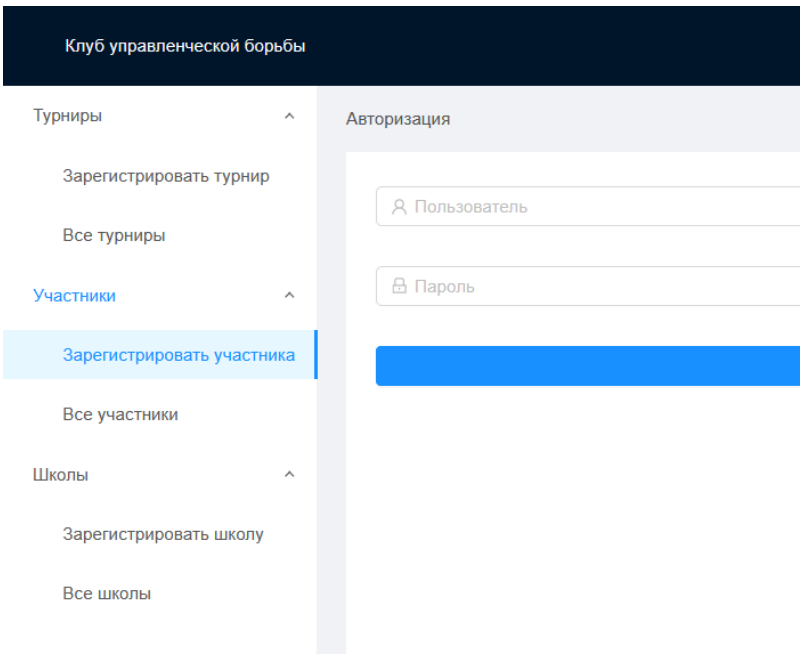

Рисунок 13. Навигационное меню

При нажатии на одну из кнопок в навигационной вкладке, открывается соответствующее окно справа если пользователь авторизировался. В противном случае будет всегда отображаться страница, требующая авторизации.

Для регистрации турнира необходимо нажать на соответствующую кнопку в навигационном меню. Вызываемое окно приведено на рисунке ниже. В данном окно необходимо заполнить форму введя данные турнира.

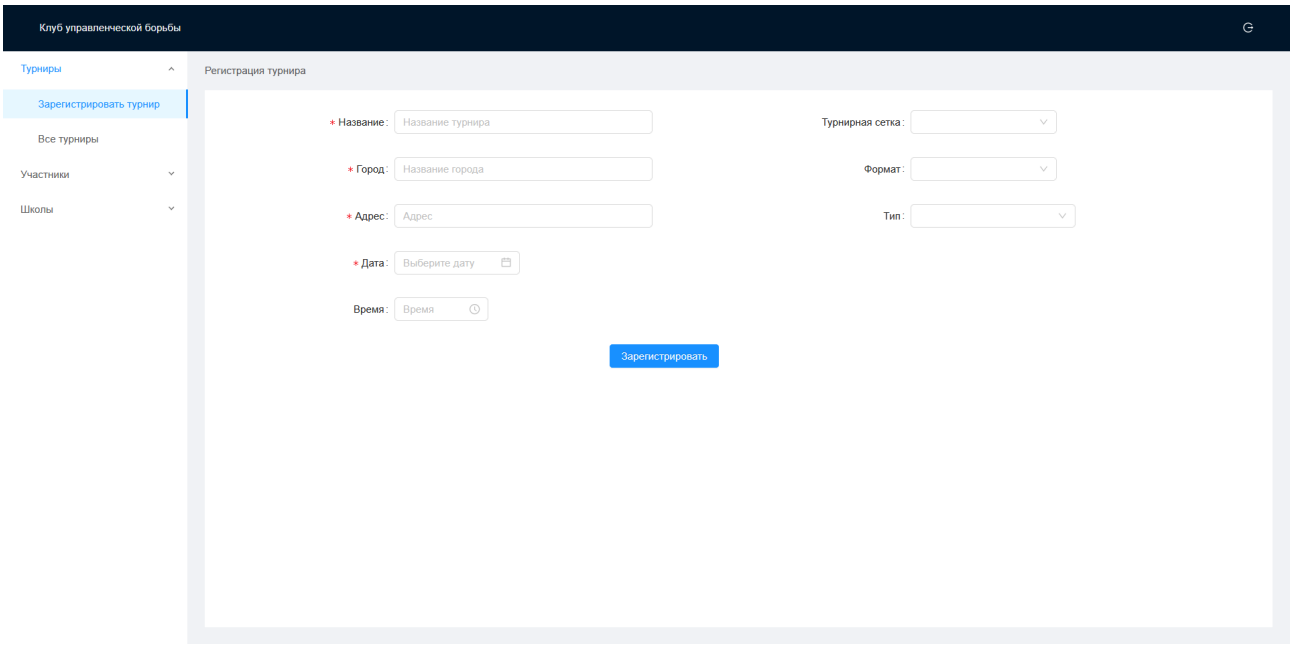

Рисунок 14. Регистрация турнира

Чтобы увидеть весь список турниров нужно нажать на кнопку «Все турниры». Список отобразится в виде таблицы, содержащей зарегистрированные турниры. В колонке «Действие» имеются кнопки позволяющие изменить данные турнира или удалить его, а также кнопка позволяющая увидеть список участвующих в турнире участников для последующего удаления участников и добавления новых. Скриншот данной страницы приведен на рисунке ниже.

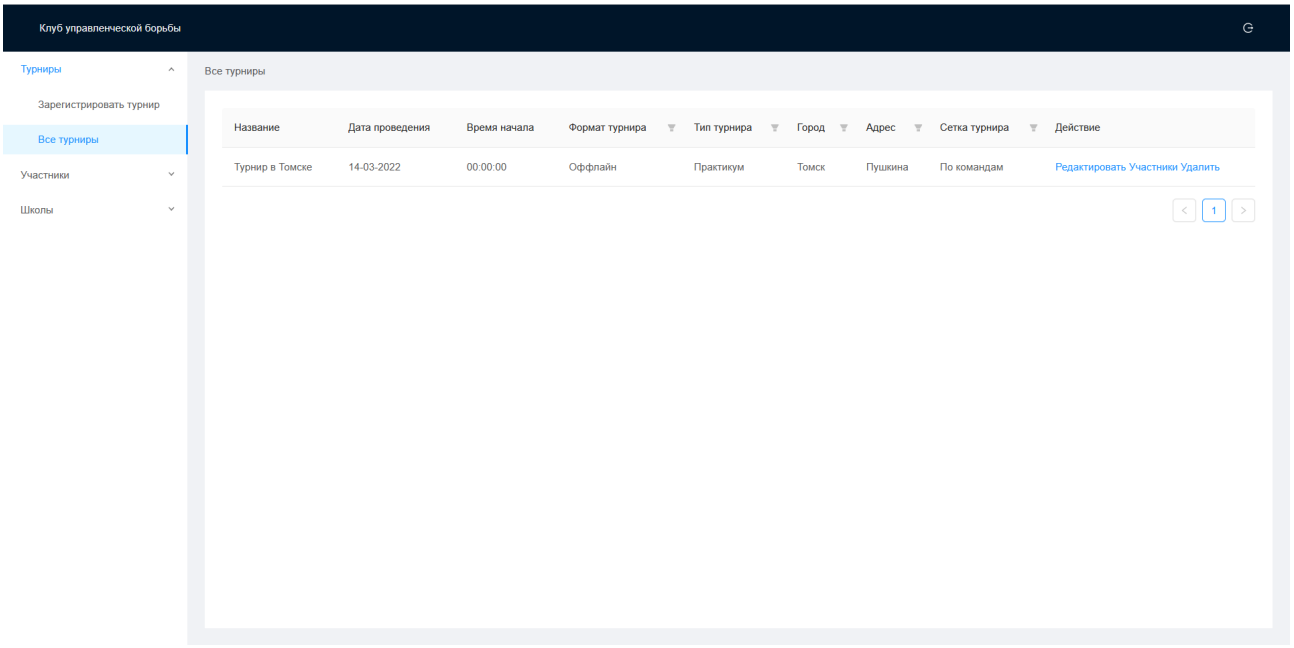

Рисунок 15. Список всех турниров

Ниже приведена область страницы, содержащая форму редактирования турнира.

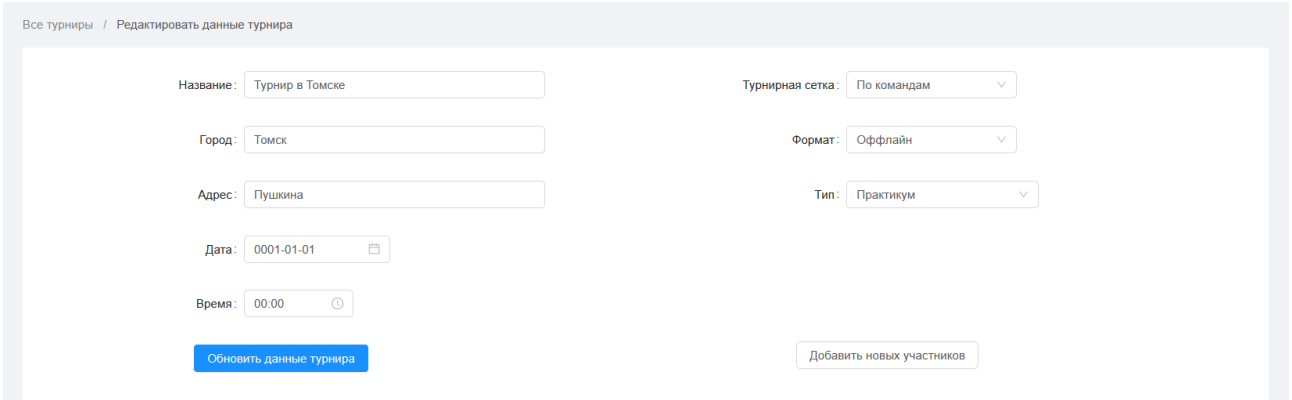

Рисунок 16. Редактирование данных турнира

При нажатии на кнопку «Добавить новых участников» отобразится страница, содержащая таблицу, приведенную на рисунке ниже. В ней можно выбран нужных участников и задать им роли в турнире. После добавления

участника запись исчезнет из таблицы, а ниже отобразиться соответствующее оповещение.

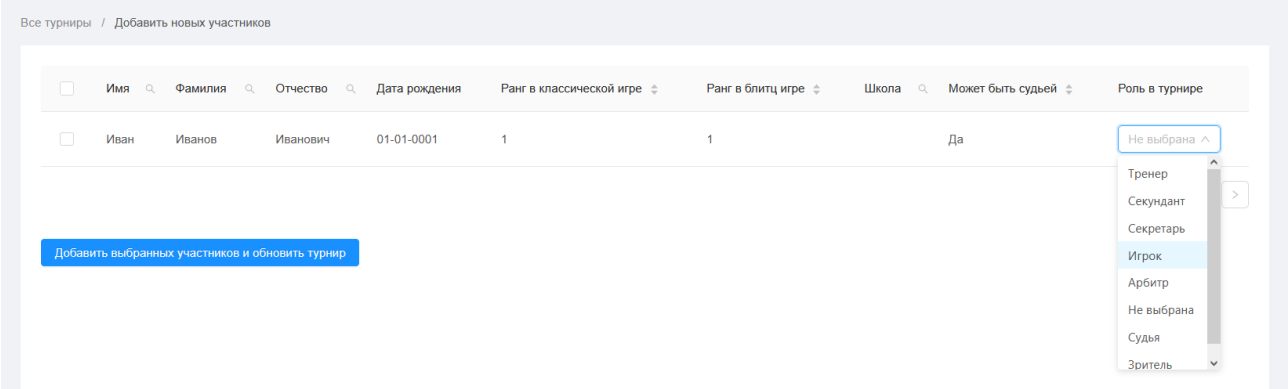

Рисунок 17. Добавление новых участников

Для удаления добавленных участников необходимо вернуться на страницу со всеми турнирами и найти нужный нам турнир. В нем необходимо нажать на кнопку «Участники», после чего отобразиться таблица с участниками в турнире. В ней можно удалить участников. Пример приведен ниже.

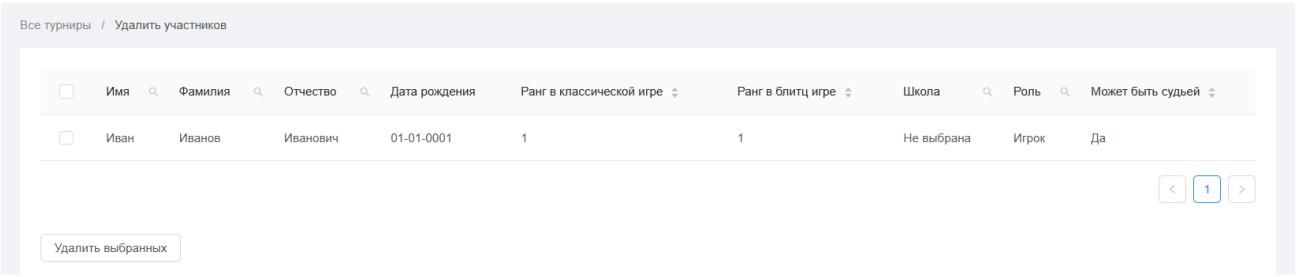

Рисунок 18. Удаление участник из турнира.

Для регистрации и управления данными участников и школ были созданы аналогичные таблицы и формы по структуре расположения кнопок и их функциям. Тем не менее, в таблице «Все участники» нет кнопки удаления участника из турнира. Данная возможность предусмотрена только для таблицы с турнирами. Скриншоты страниц регистрации, редактирования и удаления участников приведены ниже.

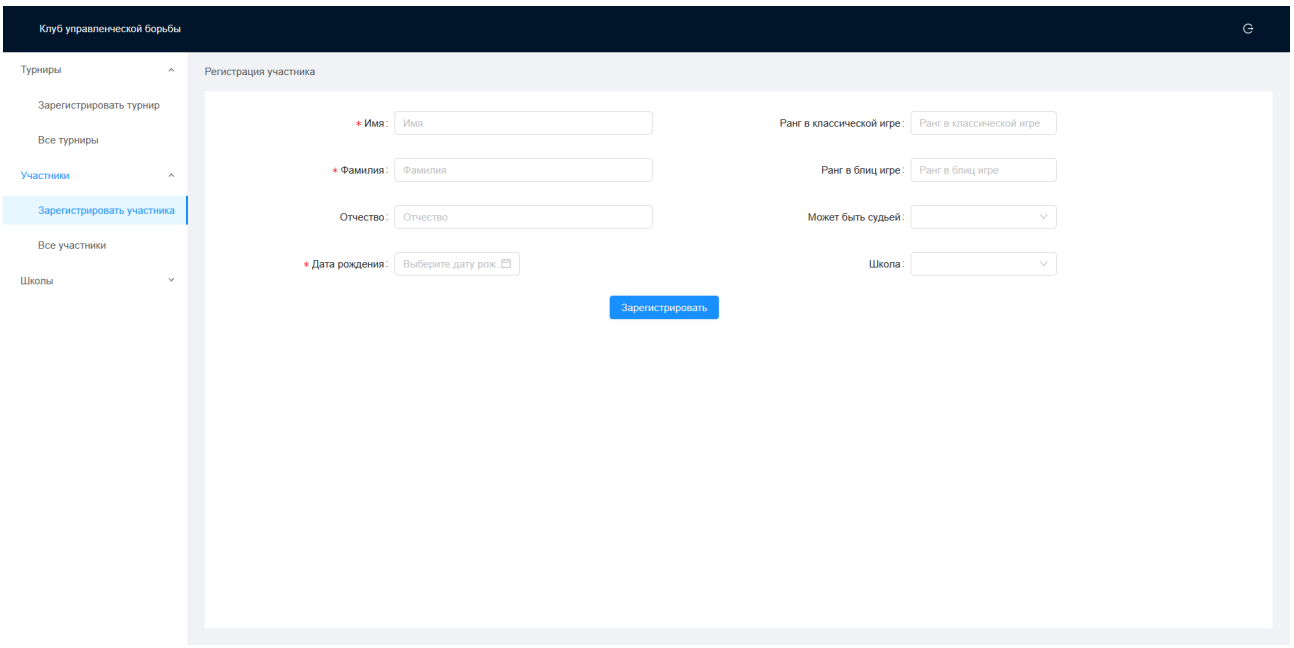

Рисунок 19. Регистрация участников

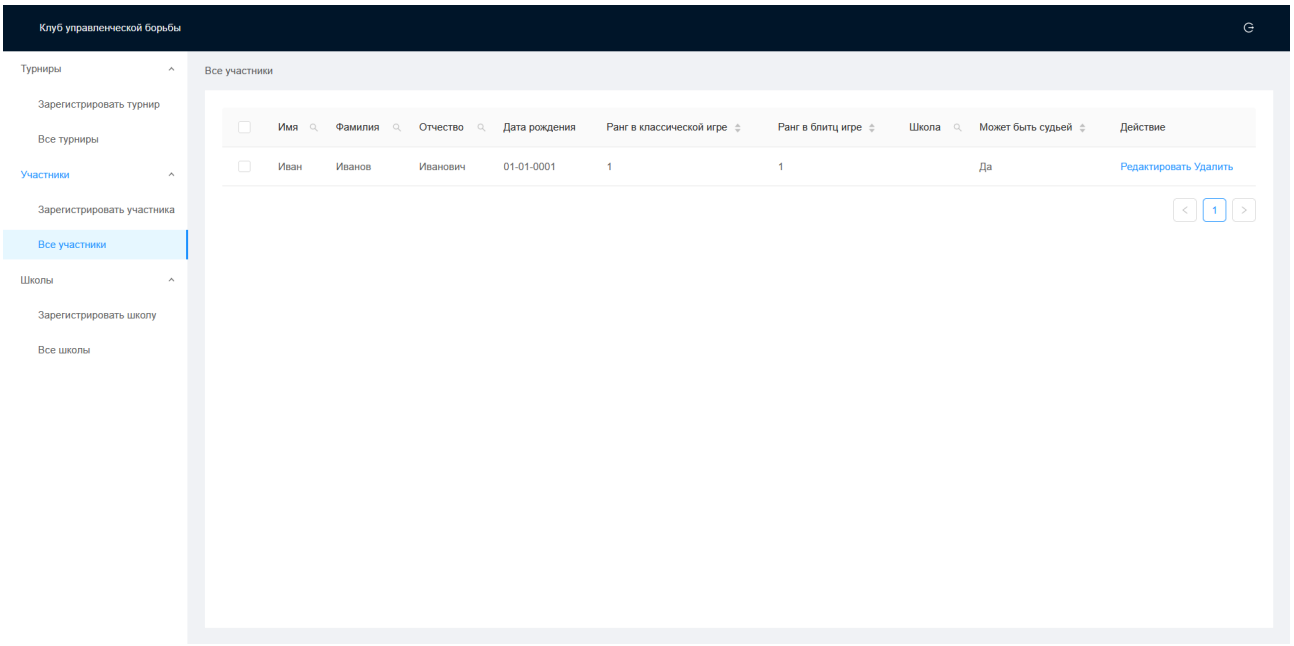

Рисунок 20. Список всех участников

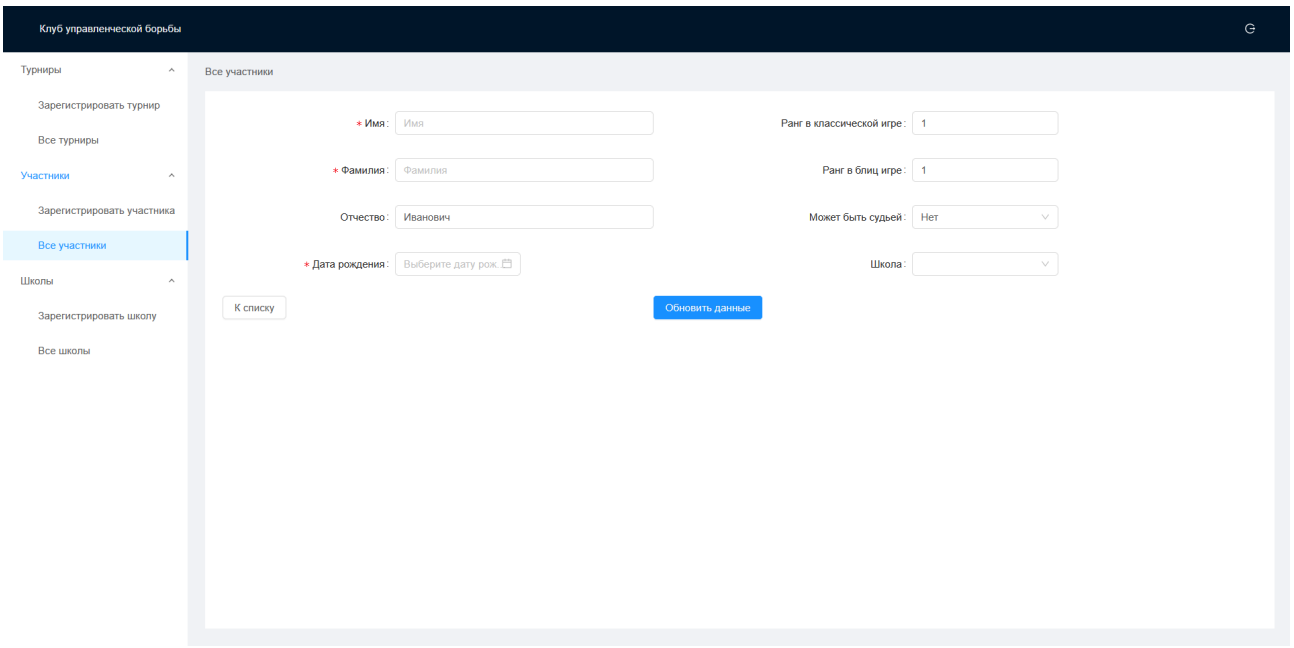

Рисунок 21. Страница изменения данных участника

Для регистрации школы достаточно ввести её название. В дальнейшем при необходимости будут добавление другие поля. Ниже приведены скриншоты страниц связанных с регистрацией, редактированием и удалением школ.

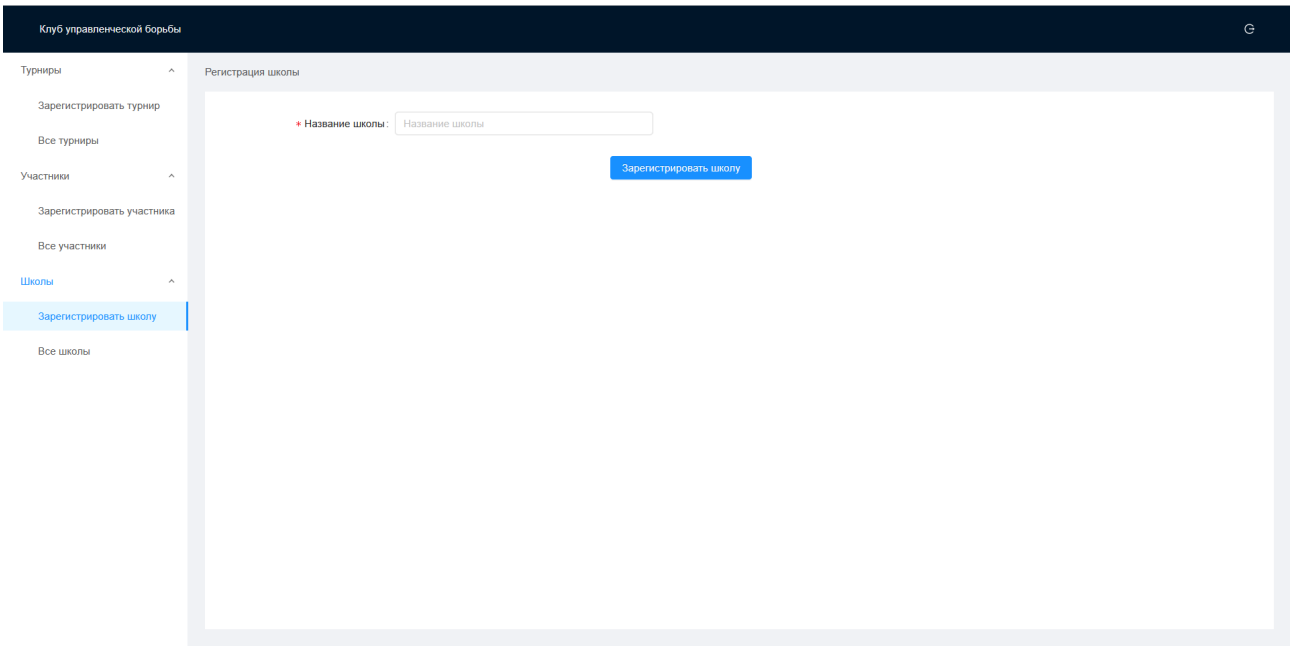

Рисунок 22. Регистрация школы

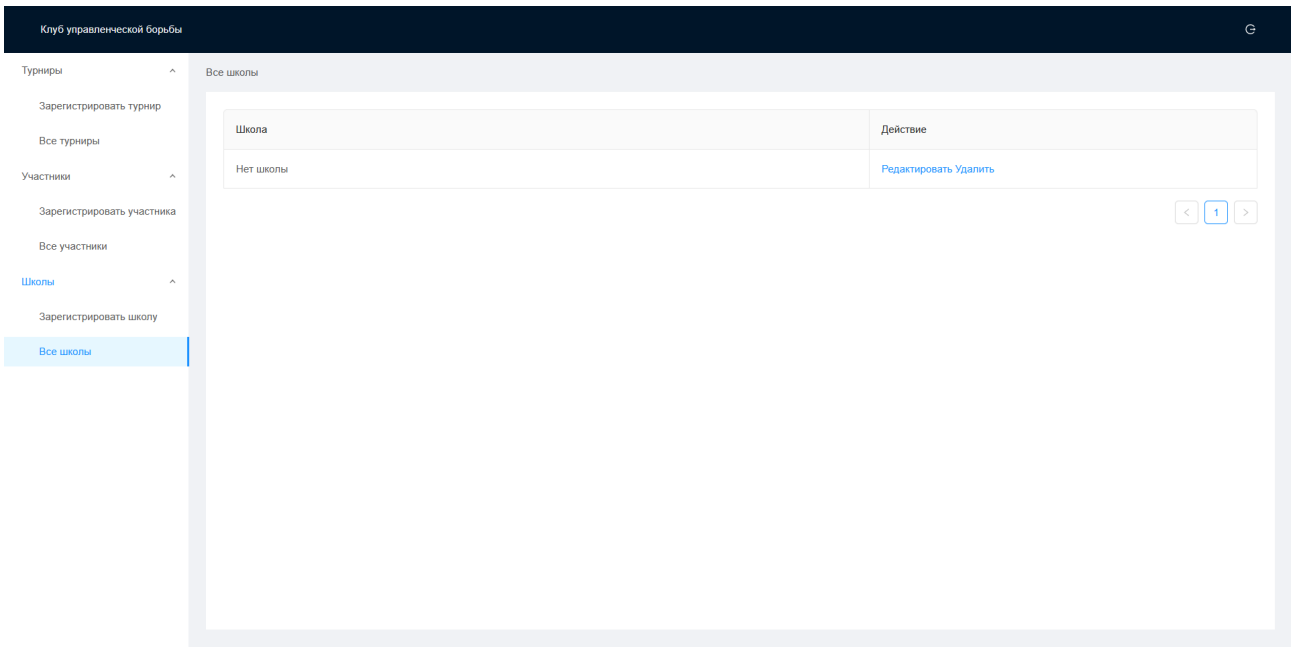

Рисунок 23. Список всех школ

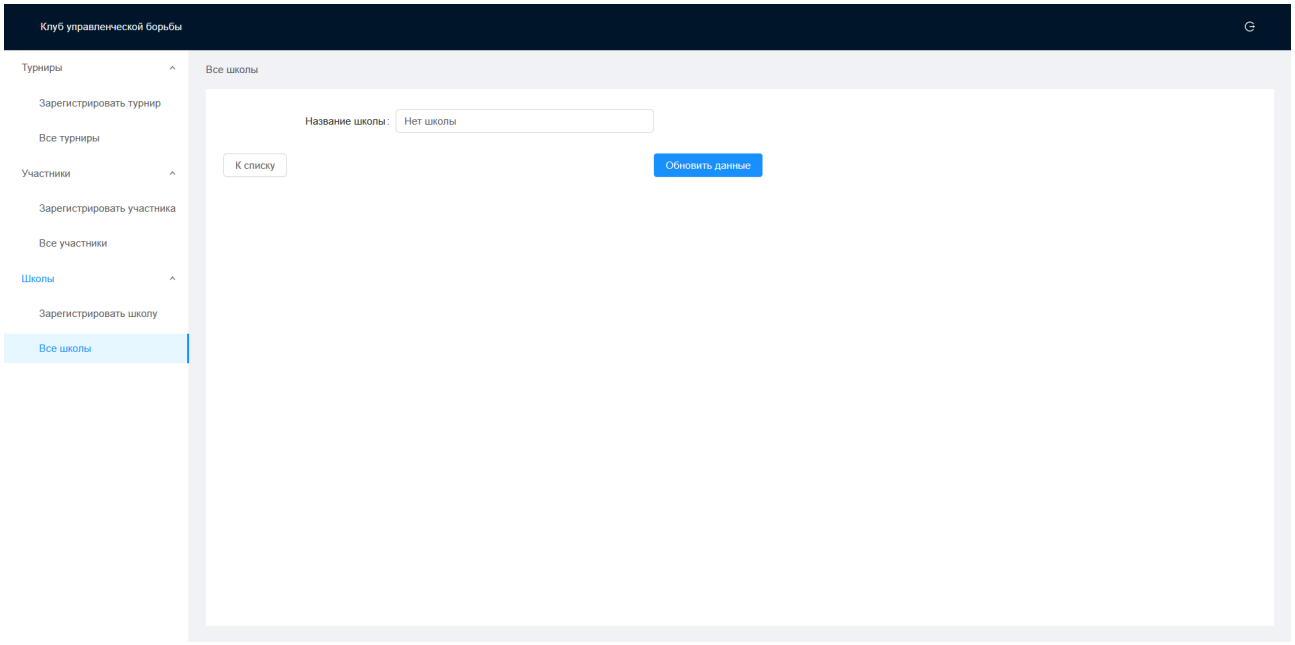

Рисунок 24. Страница изменения данных школы

3. Производительность и расширяемость серверной системы

3.1 Измерение технического долга

Для определения качества написанного кода был проведен статический анализ [21]. В качестве инструмента был использован анализатор SonarQube. Для сравнения качества кода был также проанализирован код, написанный по классической трехслойной архитектуре.

В результате анализа было обнаружено, что код, написанный по принципу DDD, имеет меньший технический долг, который отображается примерном количестве времени, затрачиваемом на его исправление. Технический долг (также известный как проектный долг или долг кода, но может быть также связан с другими техническими начинаниями) — это концепция в разработка программного обеспечения, которая отражает подразумеваемые затраты на дополнительную переделку, вызванную выбором простого (ограниченного) решения сейчас вместо использования лучший подход, который потребует больше времени [22]. Кроме того, написанный код в стиле DDD имеет меньшее количество строк кода и меньший процент дупликаций в нем. Результаты отображены на рисунках ниже.

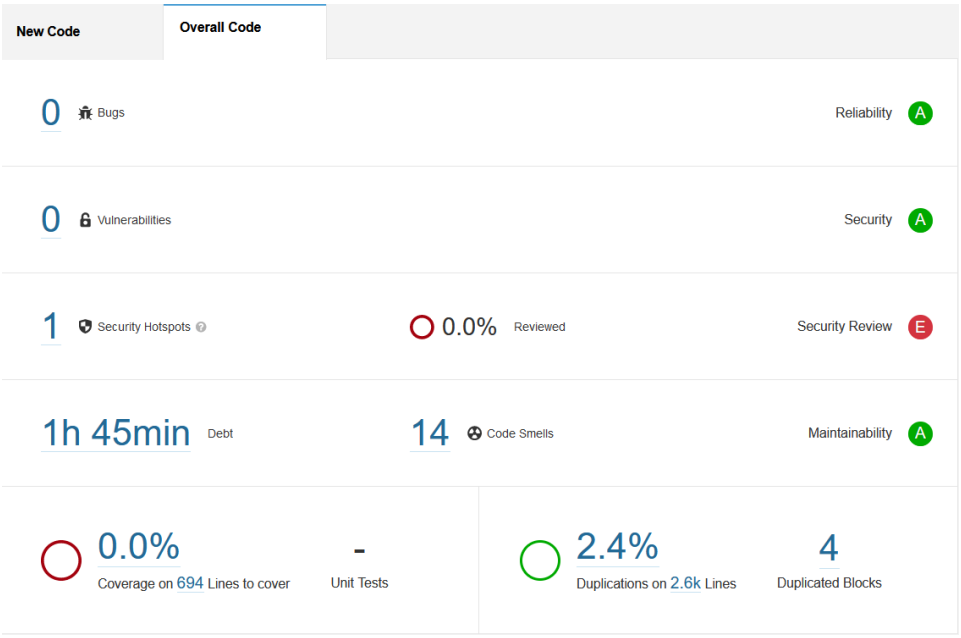

Рисунок 25. Результаты анализа кода с традиционной архитектурой

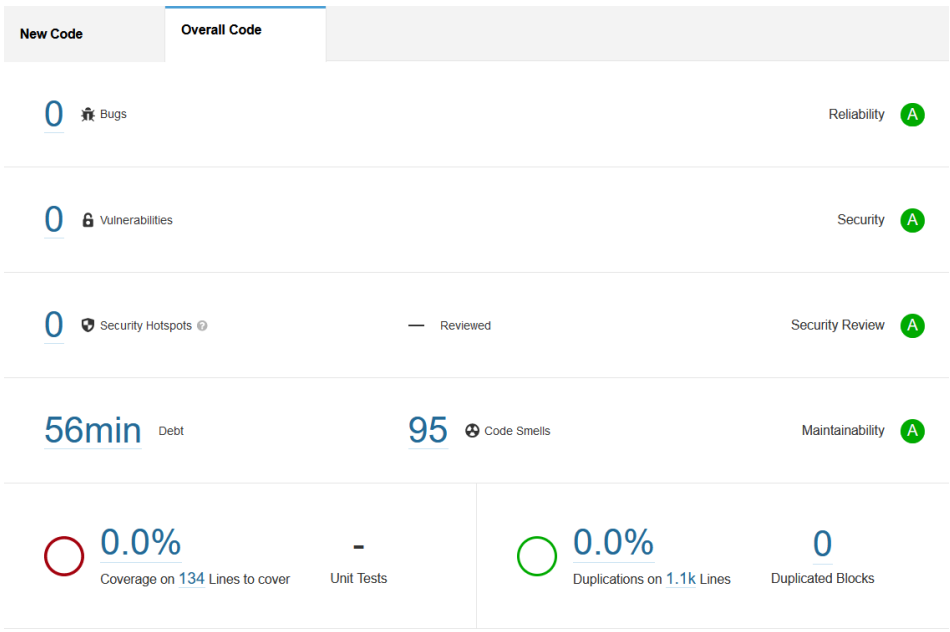

Рисунок 26. Результаты анализа кода написанного по DDD

3.2 Оценка скорости получения данных

Благодаря шаблону CQRS, в части системы, выполняющей чтение данных из базы данных, были написаны чистые запросы вместо применения ORM Entity Framework Core. Помимо гибкости составления новых запросов для выборки данных при расширении системы, данный подход в разы ускоряет производительность системы. Для анализа производительности таблица турниров в базе данных была заполнена тестовыми данными в количестве 1000 сущностей и добавлен тестовый метод с HTTP адресом конечной точки, в котором для запроса данных используется ORM. Тестирование было проведено на компьютере с техническими характеристиками, отображенными в таблице ниже.

Таблица 3. Технические характеристики компьютера

Процессор	Intel core I5 9300H (4x2.40 ГГц)
Оперативная память	RAM 8 ГБ (2666 МГц)
Жесткий диск	SSD 512 ГБ
Операционная система	Windows 10 Pro 64 bit

В методе с чистым SQL запросом и в тестовом методе были добавлены объекты класса Stopwatch, который позволяет определить скорость выполнения выбранного участка кода. Данные методы изображены на рисунках ниже.

```
public async Task<IActionResult> GetTest()
        {
            Stopwatch stopWatch = new Stopwatch();
            stopWatch.Start();
            var res = await _context.Tournaments.Select(e => new TournamentDto
            {
                Id = e.Id, TournamentName = e.TournamentName,
                Date = e.Date, StartTime = e.StartTime,
                EndTime = e.EndTime,  TournamentFormat = e.TournamentFormat.Format,
                TournamentGrid = e.TournamentGrid.Type, TournamentType =
e.TournamentType.Type,
                City = e.City,  Address = e.Address,
            }).ToListAsync();
            stopWatch.Stop();
            Console.WriteLine("raw" + stopWatch.Elapsed.Milliseconds.ToString());
            return Ok(res);
        }
```

Рисунок 27. Тестовый метод

В результате запроса данных из этого метода были получены следующие результаты.

Таблица 4. Скорость запроса данных через ORM

Номер HTTP запроса	Длительность выполнения в миллисекундах (мс)
1.	8
2.	20
3.	47
4.	14
5.	10
6.	12
7.	6
8.	13
9.	6
10.	12

```
public async Task<List<TournamentDto>> GetAllAsync()
{
   …
   Stopwatch stopWatch = new Stopwatch();
   stopWatch.Start();
   using (connection) {
      SqlCommand command = new SqlCommand(
            "SELECT Tournaments.Id, Tournaments.TournamentName, Tournaments.Date, Tournaments.StartTime, Tournaments.EndTime, TournamentFormats.Format, Tournament-GridTypes.Type, TournamentTypes.Type, Tournaments.City, Tournaments.Address
            FROM Tournaments
      INNER JOIN TournamentFormats on Tournaments.TournamentFormatId = TournamentFormats.Id
      INNER JOIN TournamentGridTypes on Tournaments.TournamentGridId = Tournament-GridTypes.Id
      INNER JOIN TournamentTypes on Tournaments.TournamentTypeId = TournamentTypes.Id ORDER BY Tournaments.Date",  connection);
         connection.Open();
         await command.ExecuteNonQueryAsync();
   …
         stopWatch.Stop();
         Console.WriteLine("raw" + stopWatch.Elapsed.Milliseconds.ToString());
         reader.Close();
   }
   return items;
   }
```

Рисунок 28. Рабочий метод

В результате запроса данных из метода с чистым SQL запросом были получены следующие результаты.

Таблица 5. Скорость запроса данных без ORM

Номер HTTP запроса	Длительность выполнения в миллисекундах (мс)
11.	6
12.	2
13.	2
14.	2
15.	2
16.	2

17.	2
18.	2
19.	2
20.	2

В ходе тестирования было выявлена существенная преимущество в скорости выполнения запросов без применения ORM. Причиной такого различия в производительности является генерация SQL запроса библиотекой Entity Framework Core и ее учет состояния данных в базе данных. Таким образом, используя шаблон CQRS, система стала более производительной благодаря применению готовых SQL запросов при чтении данных.

4. Развертывание системы

Чтобы развернуть систему было принято решение использовать облачные вычисления. Облачные вычисления обеспечивают доступ к вычислительным ресурсам, ресурсам хранения и сетевым ресурсам по запросу. В зависимости от выбранных типов услуг и моделей развертывания, облачные вычисления могут помочь контролировать расходы, позволяя при этом быстро запускать новые продукты и услуги, расширять деятельность и обеспечивать максимальную производительность и продуктивность [23]. Для развертывания системы был выбран облачный сервис Microsoft Azure. Компания Microsoft предоставляет бесплатный доступ к сервисам Azure в течение одного года.

Для развертывания серверной части системы первым делом нужно зарегистрироваться в портале Azure и получить подписку. Затем необходимо установить на компьютер Azure CLI. В терминале Azure CLI необходимо пошагово ввести команды, приведенные в таблице ниже [24].

Таблица 6. Развертывание серверного приложения

№	Команда	Описание
1.	az group create --location eastus --name kub-core-sql	Создание группы ресурсов. Группа ресурсов служит в качестве контейнера для всех служб Azure связанных с этим приложением.
2.	az appservice plan create \ --name kub-core-sql-plan-2022 \ --resource-group kub -core-sql \ --sku F1	Данная команда определяет тарифный план. Параметр –sku F1 устанавливает бесплатный сервис план.
3.	az webapp create \ --name kub-service-plan \ --runtime "DOTNET|6.0" \	Данная команда создает веб службу. В дальнейшем она будет находит-

	--plan kub-service-plan \ --resource-group kub-core-sql	ся по адресу https://kub-core-sql.azurewebsites.com
4.	az sql server create \ --location eastus \ --resource-group kub-core-sql \ --name kub-server \ --admin-user kubadmin \ --admin-password Univerza_maribor_123	Данная команда создает сервер базы данных и устанавливает параметры доступа.
5.	az sql db create \ --resource-group msdocs-core-sql \ --server kub-server \ --name kub-db	Создание сервера займет некоторое время. После завершения создания, необходимо ввести данную команду для создания базы данных.
6.	az sql server firewall-rule create \ --resource-group kub-core-sql \ --server kub-server \ --name AzureAccess \ --start-ip-address 0.0.0.0 \ --end-ip-address 0.0.0.0	Данная команда настраивает доступ к созданной базе данных
7.	az webapp deployment source config-local-git \ --name kub-app \ --resource-group kub-core-sql	Введя данную команду, мы настраиваем Git репозиторий для разворачивания
8.	az webapp deployment list-publishing-credentials \ --name kub-app \ --resource-group kub-core-sql \ --query "{Username:publishingUserName,	Этой командой мы получаем имя пользователя и пароль необходимые для развертывания через git

	Password:publishingPassword}"	
9.	git remote add azure https://kub-app.scm.azurewebsites.net/kub-app.git	Добавляем точку адрес развертывания Azure в git
10.	git push azure main:master	Отправляем код для развертывания из ветки, в которой находится код
11.	az sql db show-connection-string \ --client ado.net \ --name kub-db \ --server kub-server	Запрашиваем строку соединения для базы данных
12.	az webapp config connection-string set \ -g kub-core-sql \ -n kub-app \ -t SQLServer \ --settings MyDbConnection=<your-connection-string>	Назначаем полученную строку соединения веб службе вместо <your-connection-string>
13.	az sql server firewall-rule create --resource-group msdocs-core-sql --server kub-server --name LocalAccess --start-ip-address <your-ip> --end-ip-address <your-ip>	Этой командой мы настраиваем доступ к серверу БД. Вместо <your-ip> вписывается ip адрес компьютера, на котором происходит развертывание
14.	"ConnectionStrings": { "MyDbConnection": "Server=tcp:kub-server.database.windows.net,1433; Initial Catalog=kub-db; Persist Security Info=False;	Данный JSON атрибут необходимо добавить в файл appsettings.json в проекте .NET.

	User ID=kub-admin;Password=Univerza_maribor_123 Encrypt=True; TrustServerCertificate=False;" }	
15.	dotnet tool install -g dotnet-ef \ dotnet ef migrations add InitialCreate \ dotnet ef database update	Вызывая данную команду мы создаем схему базы данных.

Чтобы развернуть клиентскую часть системы необходимо перейти на портал Azure, нажать на кнопку «Create a Resource», найти «Static Web Apps», выбрать «Static Web Apps» и нажать «Create». Отобразиться страница, приведенная на рисунке ниже. В ней будет необходимо ввести название клиента и указать репозиторий зайдя через аккаунт Github. После этого необходимо выбрать организацию, репозиторий и ветку репозитория. Указав эти параметры и перейдя на страницу «Review + create» необходимо нажать на кнопку «Review + create».

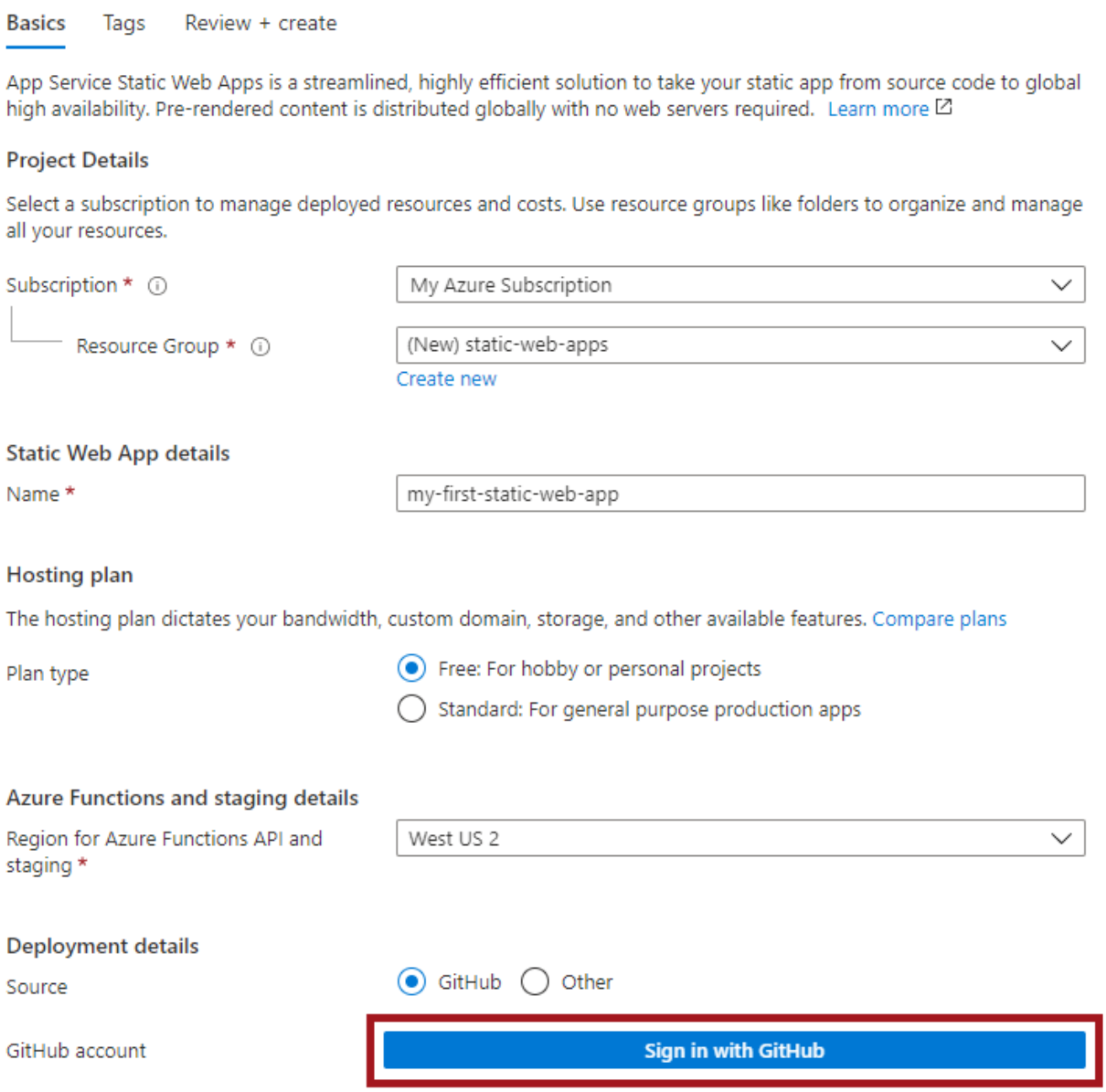

Рисунок 29. Установка репозитория Github

5. Финансовый менеджмент, ресурсоэффективность и ресурсосбережение

Целью раздела «Финансовый менеджмент, ресурсоэффективность и ресурсосбережение» является оценка перспективности и научно-исследовательского проекта и разработка способов управления проектных решений на этапе реализации. Достижение цели обеспечивается решением следующих задач:

1. оценка коммерческого потенциала и перспективности проведения научных исследований;

2. планирование научно-исследовательских работ;

3. определение ресурсной (ресурсосберегающей), финансовой, бюджетной, социальной и экономической эффективности исследования.

С учетом решения данных задач была сформирована структура и содержание раздела «финансовый менеджмент, ресурсоэффективность и ресурсосбережение».

5.1 Предпроектный анализ

Представим потенциальных потребителей результатов исследования. Разработанная система будет использоваться для проведения соревнований по управленческой борьбе, ведения учета проведенных соревнований, ведения рейтинга участников и отображения предстоящих и прошедших турниров. Данная система представляет из себя веб приложение, доступ которому можно получить через интернет браузер. Следовательно, доступ к системе можно получить с любого компьютера и выдавать его по подписке по определенному тарифу.

Основываясь на особенностях информационной системы, можно прийти к выводу, что система может быть использована компаниями, которые на регулярной основе проводят соревновательные мероприятия, в которых участвует некоторое количество людей. Кроме того, система может быть использована компаниями, не специализирующимися организацией такого рода мероприятий, однако имеющими необходимость иногда проводить менеджерские турни-

ры для своих работников.

Для сегментирования рынка услуг были проанализированы потенциальные пользователи системы и функции которые могут предоставлены системой. Результаты анализа отображены в таблице х.

Таблица 7. Карта сегментирования по разработке системы

		Пользователи		
		Компании любой сферы производства	Бизнес тренеры	Исследователи (в области психологии и менеджмента)
Функции	Платная подписка для создания турнира,	+	+	
	Платная подписка для доступа к результатам и аналитике	+	+	+
	CRM	+		
	Организация турниров	+	+	

В результате были определены основные сегменты рынка.

В качестве основного сегмента на котором будет ориентироваться предприятие были выбраны компании любой сферы производства, которые самостоятельно проводят тренинги для своих сотрудников.

Компании являются привлекательными для разработки нового функционала.

В настоящий момент на рынке существует несколько продуктов- конкурентов и компаний, которые занимаются созданием решений для регистраций и организации мероприятий:

1. Eventboost – система регистраций мероприятий, создания приглашений, мониторинга мероприятий;

2. RegToEvent – решение предлагающее автоматическую регистрацию участников, на мероприятия любой тематики и любого масштаба, продажу билетов различных категорий и с набором дополнительных опций, онлайн оплата

билетов через различные платежные системы;

3. Depreg – платформа для организации онлайн и офлайн регистрации, продажа билетов. Также были выделены следующие критерии оценки продуктов.

Технические критерии оценки ресурсоэффективности:

1. регистрация школ участников турниров;

2. добавление функционала для заказчика;

3. изменение данных участников;

4. редактирование типа турниров;

5. добавление участников в турниры;

Экономические критерии оценки эффективности:

1. обслуживание;

2. цена.

Экспертная оценка основных технических характеристик данных продуктов представлена в таблице х.

Таблица 8. Результаты оценки технических решений конкурентов

№	Критерии оценки	Вес	Баллы				Конкурентоспособность			
			Бф	Бк1	Бк2	Бк3	Кф	К1	К2	К3
	1	2	3	4	5	6	7	8	9	10
Технические критерии оценки ресурсоэффективности										
1	регистрация школ участников турниров	0,05	5	4	4	3	0,25	0,2	0,2	0,15
2	добавление функционала для заказчика	0,25	5	1	1	1	1,25	0,25	0,25	0,25
3	изменение данных участников	0,05	5	5	4	4	0,25	0,25	0,2	0,2
4	редактирование типа турниров	0,1	5	3	2	3	0,5	0,3	0,2	0,3
5	добавление участников в турниры	0,25	5	4	4	2	1,25	1	1	0,5
Экономические критерии оценки эффективности										
6	сопровождение	0,1	5	2	3	2	0,5	0,2	0,3	0,2

7	цена	0,2	5	3	2	3	1	0,6	0,4	0,6
	Итого	1	35	22	20	18	5	2,8	3.25	2,2

По полученным результатам можно сделать вывод, что разрабатываемое программное обеспечение является наиболее конкурентоспособным и наиболее эффективным для организации мероприятий по управленческой борьбе.

Был применен SWOT-анализ определения ключевых факторов внешней и внутренней среды объекта планирования. В таблице х приведена интерактивная матрица проекта.

Таблица 9. Интерактивная матрица проекта

Сильные стороны проекта				
Возможности		Си1	Си2	Си3
	В1		+	+
	В2			+
Слабые стороны проекта				
Возможности		Сл1	Сл2	Сл3
	В1		+	
	В2			+
Сильные стороны проекта				
Угрозы		Си1	Си2	Си3
	У1		+	
	У2		+	+
Слабые стороны проекта				
Угрозы		Сл1	Сл2	Сл3
	У1	+	+	
	У2		+	

Для исследуемой разработки была составлена итоговая матрица SWOT-анализа, приведенная в таблице 10.

Таблица 10. SWOT-анализ

	Сильные стороны	**Слабые стороны**
	Си1. Отсутствие аналогов	Сл1. Монолитная архитектура
	Си2. Расширяемая архитектура системы	Сл2. Зависимость от сервиса Azure
	Си3. Размещение в облачном сервисе Azure	Сл3. API веб сервиса, разработанный под веб клиента
Возможности		
В1. Доработка и	В1Си2Си3. Система	В1Сл2. При изменениях в

расширение функционала системы	может быть расширена в кратчайшие сроки благодаря модульной архитектуре и размещению в облачном сервисе	облачном сервисе Azure, могут возникнуть проблемы с дальнейшим хостингом
В2. Разработка клиента для мобильных устройств	В2Си3. Так как веб сервис уже развернут в облачном сервисе, подключение к нему с дальнейшим тестированием мобильного клиента не вызовет проблем	В2Сл3. Потребуется разработать новое API веб сервиса для подключения к мобильному клиенту
Угрозы		
У1. Проблемы с поддержкой веб клиента в силу уязвимостей библиотек на Javascript	У1Си2. Архитектура системы позволяет интегрировать ее к работе на личном ПК	У1Сл1Сл2. Может быть затрачено много времени на изменение системы. Для этой цели необходимо нанимать дополнительных людей.
У2. Появление более дешèвых аналогов	У2Си2Си3. Высокий уровень расширяемости и производительности системы позволит дорабатывать системы согласно требованиям Заказчика	У3У2Сл2.Перейти на дешевый хостинг
У3. Рост цены за хостинг веб сервиса		

В результате SWOT-анализа были обнаружены ключевые направления дальнейшего развития разработки, а также методы нивелирования слабых сторон разработки и противодействия потенциальным угрозам.

Самым большим преимуществом данной разработки является расширяемость, а недостатком – большое количество времени которое может быть потрачено на ее расширение.

Для определения степени готовности проекта к коммерциализации была

заполнена форма, приведенная в таблице 11.

Таблица 11. Форма готовности проекта к коммерциализации

№ п/п	Наименование	Степень проработанности научного проекта	Уровень имею-щихся знаний у разработчика
1.	Определен имеющийся научно-технический задел	5	4
2.	Определены перспективные направления коммерциализации научно-технического задела	5	4
3.	Определены отрасли и технологии (товары, услуги) для предложения на рынке	4	5
4.	Определена товарная форма научно-технического задела для представления на рынок	3	3
5.	Определены авторы и осуществлена охрана их прав	4	4
6.	Проведены маркетинговые исследования рынков сбыта	3	3
7.	Определены пути продвижения научной разработки на рынок	4	3
8.	Разработана стратегия (форма) реализации научной разработки	4	3
9.	Проработаны вопросы использования услуг инфраструктуры поддержки, получения льгот	3	3
10.	Проработаны вопросы финансирования коммерциализации научной разработки	3	5
11.	Имеется команда для коммерциализации научной разработки	2	4
12.	Проработан механизм реализации научного проекта	4	4
13.	**ИТОГО БАЛЛОВ**	45	45

В результате оценки было выявлено что перспективность проекта является выше средней. Для дальнейшей улучшения готовности к коммерциализации необходимо составить команду для коммерциализации научной разработки научного проекта и повысить уровень имеющихся знаний у разработчика.

В качестве метода коммерциализации исследования можно открыть ИП для разработчика.

Представим структуру устава научного проекта магистерской работы:

1. Цель и результат проекта
2. Организационная структура проекта
3. Ограничения и допущения проекта

Перед тем как определить цели работы, нужно обозначить заинтересованные в проекте стороны. Данная информация предоставлена в таблице х:

Таблица 12. Заинтересованные стороны проекта

Заинтересованные стороны	**Ожидания**
Компания-пользователь (клуб управленческой борьбы)	Управления организацией мероприятий по управленческой борьбе и веденеие учета
Разработчик	Получение заработной платы
Компания разработчика	Получение прибыли с программного продукта
Научный руководитель, студент	Выполненная выпускная квалификационная работа

Цели и результат проекта представлены в таблице 13.

Таблица 14. Цели и результат проекта

Цели проекта	– Изучить предметную область. – Разработать программное обеспечение. – Разработать реализовать бизнес логику заказчика. – Провести развертывание и тестирование системы. Внедрить разработанную систему и дать доступ заказчику.
Ожидаемые результаты	– Успешная разработка, тестирование и выпуск программного обеспечения. – Успешная сдача выпускной квалификационной работы.

Критерии приемки	Соответствие функциональным требованием в ходе проведенного тестирования.
Требования к результату проекта	Требования к программному обеспечению должны быть выполнены

Рабочая группа проекта была определена и отображена в таблице х

Таблица 15. Рабочая группа проекта

№ п/п	ФИО, основное место работы, должность	Роль в проекте	Функции
1.	Мухтар Дархан, ТПУ, магистрант	Исполнитель	Разработка системы
2.		Руководитель	Руководит проект
3.	Кирилл Кахаев, Томское региональное отделение Федерации управленческой борьбы	Заказчик	Формирование целей и требований проекта

В качестве фактора ограничения в рамках данного проекта существует только дата завершения проекта, а именно конец весны 2022 года.

5.2 Планирование научно-исследовательских работ

Работы, подлежащие выполнению, выглядят следующим образом:

- определение комплекса работ в разработке системы;
- определение участников и исполнителей для каждой работы;
- установление длительности работ;
- составление планируемого графика разработки.

В таблице х приведен список работ, подлежащих выполнению в процессе разработки.

Таблица 16. Перечень работ и исполнителей при разработке

№	Наименование работы	Исполнители работы
1.	Выбор научного руководителя работы	Мухтар Дархан Русланулы

2.	Составление и утверждение темы работы	Мирошниченко Евгений Александрович, Мухтар Дархан Русланулы
3.	Составление календарного плана-графика выполнения работы	Мирошниченко Евгений Александрович, Мухтар Дархан Русланулы
4.	Подбор и изучение литературы по теме работы	Мухтар Дархан Русланулы
5.	Анализ предметной области	Мухтар Дархан Русланулы
6.	Проектирование схемы БД	Мухтар Дархан Русланулы
7.	Разработка серверной части системы	Мухтар Дархан Русланулы
8.	Разработка клиентской части системы	Мухтар Дархан Русланулы
9.	Согласование выполненной работы с научным руководителем	Мирошниченко Евгений Александрович, Мухтар Дархан Русланулы
10.	Выполнение других частей работы (финансовый менеджмент, социальная ответственность)	Мухтар Дархан Русланулы
11.	Подведение итогов, оформление работы	Мирошниченко Евгений Александрович, Мухтар Дархан Русланулы

Трудоемкость выполнения научного исследования оценивается экспертным путем в человеко-днях и носит вероятностный характер, т.к. зависит от множества трудно учитываемых факторов. Для определения ожидаемого (среднего) значения трудоемкости используется следующая формула:

$$t_{ож} = \frac{3t_{min} + 2t_{max}}{5} \quad (1)$$

где t_{min} – предположительно минимальная продолжительность этапа в рабочих днях, определяемая методом экспертной оценки; t_{max} – предположительно максимальная продолжительность этапа в рабочих днях, определяемая методом экспертной оценки.

Исходя из ожидаемой трудоемкости работ, определяется продолжительность каждой работы в рабочих днях *T*р, учитывающая параллельность выполнения работ несколькими исполнителями.

$$T_{p_i} = \frac{t_{ожi}}{ч_i} \quad (2)$$

где T_{p_i} – продолжительность одной работы, раб. дн.;

$t_{ожi}$ – ожидаемая трудоемкость выполнения одной работы, чел.-дн.

Чi – численность исполнителей, выполняющих одновременно одну и ту же работу на данном этапе, чел.

Продолжительность каждого этапа рассчитывается по формуле:

$$t_{раб} = t_{ож} \cdot k_d \quad (3)$$

где $t_{раб}$ – длительность этапов в рабочих днях;

k_d – коэффициент, учитывающий дополнительное время на консультации и

согласование работ, k_d = 1.2

Представим диаграмму Ганта и график работы для проекта.

Линейный график строится на основании полученных значений tраб, предварительно переведенных в календарные дни по формуле:

$$t_к = t_{раб} \cdot K_н \quad (4)$$

где $t_к$ – длительность этапов работ в календарных днях; $K_н$ – коэффициент календарности.

Коэффициент календарности рассчитывается по формуле:

$$K_н = \frac{Тк}{Тк - Твд - Тпд} \quad (5)$$

где Tк – количество календарных дней, Tк = 365;

Tвд – количество выходных дней, Tвд 108;

Tпд – количество праздничных дней, Tпд =10;

Все расчеты сведены в таблицу х.

Таблица 17. Временные показатели проведенных работ

№	Исполнители	Продолжительность работ в днях			Трудоемкость	
		t_{min}	t_{max}	$t_{ож}$	$t_{раб}$	t_k
1.	НР	1	2	1,4	1,68	2,48
2.	С, НР	3	5	3,8	4,56	6,73
3.	С, НР	2	3	2,4	2,88	4,25
4.	С	4	5	4,4	5,28	7,80
5.	С	2	5	3,2	3,84	5,67
6.	С	5	6	5,4	6,48	9,57
7.	С	6	8	6,8	8,16	12,06

8.	С	5	7	5,8	6,96	10,28
9.	С, НР	2	4	2,8	3,36	4,96
10.	С	4	5	4,4	5,28	7,80
11.	С, НР	10	12	10,8	12,96	19,15

График строится в виде таблицы с разбивкой по декадам (10 дней) за период времени выполнения научного проекта. При этом работы на графике следует выделить различным цветом, в зависимости от исполнителей, ответственных за ту или иную работу.

Таблица 18. Календарный план-график проведения НИОКР по теме

№	Вид	Исп	Т раб, дн.	Продолжительность														
				Янв.			Февр.			Март			Апр.			Май		
				1	2	3	1	2	3	1	2	3	1	2	3	1	2	3
1	Выбор научного руководителя работы	С	4,16															
2	Составление и утверждение темы работы	С, НР	11,29															
3	Составление календарного плана-графика выполнения работы	С, НР	7,13															
4	Подбор и изучение литературы по теме работы	С	13,08															
5	Анализ предметной области	С	9,51															
6	Проектирование схемы БД	С	16,05															
7	Разработка серверной части системы	С	20,22															
8	Разработка клиентской части системы	С	17,24															

9	Согласование выполненной работы с научным руководителем	С, НР	8,32																
10	Выполнение других частей работы	С	13,08																
11	Выполнение других частей работы, подведение итогов, оформление работы	С, НР	32,11																

– Руководитель(Р)

– Студент (С)

Итого руководитель потратит на проект около 5 дней, а студент – 112 дня.

Обратимся к расчету бюджета вариантов исполнения проекта.

Представим результат расчета затрат на специальное оборудование для научных (экспериментальных) работ. Для определения срока полезного использования был использовано Постановление Правительства РФ 01.01.2002. Амортизация для программного обеспечения не рассчитывается.

Таблица 19. Специальное оборудование для первого исполнения

Наименование	Единица измерения	Количество	Цена за ед., руб.	Затраты на материалы, руб.	Срок полезного использования	Амортизация
Ноутбук Lenovo Legion Y540-15IRH	шт	1	64000	64000	5	21300
Microsoft Windows 10 Pro	шт	1	11900	11900	1	11900
Пакет Microsoft Office 2019 Home and Student RU x32/x64	шт	1	6999	6999	1	6999
Итого				82899		40199

Таблица 20. Специальное оборудование для второго исполнения

Наименование	Единица измерения	Количество	Цена за ед., руб.	Затраты на материалы, руб.	Срок полезного использования	Амортизация
Другой ноутбук	шт	1	33000	33000	5	6600
Microsoft Windows 10 Домашняя	шт	1	10590	10590	1	10590
Пакет Microsoft Office 2019 Home and Student RU x32/x64	шт	1	6999	6999	1	6999
Итого				50589		24189

В таблице X представим стоимость всех материалов, используемых при разработке проекта.

Тариф за 1 кВт электроэнергии равен 4,04. Запланируем затраты на печать отчетов, в среднем по г.Томску 1 страница – 1 руб.

К данному виду затрат относятся затраты на электроэнергию. Для юридических лиц стоимость 1 кВт·ч составляет 4,04 рублей. При умеренном пользовании ноутбук средней мощности потребляет 100 Вт в час в среднем. В день на работу затрачивается 6 часов, всего на работу с компьютером и оборудованием затрачивается 34 дня у студента и 9 дней у руководителя. Тогда затраты на электроэнергию составят:

$$З_{эн} = 100 \cdot (4{,}04/1000) \cdot 6 \cdot 43 = 104{,}23 \text{ руб.}$$

Таблица 21. Материальные затраты

Наименование	Единица измерения	Количество	Цена за ед., руб.	Затраты на материалы, руб.
Элекроэнергия	кВт	104	4,04	420,16
Бумага	-	120	1	120
			Итого	540,16

Электроэнергия рассчитывается на год.

Основная заработная плата руководителя рассчитывается на основании

отраслевой оплаты труда (оклад, стимулирующие выплаты, районный коэффициент). Величина расходов по заработной плате определяется исходя из трудоемкости выполняемых работ и действующей системы окладов и тарифных ставок. В состав основной заработной платы включается премия, выплачиваемая ежемесячно из фонда заработной платы в размере 20 –30 % от тарифа или оклада. Для студента-дипломника основную заработную плату составляет государственная стипендия с учетом районного коэффициента.

Для расчета основной заработной платы необходимо привести действительный годовой фонд рабочего времени руководителя и студента

Таблица 22. Баланс времени

Показатели рабочего времени	Руководитель	Студент-дипломник
Календарное число дней	365	365
Количество нерабочих дней (выходные дни и праздничные дни)	52 и 10	108 и 10
Потери рабочего времени (отпуск, больничные), дни	56	24
Действительный годовой фонд рабочего времени, дни	247	223

Статья включает основную заработную плату работников, непосредственнозанятых выполнением НТИ, и дополнительную заработную плату:

$$З_{зп} = З_{осн} + З_{доп}, \quad (6)$$

где $З_{осн}$ – основная заработная плата;

$З_{доп}$ – дополнительная заработная плата (10% от основной).

Основная заработная плата руководителя (лаборанта, инженера) от предприятия рассчитывается по следующей формуле:

$$З_{осн} = З_{дн} \cdot T_р \quad (7)$$

где $З_{осн}$ – основная заработная плата одного работника;

$З_{дн}$ – среднедневная заработная плата работника, руб.;

$T_р$ – продолжительность работ, выполняемых работником, раб.дн.

Среднедневная зарплата рассчитывается по формуле:

$$З_{дн} = \frac{Зм \cdot M}{F} \quad (8)$$

где $З_м$ – месячный должностной оклад работника, руб.;

M – количество месяцев работы без отпуска в течение года: при отпуске в 56 раб. дней М = 10,4 месяца, 6-дневная неделя;

$F_д$ – действительный годовой фонд рабочего времени научно-техническогоперсонала, раб. дн., равный 251.

Месячный должностной оклад работника:

$$З_м = З_{тс} \cdot k_р \quad (9)$$

где $З_{тс}$ – заработная плата по тарифной ставке, руб.;

$k_р$ – районный коэффициент, равный 1,3 для Томска.

Затраты по дополнительной заработной плате исполнителей темы учитываютвеличину предусмотренных Трудовым кодексом РФ доплат за отклонение от нормальных условий труда, а также выплат, связанных с обеспечением гарантий и компенсаций (при исполнении государственных и общественных обязанностей, при совмещении работы с обучением, при предоставлении ежегодного оплачиваемого отпуска и т.д.).

Расчет дополнительной заработной платы ведется по следующей формуле:

$$З_{доп} = k_{доп} \cdot З_{осн} = 0{,}15 \cdot 109{,}476 = 16{,}421$$

где $З_{осн}$ – основная заработная плата;

$k_{доп}$ – коэффициент дополнительной заработной платы на стадии проектирования,принимается равным 0,15.

Расчет основной и дополнительной платы руководителя представлен в таблице 23.

Таблица 23. Расчет основной и дополнительной заработной платы

Исполнитель	$З_{ОК}$, руб.	$к_{доп}$	$З_М$, руб	$З_{ДН}$, руб.	$Т_Р$, раб. дн.	$З_{ОСН}$, руб.	$З_{ДОП}$	Всего ($З_{ОСН}$ + $З_{ДОП}$)

Руководитель	37700	1,3	49010	2065,6	53	109,476	16,421	125,897
Инженер-исследователь	23800	1,3	30940	1495,24	139	207,838	31,175	239,013

Представим расчет отчислений во внебюджетные фонды (страховые отчисления), установленные законодательством Российской Федерации нормам органами государственного социального страхования (ФСС), пенсионного фонда (ПФ) и медицинского страхования (ФФОМС) от затрат на оплату труда работников.

Величина отчислений во внебюджетные фонды определяется исходя из следующей формулы:

$$З_{внеб} = k_{внеб} \cdot (З_{осн} + З_{доп}) \quad (10)$$

где $k_{внеб}$ – коэффициент отчислений на уплату во внебюджетные фонды (пенсионный фонд, фонд обязательного медицинского страхования и пр.).

Для учреждений, осуществляющих образовательную и научную деятельность в 2022г., водится пониженная ставка 30% (ст. 425 Закона 117- ФЗ).

Расчет отчислений во внебюджетные фонды приведен в таблице х.

Таблица 24. Отчисления во внебюджетные фонды

Исполнитель	Руководитель	Инженер-исследователь
Всего заработная плата, руб.	125897	239013
Коэффициент отчислений во внебюджетные фонды	30	
Сумма отчислений	37,769	71,703

Накладные расходы учитывают прочие затраты организации, не попавшие в предыдущие статьи расходов: печать и ксерокопирование материалов исследования, оплата услуг связи, электроэнергии, почтовые и телеграфные расходы, размножение материалов и т.д. Их величина определяется по следующей формуле:

$$З_{накл} = k_{накл} \cdot (З_{осн} + З_{доп}) \quad (11)$$

где $k_{накл}$ – коэффициент накладных расходов, 16 %.

Получим:

$$З_{накл} = 0,16 \cdot (109,476 + 16,421) = 20,143 \text{ руб.}$$

Результат определения бюджета затрат на научно-исследовательский проект был рассчитан на два исполнения, с инженером-исследователем и без него. Результаты приведены таблице 25.

Таблица 25. Величина затрат научно-исследовательской работы

Наименование статьи	**Исполнение 1.**	**Исполнение 2**
1. Специальное оборудование и ПО, в тыс. руб.	40,199	24,189
2. 2. Сырье и материалы (электроэнергия), в руб.	540,16	
3. Основная заработная плата исполнителей проекта, в тыс. руб.	109,476	207,838
4. Дополнительная заработная плате исполнителей проекта, в тыс. руб.	16,421	31,175
5. Отчисления во внебюджетные фонды, в тыс. руб.	37,769	71,703
6. Накладные расходы, в тыс. руб.	20,143	38,242
Бюджет затрат НТИ	224548,16	373687,16

5.3 Определение ресурсной (ресурсосберегающей), финансовой, бюджетной, социальной и экономической эффективности исследования

Определение эффективности происходит на основе расчета интегрального показателя эффективности научного исследования. Его нахождение связано с определением двух средневзвешенных величин: финансовой эффективности и ресурсоэффективности.

Интегральный финансовый показатель разработки определяется как:

$$I^{\text{испi}} = \underline{\Phi_{\text{pi}}} / \Phi_{\text{max}} \qquad (12)$$

фи

где $I^{\text{испi}}$ – интегральный финансовый показатель разработки;

Φ_{p_i} – стоимость -го варианта исполнения;

Φ_{max} – максимальная стоимость исполнения научно-исследовательского проекта.

Φ_{max} зависит от сложности проекта для которого разрабатывается ПО.

Т.к. стоимость всех вариантов исполнения одинакова, интегральные финансовые показатели также будут одинаковы и равны 1.

Интегральный показатель ресурсоэффективности вариантов исполнения объекта исследования можно определить следующим образом:

$$Ipi = \sum ai \cdot bi \,, \qquad (13)$$

где I_{p_i} – интегральный показатель ресурсоэффективности для n-го варианта исполнения разработки;

a_i – весовой коэффициент i-го варианта исполнения разработки;

b^a, b^p – бальная оценка -го варианта исполнения разработки, устанавливается экспертным путем по выбранной шкале оценивания.

Интегральный показатель эффективности вариантов исполнения разработки определяется на основании интегрального показателя ресурсоэффективности и интегрального финансового показателя по формулам:

$$I_{\text{исп1}} = I_{p\text{-исп1}} / I^{\text{испi}}_{\text{фи}} \qquad (14)$$

$$I_{\text{исп2}} = I_{p\text{-исп2}} / I^{\text{испНр}}_{\text{фи}}$$

Так как интегральные финансовые показатели одинаковы и равны 1, то интегральные показатели эффективности вариантов исполнения разработки равны соответствующим интегральным показателям ресурсоэффективности.

Сравнение интегрального показателя эффективности вариантов исполнения разработки позволит определить сравнительную эффективность проекта и выбрать наиболее целесообразный вариант из предложенных.

Сравнительная эффективность проекта:

$$Э_{\text{ср}} = I_{\text{испi}} / I_{\text{исп1}} \qquad (15)$$

Таблица 26. Оценка характеристик вариантов исполнения проекта

Критерии	Весовой коэффициент	Исп.1	Исп.2
1.Надежность	0,2	5	4
2.Производительность оборудования	0,3	5	3

3.Объем выполняемых работ	0,2	3	5
4.Срок разработки	0,1	4	4
5.Техническая поддержка платформы	0,2	4	5
I_{p^i}		4,3	4,1

На основании полученных показателей выполним сравнение интегрального показателя эффективности вариантов исполнения разработки (табл. х)

Таблица 27. Сравнительная эффективность разработки

Показатели	Исп.1	Исп.2
Интегральный финансовый показатель разработки	0,6	1
Интегральный показатель ресурсоэффективности разработки	4,3	4,1
Интегральный показатель эффективности	7,16	4,1
Сравнительная эффективность вариантов исполнения	1,7	

5.4 Вывод

Таким образом, полученная величина интегрального финансового показателя по исполнению 1 отражает удешевление бюджета затрат так как полученный коэффициент меньше единицы, но больше нуля. Интегральный показатель по второму исполнению указывает на увеличение бюджета затрат разработки.

Итак, полученная величина интегрального показателя ресурсной эффективности выше по первому исполнению. Характеризуя исполнение первое, было установлено следующее. Во-первых, надежность исполнения первого повышается за счет обеспечения отказоустойчивости программных средств разработки с помощью используемой Microsoft Windows 10 Pro. Во-вторых, в первом исполнении запланировано использовать ноутбук Lenovo Legion Y540, который мощнее по ряду параметров чем ноутбук во втором исполнении. Это позволит сократить время на выполнение ряда этапов разработки. Отметим, что, испол-

нение первое уступает исполнению второму по ряду критериев. Объясняется это тем, что запланированное количество исполнителей позволяет выполнить больший объем работ (по дополнительным требованиям от Заказчика), и, обеспечит непрерывность технической поддержки системы.

Сравнительная эффективность вариантов исполнения показала, что первый вариант исполнения наиболее выгодный. Данный вариант исполнения и используется в выпускной квалификационной работе.

6. Социальная ответственность

В результате выполнения выпускной квалификационной работы, будет разработана система, используемая для проведения соревнований по управленческой борьбе, ведения учета проведенных соревнований, ведения рейтинга участников и отображения предстоящих и прошедших турниров. Данная система представляет из себя веб приложение, доступ которому можно получить через интернет браузер.

Предполагается, что в качестве места работы будет выбран компьютерный класс в Кибернетическом центре ТПУ. Основными применяемыми средствами работы являются персональный компьютер и сеть Интернет.

В данном разделе будут обозначены и рассмотрены опасные и вредные факторы, оказывающие влияние на производственную деятельность исполнителя работы, факторы воздействия разрабатываемой информационной системы на окружающую среду, как правовые и организационные вопросы, так и вопросы связанные с мероприятиями в чрезвычайных ситуациях.

6.1 Правовые и организационные вопросы обеспечения безопасности

Трудовой кодекс РФ осуществляет регулирование отношений между организацией и сотрудниками, регулирование требований к продолжительности рабочего времени и оплаты труда [1].

Продолжительность времени потраченного на работу не должно превышать 40 часов в неделю. Трудовой договор устанавливает продолжительность рабочего времени и размер оплаты труда для конкретного работника.

Проведение разработки на персональном компьютере требует соблюдение правовых норм к работе на персональном компьютере и организации рабочего места [2, 3, 4]. Следующие требования должны быть соблюдены при организации рабочего места должны:

– Требования к персональному компьютеру;

– Требования к помещениям для работы с персональным компьютером;

– Требования к микроклимату, содержанию аэроионов и вредных химических веществ в воздухе на рабочих местах, оборудованных персональными компьютерами;

– Требованияк уровням шума и вибрации на рабочих местах, оборудованных персональными компьютерами;

– Требования к освещению на рабочих местах, оборудованных персональными компьютерами;

– Требования к уровням электромагнитных полей на рабочих местах, оборудованных персональными компьютерами;

– Общие требования к организации рабочих мест пользователей персональных компьютеров;

– Требованияк организации медицинского обслуживания пользователей персональных компьютеров;

– Требованияк проведению государственного санитарно-эпидемиологического надзора и производственного контроля.

– Разработка и эксплуатация программных средств, осуществляется на персональных компьютерах с соблюдением всех вышеперечисленных требований.

6.2 Производственная безопасность

При выполнении работ могут возникнуть вредные и опасные факторы. Данные факторы были выявлены согласно ГОСТ 12.0.003- 2015 и приведены в таблице 28 [5].

Таблица 28. Опасные и вредные факторы при выполнении работ

Факторы (ГОСТ 12.0.003-2015)	Этапы работы		Нормативные документы
	Разр.	Эксп.	
1. Повышенный уровень шума нарабочем месте	+	+	Шум. Общие требования безопасности,СН 2.2.4/2.1.8.562–96; [6]
2.Отклонение показателей микроклимата	+	+	СанПиН 2.2.3670-20 «Санитарно-эпидемиологические требования к условиям труда» [7] СанПиН 2.2.1/2.1.1.1278-03 «Гигиенические требования к микроклимату производственных помещений» [8]
3. Недостаточная освещен-	+	+	СанПиН 1.2.3685-21 «Гигиениче-

ность рабочей зоны			ские нормативы и требования к обеспечению безопасности и (или) безвредности для человека факторов среды обитания» [9]
4. Повышенное образование электростатических зарядов	+	+	СанПиН 2.2.2.542-96 «Гигиенические требования к видеодисплейным терминалам, персональным электронно-вычислительным машинам и организации работ» [10]

6.2.1 Повышенный уровень шума на рабочем месте

Шум является одним из ключевых вредных факторов на компьютеризированных рабочих местах. Чаще всего источники шума это вентиляторы системного блока, накопители, принтеры. Шум оказывает нагрузку на нервную систему человека, оказывая на него психологическое воздействие и вызывая изменения в органах слуха человека.

Согласно СН 2.2.4/2.1.8.562–96 [11], допустимым уровнем звука на рабочих местах является 50 дБА. Нормальная вентиляция системного блока минимизирует шум.

6.2.2 Отклонение показателей микроклимата рабочей зоны

Оптимальное соотношение температуры и влажности воздуха определяют уровень комфорта при выполнении работ. При работе в помещении с плохим сочетанием температуры и влажности возникает возможность развития инфекций, обострения аллергических заболеваний и астмы.

В соответствии с СанПиН 2.2.1/2.1.1.1278-03, в рабочих местах пользователей персональных компьютеров должны быть оптимальные показатели микроклимата [8]. Требования, подлежащие соблюдению описаны в таблице 29.

Таблица 29. Оптимальные параметры микроклимата впроизводственных помещениях пользователя ПК

Периодгода	**Температура воздуха, °С**	**Температура поверхностей, °С**	**Относительная влажность,%**	**Скорость движения воздуха, м/с**
Холодный	22-24	21-25	60-40	0,1

Теплый	23-25	22-26	60-60	0,1

Основными средствами поддержания необходимых значений микроклимата являются системы отопления и кондиционирования воздуха и тепловая изоляция нагретых поверхностей оборудования. В результате исследования микроклимата выявилось соответствие показателей микроклимата требованиям, указанным в СанПиН.

6.2.3 Отсутствие или недостаток естественного света, а также недостаточная освещенность рабочей зоны

Для снижения утомляемости и повышения производительности труда необходимо подходящее производственное освещение. Основными искусственными и естественными источниками света в компьютерных залах являются люминесцентные лампы типа ЛД и прямые солнечные лучи.

Для искусственного освещения в компьютерных залах должна быть использована система общего равномерного освещения. Местное освещение не должно создавать бликов на поверхности экрана и увеличивать освещенность экрана более 300 лк, согласно СанПиН 1.2.3685-21 «Гигиенические нормативы и требования к обеспечению безопасности и (или) безвредности для человека факторов среды обитания» [12].

Коэффициент естественной освещенности (КЕО) и минимальные допустимые значения КЕО применяются для оценки используемого естественного света. КЕО представляет собой отношение освещенности Ев внутри помещения за счет естественного света к наружной освещенности Ен от всей полусферы небосклона, выраженное в процентах:

$$КЕО = (E_{в}/E_{н})/100\% \qquad (16)$$

При боковом естественном освещении в аудитории, лаборатории на рабочих столах и партах должен обеспечиваться КЕО = 1,5 %. К средствам нормализации освещения производственных помещений и рабочих мест относятся: источники света, осветительные приборы, световые проемы, светозащитные

устройства, светофильтры

6.2.4 Повышенное образование электростатических зарядов

Суть электризация заключается становлении нейтральных тел, в нормальном состоянии не проявляющих электрических свойств, электрозаряженными при условии отрицательных контактов. Опасность возникновения статического заряда электричества заключается в риске появления электрической искры и вредном воздействии его на человеческий организм как при непосредственном контакте с зарядом, так и за счет действий электрического поля появляющимся при заряде. Статическое электричество накапливается на экране дисплея при включенном питании компьютера. Несмотря на то, что электрический ток искрового разряда статического электричества не может напрямую вызвать поражение человека, электризующаяся пыль оседает на экране и искажает резкость восприятия информации на экране а также может попадать на лицо работающего и в его дыхательные пути.

Заземление оборудования, увлажнение окружающего воздуха и полы из антистатического материала являются основными способами защиты от статического электричества.

Перед тем как начать выполнение работ, стоит убедиться в отсутствии свешивающихся со стола или висящих под столом проводов электропитания, в целостности вилки и провода электропитания, в отсутствии видимых повреждений аппаратуры и рабочей мебели, в отсутствии повреждений и наличии заземленияприэкранного фильтра.

Выполнение правил, указанных в «ССБТ. Электробезопасность. Общие требования и номенклатура видов защиты» обезопасит от поражения электрическим током [13].

6.3 Обоснование мероприятий по защите исследователя от действия опасных и вредных факторов

Перечисленные требования безопасности и промышленной санитарии должны быть учтены при организации рабочего пространства и режима труда.

Рабочие места должны быть оснащены системами отопления, вентиля-

ции, кондиционирования воздуха, а шумящее оборудование должно располагаться вне помещения с персональным компьютером.

Рабочее пространство должно иметь хорошее естественное и искусственное освещение. Для защиты пользователей персонального компьютера от негативного воздействия электромагнитных полей необходимо, чтобы используемая техника удовлетворяла нормам и правилам сертификации.

Невыполнение вышеперечисленных рекомендация может привести к повышению уровня воздействия опасных и вредных факторов на работающего.

6.4 Экологическая безопасность

Разработка программного обеспечения и ее использование не оказывают негативного воздействия на окружающую среду в отличии от используемых для разработки и эксплуатации веществ, оказывающих негативное влияние на атмосферу, гидросферу и литосферу. Требования к обращению с отходами описаны в ГОСТ Р 53692-2009 [14].

В случае выхода из строя персонального компьютера, батарейки, оргтехники и прочего оборудования, необходимо произвести утилизацию через специализированные организации, имеющие право на ведение такой деятельности. В случае неправильной утилизации канцелярских товаров, продуктов питания, продуктов личной гигиены и других твердых отходов могут оказываться негативное воздействие на почвенный покров. Чтобы предотвратить загрязнение почвенного покрова необходимо производить сбор, сортировку и утилизацию отходов и их организованное захоронение.

Все отходы, образующиеся в ходе разработки и эксплуатации результатов выпускной квалификационной работы, утилизируются без оказания негативного влияния на окружающую среду.

6.5 Безопасность в чрезвычайных ситуациях

Пожар является наиболее вероятной ЧС в рамках рассматриваемого помещения. В современных персональных компьютерах очень высокая плотность размещения элементов электронных схем и расположения соединительных проводов что может привести к значительному количеству теплоты при подаче

электрического тока. В свою очередь, высокая температура может привести к возникновению возгорания. Возникновение других видов чрезвычайных ситуаций маловероятно.

Необходимо соблюдение техники пожарной безопасности для предотвращения возникновения и распространения пожара. Общие требования к пожарной безопасности описаны в ГОСТ 12.1.004-91 [15].

Ниже перечислены мероприятия, подлежащие выполнению при возникновении пожара:

– В случае возникновения пожара необходимо покинуть помещение в течение минимального времени согласно плану эвакуации.

– Запрещается применение воды и пены в помещениях с компьютерной техникой ввиду опасности повреждения или полного выхода из строя дорогостоящего электронного оборудования.

– Для тушения пожаров необходимо применять углекислотные и порошковые огнетушители, которые обладают высокой скоростью тушения, большим временем действия, возможностью тушения электроустановок, высокой эффективностью борьбы с огнем. Воду разрешено применять только во вспомогательных помещениях.

– Помещение должно быть оборудовано пожарными извещателями, которые позволяют оповестить дежурный персонал о пожаре.

Рисунок 30. План эвакуации

В помещении, где происходила разработка и происходит эксплуатация программных средств, были соблюдены все правила техники безопасности. Сотрудникам были объяснены правила поведения в случае возникновения и распространения пожара.

6.6 Выводы по разделу

В результате анализа условий труда на месте, где была разработана магистерская работа, можно сделать вывод, что помещение удовлетворяет необходимым нормам и в случае соблюдения техники безопасности и правил пользования компьютером работа в данном помещении не приведет к ухудшению здоровья работника. Само помещение и рабочее место в нем удовлетворяет всем нормативным требованиям. Кроме того, действие вредных и опасных факторов сведено к минимуму, т.е. микроклимат, освещение и электробезопасность соответствуют требованиям, предъявленным в соответствующих нормативных документах.

Работу с персональным компьютером можно отнести к экологически безопасным видам деятельности, если должным образом утилизировать отходы

данной деятельности.

Относительно рассмотренного вопроса об экологической безопасности можно сказать, что деятельность в ходе выполнения выпускной квалификационной работы не представляет опасности окружающей среды

6.7 Законодательные акты

1. Трудовой кодекс Российской Федерации от 30.12.2001 N 197- ФЗ (ред. От 30.12.2015).
2. ГОСТ Р 50923-96. Дисплеи. Рабочее место оператора. Общие эргономические требования и требования к производственной среде. Методы измерения.
3. ТОИ Р-45-084-01 Типовая инструкция по охране труда при работе на персональном компьютере.
4. ГОСТ 12.2.032-78 Система стандартов безопасности труда (ССБТ). Рабочее место при выполнении работ сидя. Общие эргономические требования.
5. ГОСТ 12.0.003-2015 Система стандартов безопасности труда (ССБТ). Опасные и вредные производственные факторы
6. Шум. Общие требования безопасности, СН 2.2.4/2.1.8.562–96;
7. СанПиН 2.2.3670-20 «Санитарно- эпидемиологические требования к условиям труда»
8. СанПиН 2.2.1/2.1.1.1278-03 «Гигиенические требования к микроклимату производственных помещений»
9. СанПиН 1.2.3685-21 «Гигиенические нормативы и требования к обеспечению безопасности и (или) безвредности для человека факторов среды обитания»
10. СанПиН 2.2.2.542-96 «Гигиенические требования к видео-дисплейным терминалам, персональным электронно-вычислительным машинам и организации работ»
11. СН 2.2.4/2.1.8.562-96. Шум на рабочих местах, в помещениях жилых, общественных зданий и на территории жилой застройки;
12. СанПиН 1.2.3685-21 «Гигиенические нормативы и требования к обеспечению безопасности и (или) безвредности для человека факторов среды обита-

ния»
13. ГОСТ 12.1.019-2017 «Электробезопасность. Общие требования и номенклатура видов защиты»
14. ГОСТ Р 53692-2009. Ресурсосбережение. Обращение с отходами. Этапы технологического цикла отходов. – М.: ИПК Издательство стандартов, 2001 год.
15. ГОСТ 12.1.004-91. ССБТ Пожарная безопасность. Общие требования. – М.: ИПК Издательство стандартов, 2001.

ЗАКЛЮЧЕНИЕ

В результате исследовательской работы были изучены современные подходы и архитектурные шаблоны, применяемые в разработке веб-приложений. Был изучен подход предметно-ориентированного проектирования, который позволяет проектировать модульные и расширяемые системы благодаря своим принципам написания программного кода нацеленных на тесную взаимосвязь с предметной областью Заказчика программного обеспечения. При разработке данный подход требует использования чистой архитектуры кодовой базы, которая подразумевает отделение модулей, связанных с бизнес-логикой, от прочих модулей кода, реализующих технические функции системы. Кроме того, применение шаблона CQRS позволило разработать систему, которая оказалось более производительной при запросе данных из базы данных. Для разработки системы были использованы фреймворки разработки серверных сервисов и клиентских приложении.

Для анализа расширяемости кодовой базы системы, были проанализированы система, написанная без применения DDD и система, написанная по DDD. В качестве инструмента статического анализа был использован статический анализатор SonarQube. В ходе анализа выяснилось, что код, написанный согласно принципам DDD при проектировании информационной системы для Клуба управленческой борьбы, является более расширяемым и имеет низкий уровень технического долга.

СПИСОК ПУБЛИКАЦИЙ СТУДЕНТА

1. Мухтар Д.Р. ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ ОПТИЧЕСКОГО РАСПОЗНАВАНИЯ СИМВОЛОВ ДЛЯ МОБИЛЬНЫХ УСТРОЙСТВ С ПРИМЕНЕНИЕМ ОБЛАЧНЫХ ВЫЧИСЛЕНИЙ // XIX Международная научно-практическая конференция студентов, аспирантов и молодых ученых, Молодежь и современные информационные технологии, Томск, 21–25 марта 2022 г.

ЛИТЕРАТУРА

1. Понятие информационной системы (ИС): основные термины и определения. [Электронный ресурс] URL: https://cde.osu.ru/courses2/course157/text/1.5.htm (дата обращения 04.10.2020).
2. Информационные системы [Электронный ресурс] URL: http://dit.isuct.ru/IVT/BOOKS/IS/IS1/inform/glaves/glava3/gl_3_1.htm (дата обращения 04.10.2020).
3. О Федерации управленческой борьбы [Электронный ресурс] URL: http://www.poedinki.ru/about/ (дата обращения 04.10.2020).
4. Технология проведения управленческих поединков [Электронный ресурс] URL: http://www.poedinki.ru/about/ (дата обращения 04.10.2020). http://www.poedinki.ru/skill/technology/
5. Эрик Еванс. Предметно-ориентированное проектирование (DDD). Структуризация сложных программных систем. – 2020. – 448 с.
6. Vlad Khononov, Learning Domain-Driven Deseign. Aligning Software Architecture and Business Strategy. – 2021. – 446 с.
7. Ми Роберт, Фаулер Мартин, Шаблоны корпоративных приложений. – 2020 – 544
8. Cesar de la Torre, Bill Wagner, Mike Rousos. NET Microservices Architecture for Containerized NET Applications – 2022. – 335 с.
9. Greg Young, Mani Subramanian, Fernando Simonazzi. Exploring CQRS and Event Sourcing: A journey into high scalability, availability, and maintainability with Windows Azure – 2013. – 376 с.
10. Бертран Мейер. Объектно-ориентированное конструирование программных систем. – 2005. – 1204 с.
11. Введение в CQRS, URL: https://docs.microsoft.com/en-us/azure/architecture/patterns/cqrs
12. Dmitri Korotkevitch. PRO SQLServer Internals. //Apress. – 2016. – 793 с.
13. Jon P Smith. Entity Framework Core in Action. //Manning. – 2018. – 471 с.

14. Выполняемые на сервере веб-фреймворки. URL: https://developer.mozilla.org/ru/docs/Learn/Server-side/First_steps/Web_frameworks

15. Адам Фримен, Стивен Сандерсон. ASP.NET MVC 3 Framework с примерами на C# для профессионалов. //Вильямс – 2011. – 672 с.

16. Paul Halliday. Vue.js 2 Design Patterns and Best Practices. //Packt. – 2018 — 421 с.

17. Эрик Хэнчетт, Бенджамин Листуон. Vue.js в действии. //СПб Питер. – 2019 – 304 с.

18. Джеффри Рихтер. CLR via C#. Программирование на платформе Microsoft .NET Framework 4.5 на языке C#. 4-е изд. //СПб Питер. – 2022. – 896с.

19. James Lee, Tao Wei, Suresh Kumar Mukhiya. Redux Quick Start Guide. //Packt. – 2019. – 204 с.

20. James Kolce, Mark Brown, Craig Buckler, Michael Wanyoike, Nilson Jacques. Modern JavaScript Tools & Skills. //SitePoint. – 2018. – 136 с.

21. Статический анализ кода. URL: https://dic.academic.ru/dic.nsf/ruwiki/129701

22. Технический долг, Стив МакКонелл, URL: https://www.construx.com/resources/whitepaper-managing-technical-debt/

23. Что такое облачные вычисления, URL: https://www.oracle.com/ru/cloud/what-is-cloud-computing/

24. Развертывание приложения ASP.NET Core с подключением к Базе данных SQL Azure в Службе приложений Azure, URL: https://docs.microsoft.com/ru-ru/azure/app-service/tutorial-dotnetcore-sqldb-app?tabs=azure-portal%2Cvisualstudio-deploy%2Cdeploy-instructions-azure-portal%2Cazure-portal-logs%2Cazure-portal-resources

ПРИЛОЖЕНИЕ А

(обязательное)

Раздел ВКР, выполненный на иностранном языке

Раздел 2 Design and development of information management system

Студент:

Группа	ФИО	Подпись	Дата
8ИМ02	Мухтар Д.Р.		

Руководитель ВКР:

Должность	ФИО	Ученая степень, звание	Подпись	Дата
Доцент	Мирошниченко Е.А.	к.т.н.		

Консультант – лингвист отделения (НОЦ) школы ОИЯ, ШБИП:

Должность	ФИО	Ученая степень, звание	Подпись	Дата
Доцент	Сидоренко Т.В.	к.п.н.		

Design and development of information management system

Database schema design

The database consists of reference and main (filled with data about participants and tournaments) tables: reference tables are the tables named “Roles”, “Tournament format”, “Tournament type”, “Tournament grid”. The rest of the tables are the main ones.

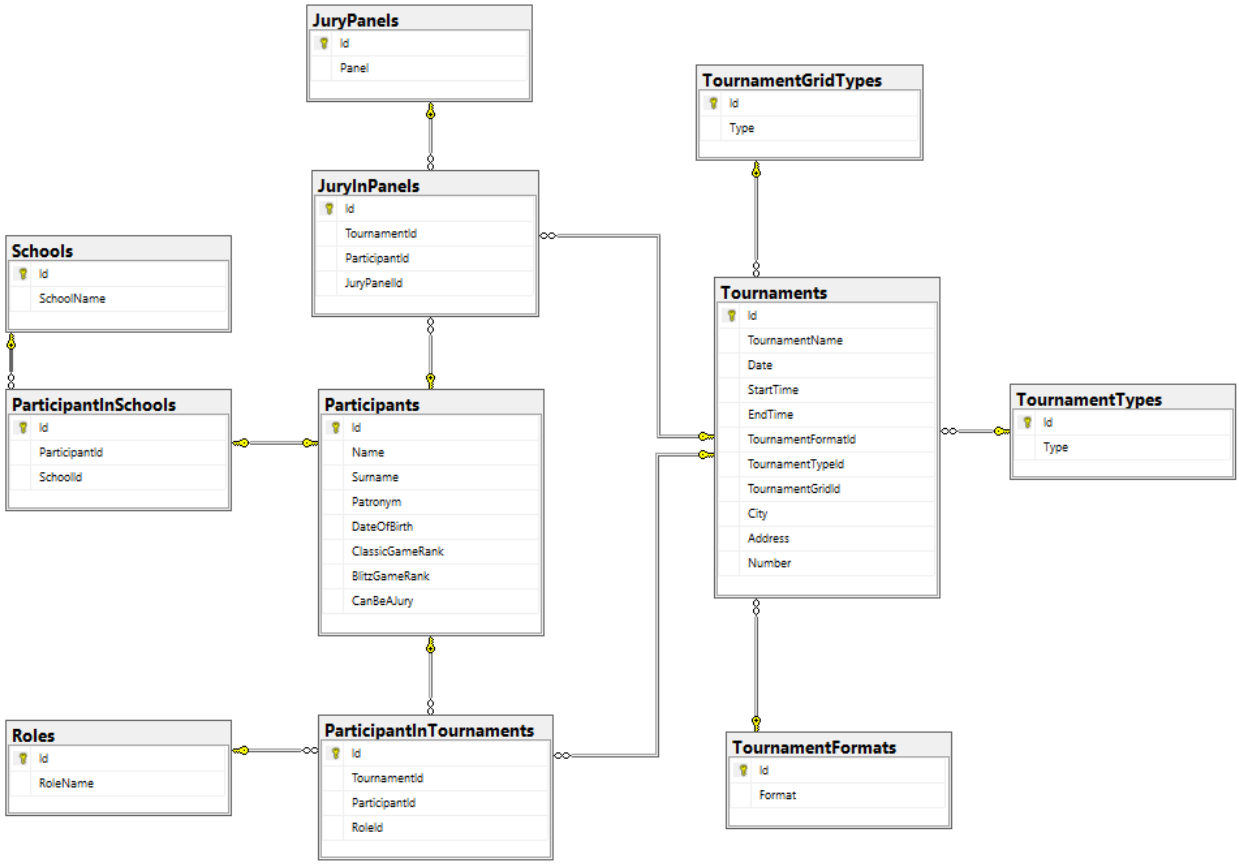

Fig. 1. Database schema

The database schema is subdivided into two aggregate clusters [5]. According to DDD, aggregate is cluster of domain objects that can be considered as a whole. The goal of aggregates is to maintain data consistency by denoting the boundaries within which data is modified in order to avoid violations in business logic. In the case of the designed data schema, we have two main clusters of aggregates:

– tables “Participant in schools”, “Schools”, “Participant in tournament”, “Roles”, “Judges”, “Judges' panels”, “Judges in panels”;

– tables “Tournaments”, “Types of Tournament Brackets”, “Types of Tournament”, “Tournament Formats”.

An aggregate cluster must have a root object (aggregate root), through which all other cluster objects are accessed and data change commands are executed [7]. Any work with aggregate cluster objects should be done only using the root aggregate. Thus, the root can ensure the data consistency of its aggregates. The aggregate is the main element of the data transfer and transactions should not cross the boundaries of another cluster of aggregates. Following figure illustrate the root objects of two aggregate clusters. These objects will contain methods related to business logic.

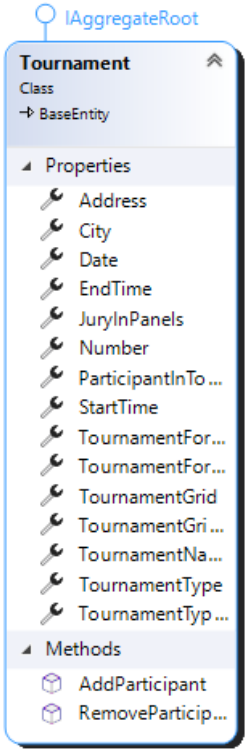

Fig. 4. Main aggregate roots

Using clean architecture in the development

The clean architecture was chosen as the main application architecture. The source code consists of the Web project, which depends on three libraries, SharedKernel, Infrastructure and Core. The Infrastructure library is responsible for the implementation of data transfer with database and depends on the Core library, since this library stores classes of domain entities and their relationships, according to which tables will be created in the database. The SharedKernel library includes Data

Transfer Objects (DTOs) and other models required for all layers. For clarity, the following illustration is given.

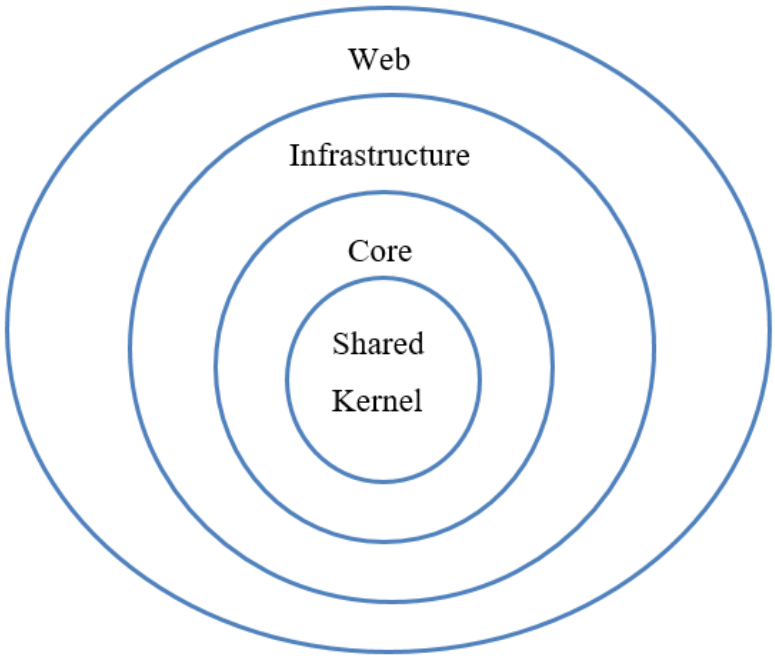

Fig. 5. Clean architecture layers illustrated

The Core library contains classes that implement the business logic of the system. For example, to register a participant in a tournament, the necessary data is requested at the “Infrastructure” abstraction layer and then transferred to the lower level where validation and other operations related to business logic take place. Thus, the code that implements the business logic does not depend on the ORM library used and other technical details and dependencies.

To implement this architecture, standard C# class libraries were created in the codebase for each layer and dependencies were added according to the principle of the architecture itself, that is, the Web project depends on the Infrastructue and Core libraries, and the Infrastructure library, in turn, depends on the Core library. At the same time, all libraries have access to the SharedKernel library. Below is the project tree in the IDE [18].

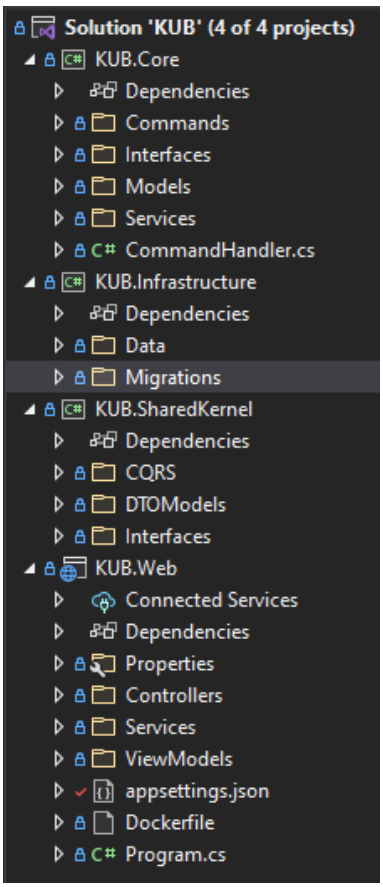

Fig. 6. Project tree in IDE

Implementation of CQRS

In order to implement this pattern, the interfaces of repositories and services were written in the Core library. The classes which implement the repository interfaces have been added to the Infrastructure library because they are not abstract and may change in the future. At the same time, the class that implements the IService interface has been added to the Core library since it's not supposed to change. Below is a table with the classes and their descriptions.

Table 1. Description of classes

No	Interface	Inheriting class	Purpose
7.	IBaseCommandHandler	CommandHandler	Responsible for all operations related to adding, changing and deleting data. Encapsulates classes that implement the IEventRepository

			and IBaseWriteRepository interfaces
8.	IEventRepository	EventRepository	Responsible for logging events related to adding, changing and deleting data
9.	IService	Service	Executes methods managing data and passes commands to the CommandHandler class and logs events through the EventRepository class.
10.	IUnitOfWork	UnitOfWork	Distributes transactions in repositories for operations related to data manipulation. Encapsulates the class BaseWriteRepository, EventRepository.
11.	IReadRepository	SchoolReadRepository, ParticipantReadRepository, TournamentReadRepository	Contain SQL queries in the form of strings. Request data from a database
12.	IWriteRepository	BaseWriteRepository	Updates, adds, and deletes data

The HTTP request processing algorithm is as follows:

– HTTP requests are processed by the Kestrel web server and wrapped in the HttpContext class. Then objects of the Controller class are created in which the appropriate method for processing the request is called, depending on the URL path. In total, four controllers were written, SchoolsController, TournamentsController and ParticipantsController responsible for working with the entities "Schools", "Tournaments" and "Participants", respectively, as well as AuthController responsible for user authentication and authorization

– If it is a request for data, the controller directly accesses the repository to perform a read operation

– If it is a request to add, change, or delete data, the controller calls the appropriate method in the Service class, which then passes the command to the CommandHandler command handler. The handler calls a method on the WriteRepository object and stores the event in the EventRepository.

The algorithm scheme is shown in Figure 7

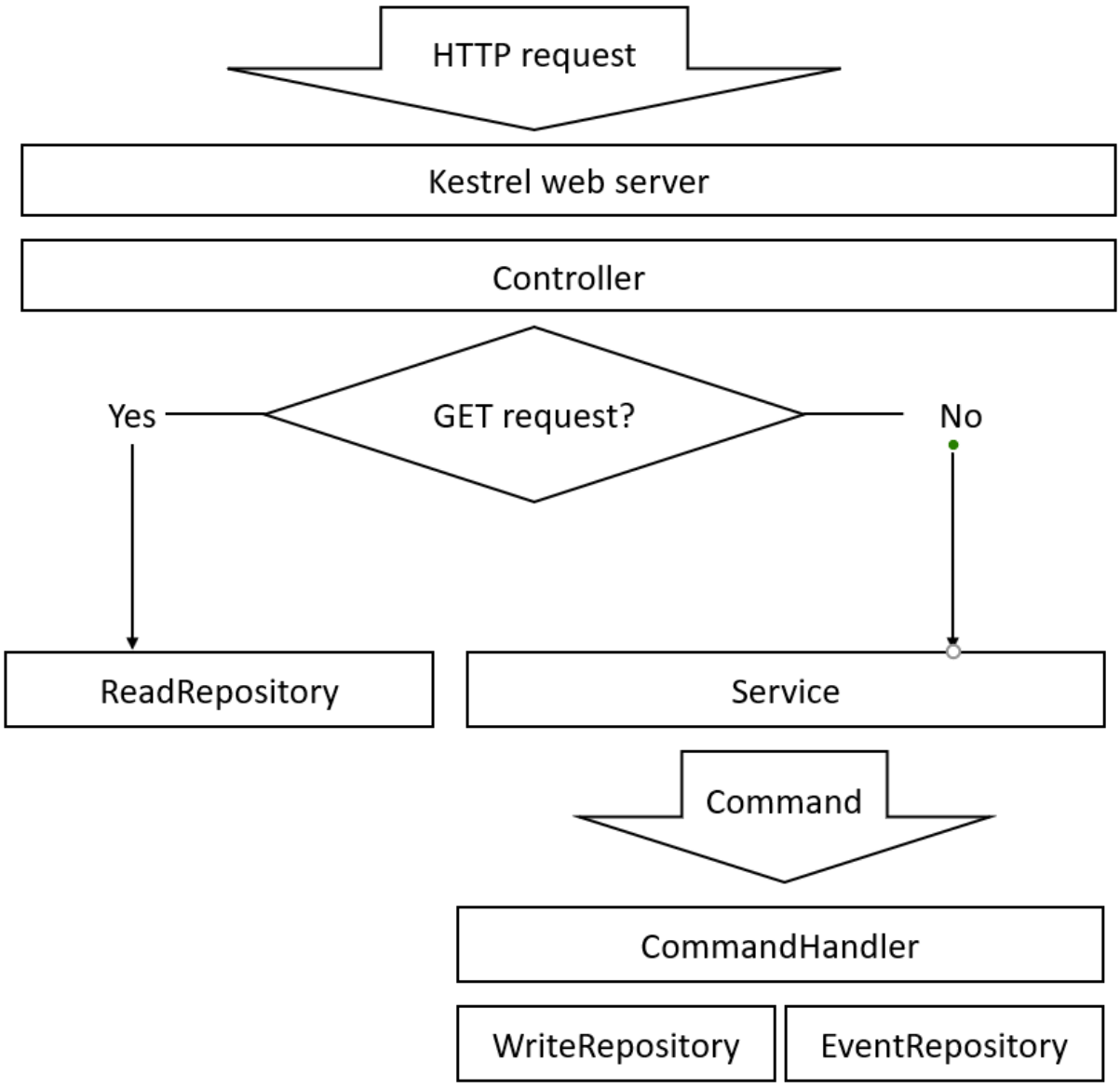

Fig. 7. HTTP request handling algorithm

Web client application development

Components are the basic elements of the code that allow you to subdivide the application into logical blocks. They define the HTML skeleton of the user interface and use Javascript code responsible for any logical operations. Components were created in the web client project folder and subdivided into directories according to their functional tasks and location on web pages. The figure below shows these components in the IDE.

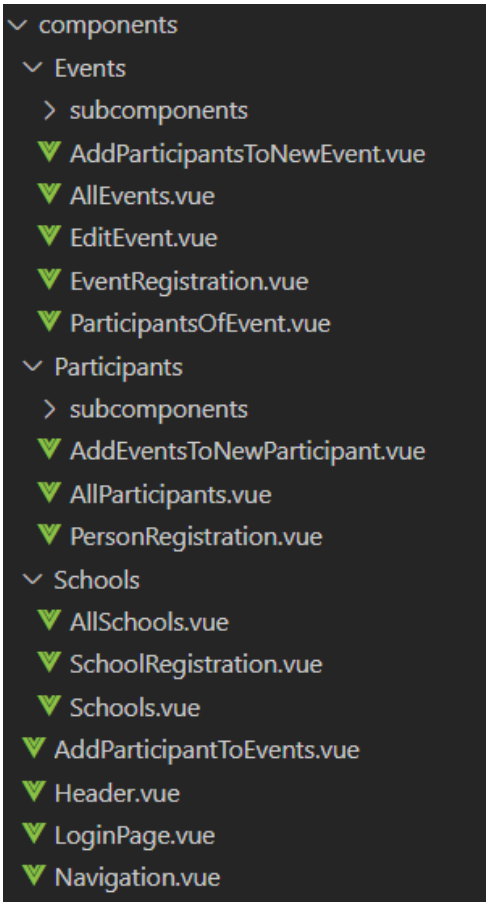

Fig. 8. Component tree

All components can be reused in any part of the web client application. In order to use the created component, you must assign an HTML tag to it and place the assigned tag in another component or HTML element.

Below is an example of creating a Header component.

```
<template>
  <a-layout-header class="header">
    <div class="logo">
      <p style="color: white">Клуб управленческой борьбы</p>
    </div>
    <div class="logout" @click="logout">
      <a-icon type="logout" />
    </div>
  </a-layout-header>
</template>
```

Fig. 9. Header component

Ant Design design component system was used for HTML forms and other elements. Ant Design is a library containing ready-made Vue components such as buttons, icons, form fills and other interface elements [19]. For example, the library has an "a-layout-header" component that was added to the generated Header component, which in turn was added to the App component by being imported from the "./components/Header.vue" directory. Importing and exporting components takes place in a special section of code, indicated by the script tag. In addition to importing and exporting components, this block adds any methods and objects of the Javascript language necessary for any data manipulation in the component's HTML elements. Each component file also contains a style block, which contains CSS classes that set the style for the component elements. Sample code is provided below.

```
<template>
  <div id="app">
    <a-layout style="height: 100vh">
      <Header />
    </a-layout>
  </div>
</template>
<script>
import Header from "./components/Header.vue";
export default {
  name: "App",
  components: {
    Header,
  },
};
</script>
<style>
#app {
  font-family: Avenir, Helvetica, Arial, sans-serif;
  -webkit-font-smoothing: antialiased;
  -moz-osx-font-smoothing: grayscale;
  text-align: center;
  color: #2c3e50;
  margin-top: 0;
}
</style>
```

Fig. 10. App component

One of the challenges in client-side web application development is sharing the state of a Javascript variable from other components. Unlike traditional web applications, SPA applications allow you to save and pass data in variables between components. In the Vue framework, the Vuex state management system is responsible for this. This system was developed under the influence of the Flux project, developed by Facebook. Vuex allows to store the state of the application in the form of a centralized repository and also provide general access to changing storage data through public methods in this repository. When developing the web client, such storage was used as a buffer for temporarily storing data on new registrations, tour-

naments and schools, lists of participants and schools in order to fill in HTML forms when switching to a new page or editing data in the user interface. In addition, the storage has variables that identify the successful dispatch of HTTP requests and, depending on their state, an alert window is displayed. All components have access to this storage, namely its variables and methods, and interact with it if necessary. Below is an illustration describing this relationship [17].

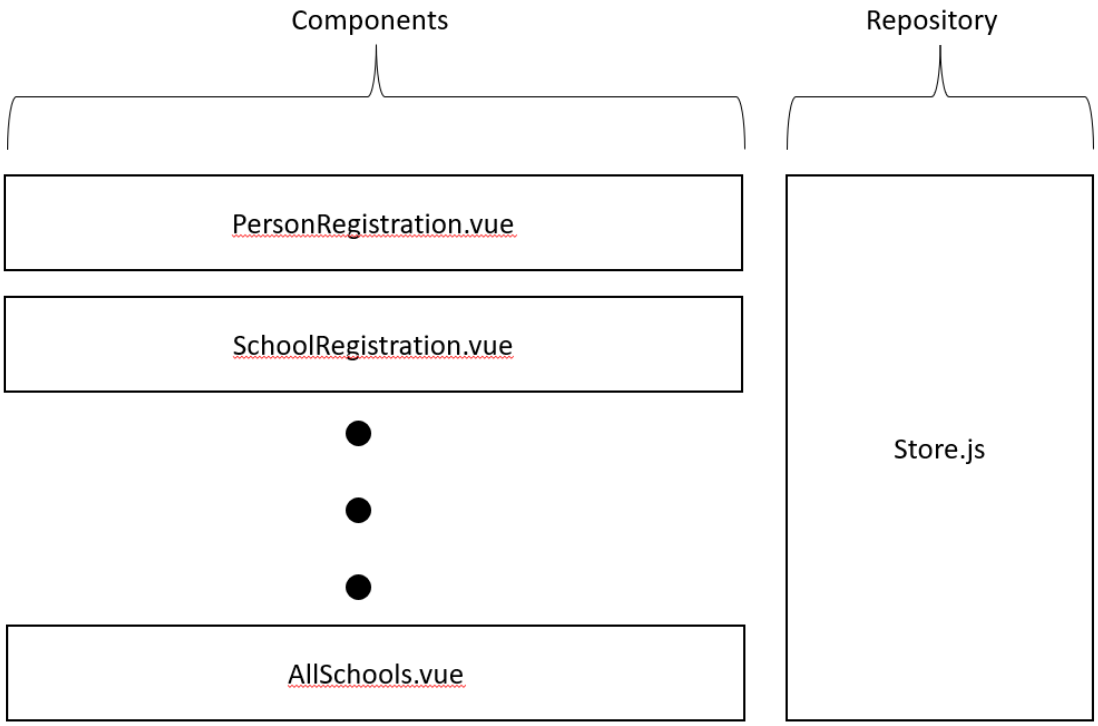

Fig. 11. Components and repository

One of the main tasks of any client web framework is to support data exchange with the server via the HTTP protocol. A library called “axios” based on the asynchrony of the Javascript language was added to the project [20]. All methods created for HTTP requests were divided into two groups: methods sending a GET request and methods sending POST, PUT and DELETE requests. The methods of the first group have been moved to a separate apiMethods.js file, while the methods of the second group have been added to store.js. They were added to the repository to have access to the data to be sent to the server, such as entity IDs and data from fillable entity registration forms. All methods are called from components directly. Below is an example code that sends a GET request.

```
export async function getSchools() {
  var data;
  var token = JSON.parse(localStorage.getItem("user")).token;
  await axios.get(process.env.VUE_APP_ROOT_API + "schools", {
     headers: {
       Authorization: `Bearer ${token}`,
     },
   }).then((res) => {
     data = res.data;
   });
  return data;}
```

Fig. 12. Method sending GET request

There are three main areas on the page SPA web page: a header with a logout button, a vertical navigation menu on the left, and a main window on the right that will display all data entry forms and tables. There are three main navigation tabs in the navigation menu, containing two buttons for registering and managing tournaments, participants and schools data respectively. The navigation menu with opened navigation tabs is shown in Figure 13.

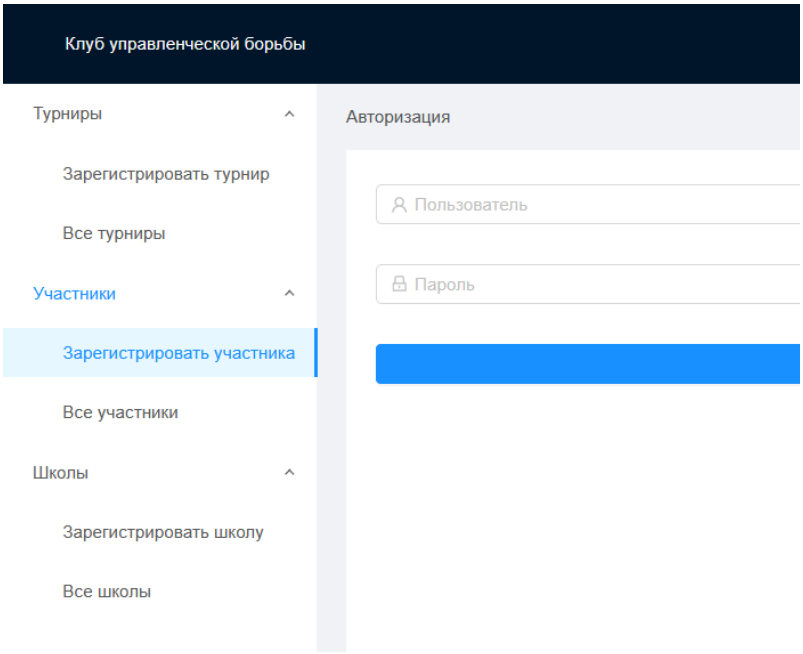

Fig. 13. Navigation menu

When a user clicks on one of the buttons in the navigation tab, the corresponding window opens on the right if the user is logged in. Otherwise, the page that requires authorization will always be displayed.

To register a tournament, the user must click on the appropriate button in the navigation menu. The called window is shown in the figure below. In this window, you must fill out the form by entering the tournament data.

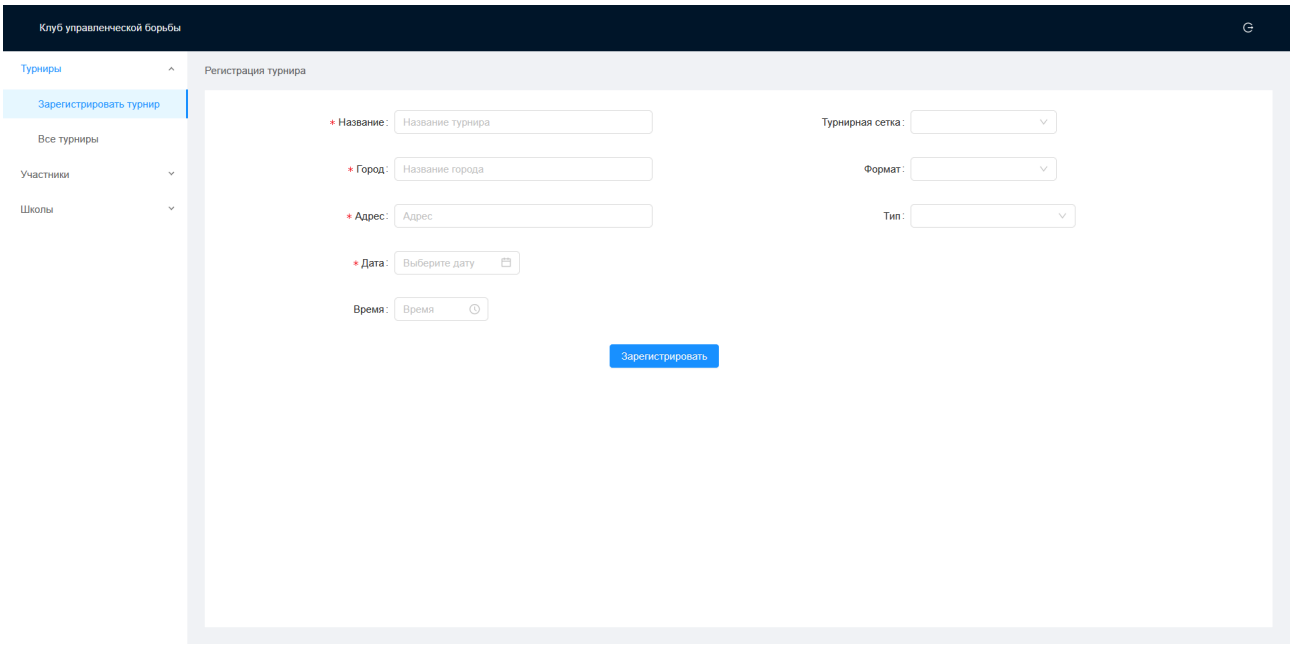

Fig. 14. Tournament registration window

In order to see the entire list of tournaments, the users need to click on the "All tournaments" button. The list will be displayed in form of a table containing registered tournaments. In the "Action" column there are buttons that allow to change the tournament data or delete it, as well as a button that allows to see the list of participants in the tournament for subsequent removal of participants and adding new ones. A screenshot of this page is shown in the figure below.

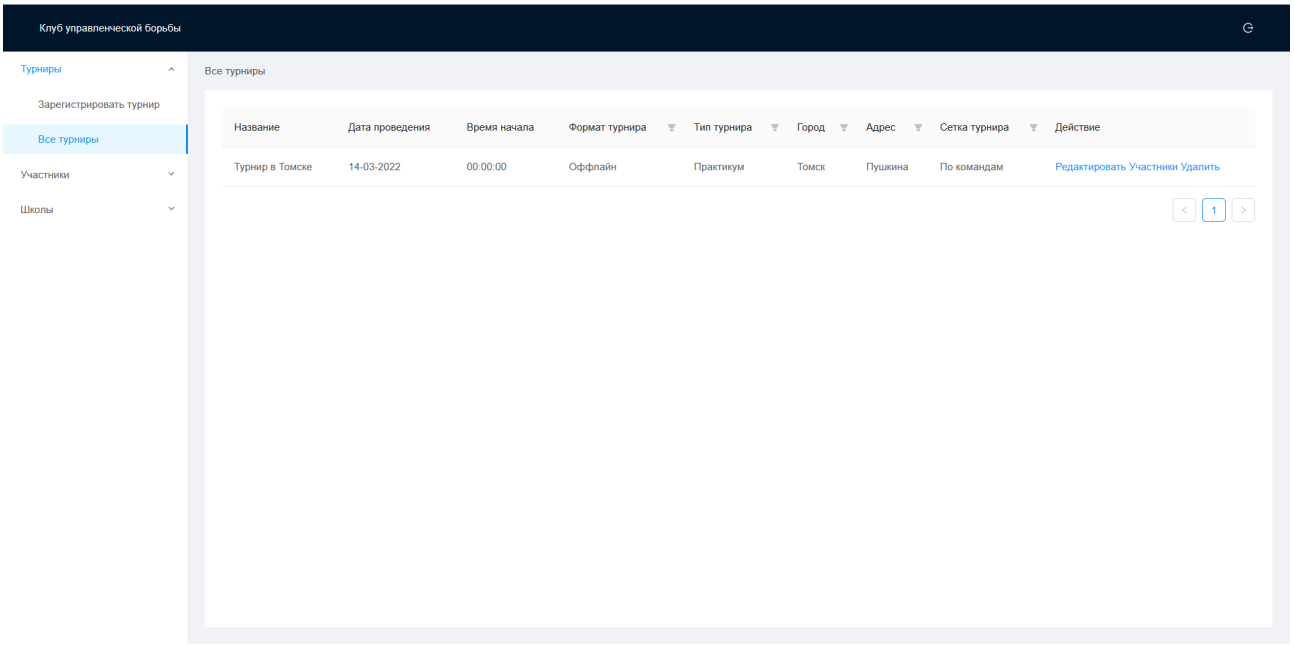

Fig. 15. Tournament list

Below is the page area containing the tournament editing form.

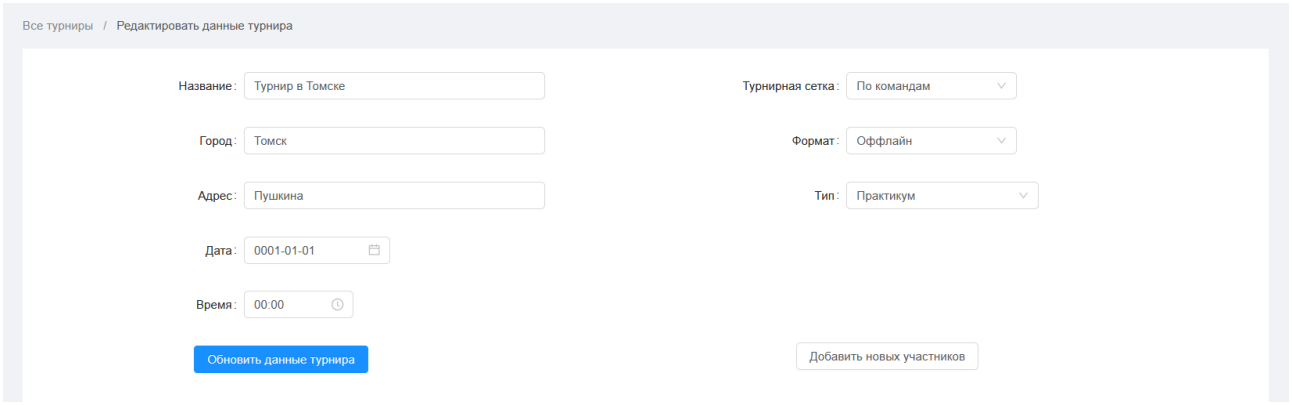

Fig.16. Tournament editing form

Upon clicking on the "Add New Members" button, a page containing the table shown in the figure below will be displayed. The users can select the necessary participants and set their roles in the tournament. After adding a participant, the entry will disappear from the table, and a corresponding notification will be displayed below the table.

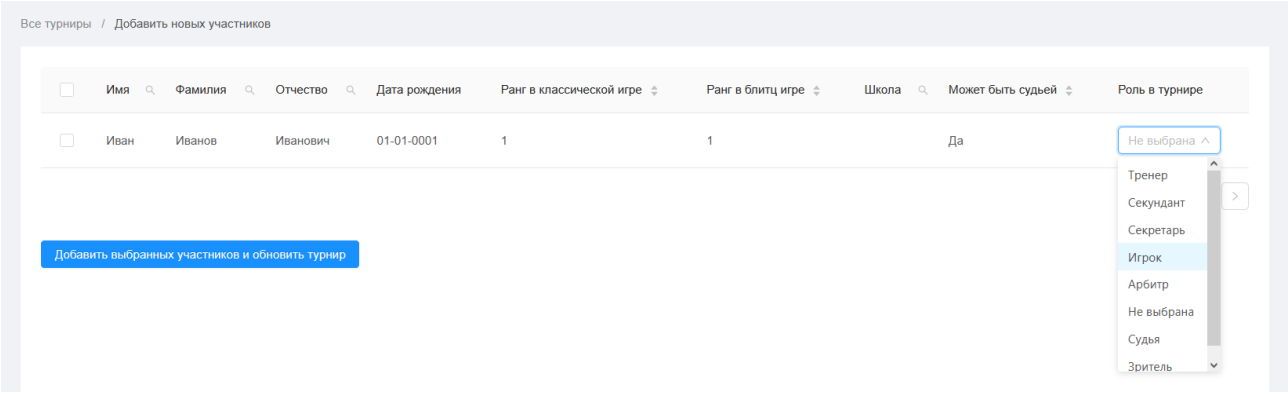

Fig. 17. Adding new participants

To delete added participants, the users needs to return to the page with all tournaments and find the tournament he needs. In it, he needs to click on the "Participants" button, after which a table with participants in the tournament will be displayed. He can remove members from it. An example is shown below.

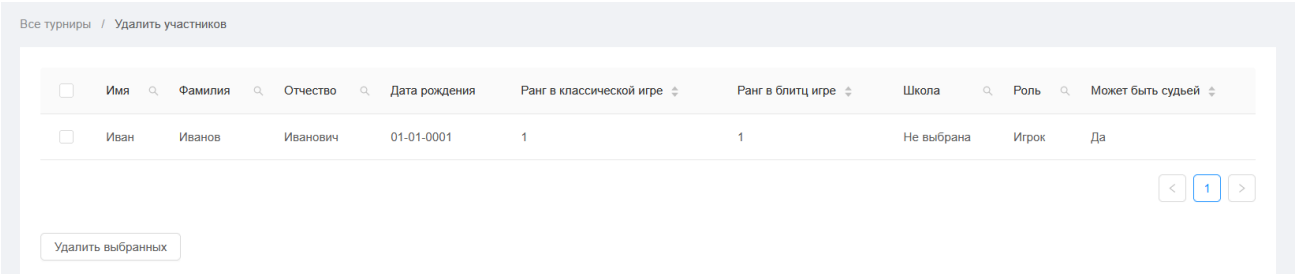

Fig. 18. Deleting participants.

To register and manage the data of participants and schools, similar tables and forms were created in terms of the structure page design and button functions. However, there is no button to remove a participant from the tournament in the "All participants" table. This feature is provided only for the table with tournaments. Screenshots of the pages for registering, editing and deleting members are shown below.

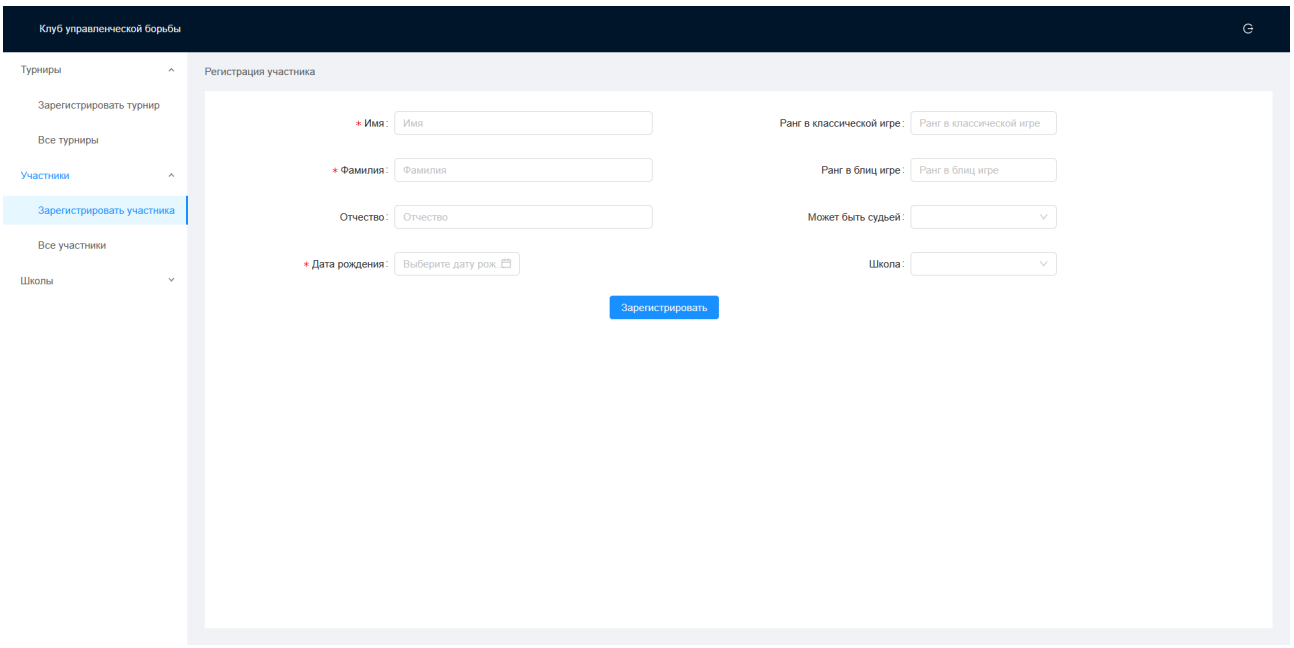

Fig. 19. Registration of participants

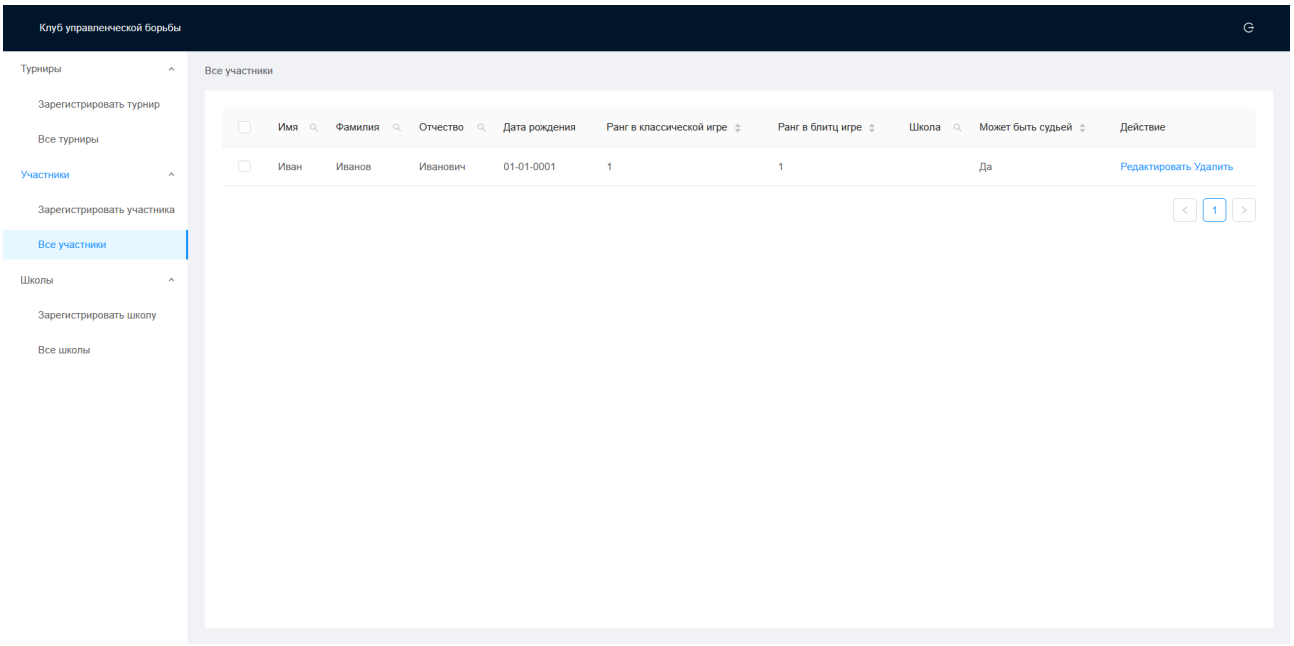

Fig. 20. Participant list

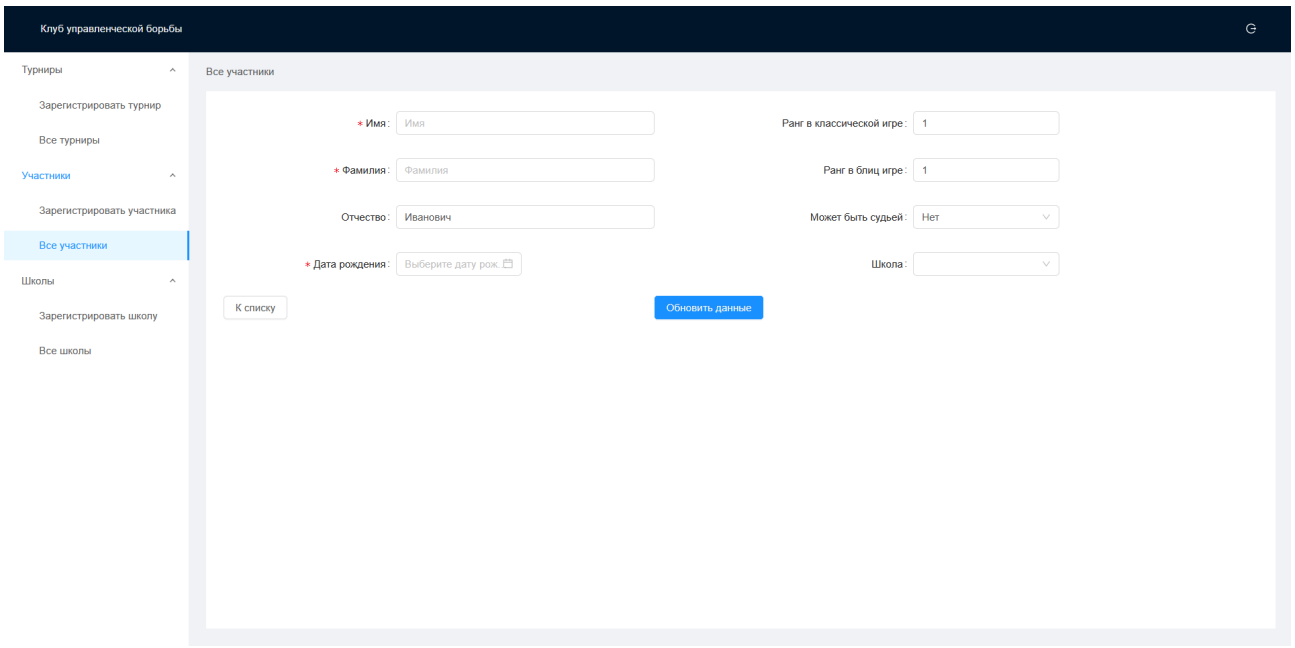

Fig. 21. Participant editing form

To register a school its enough to enter its name. In the future, if necessary, other fields will be added. Below are screenshots of pages related to registration, editing and deleting schools.

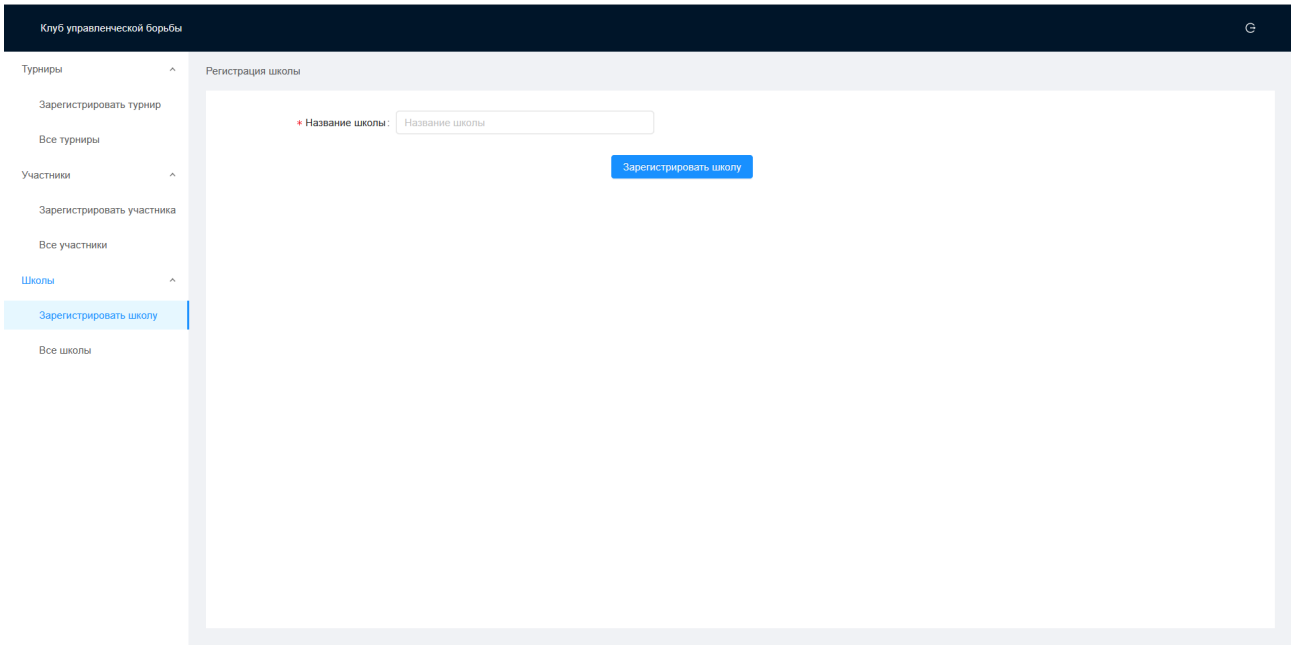

Fig. 22. School registration

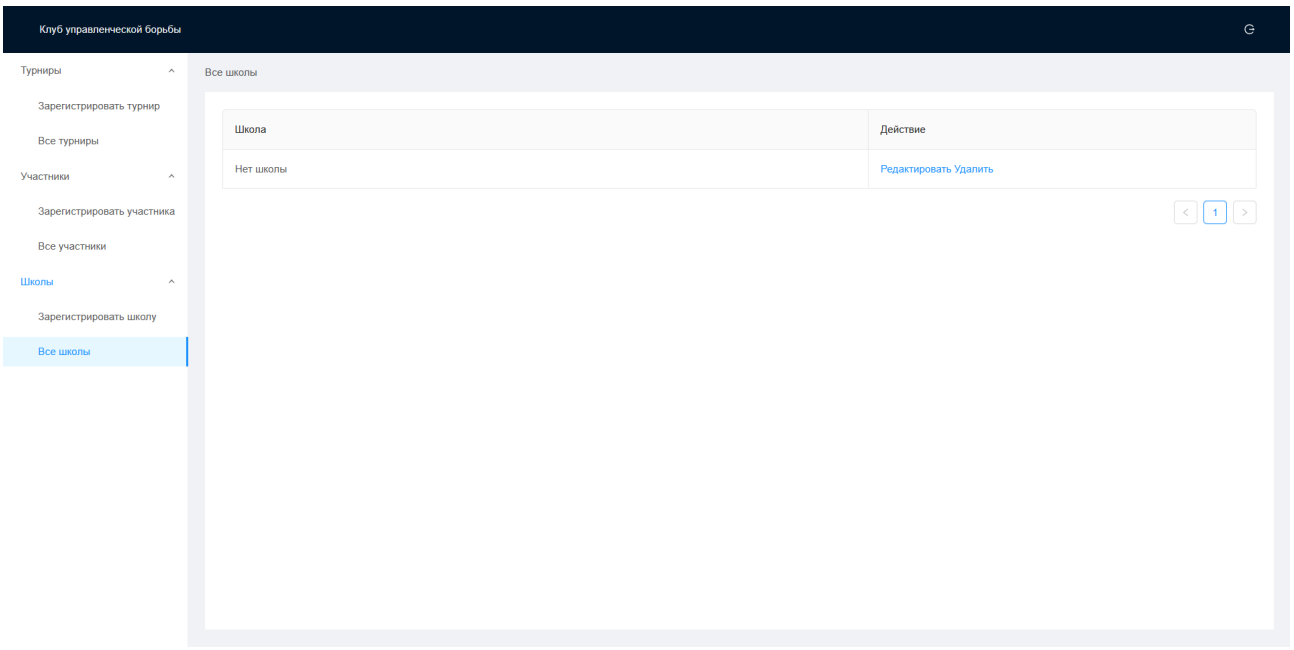

Fig. 23. School list

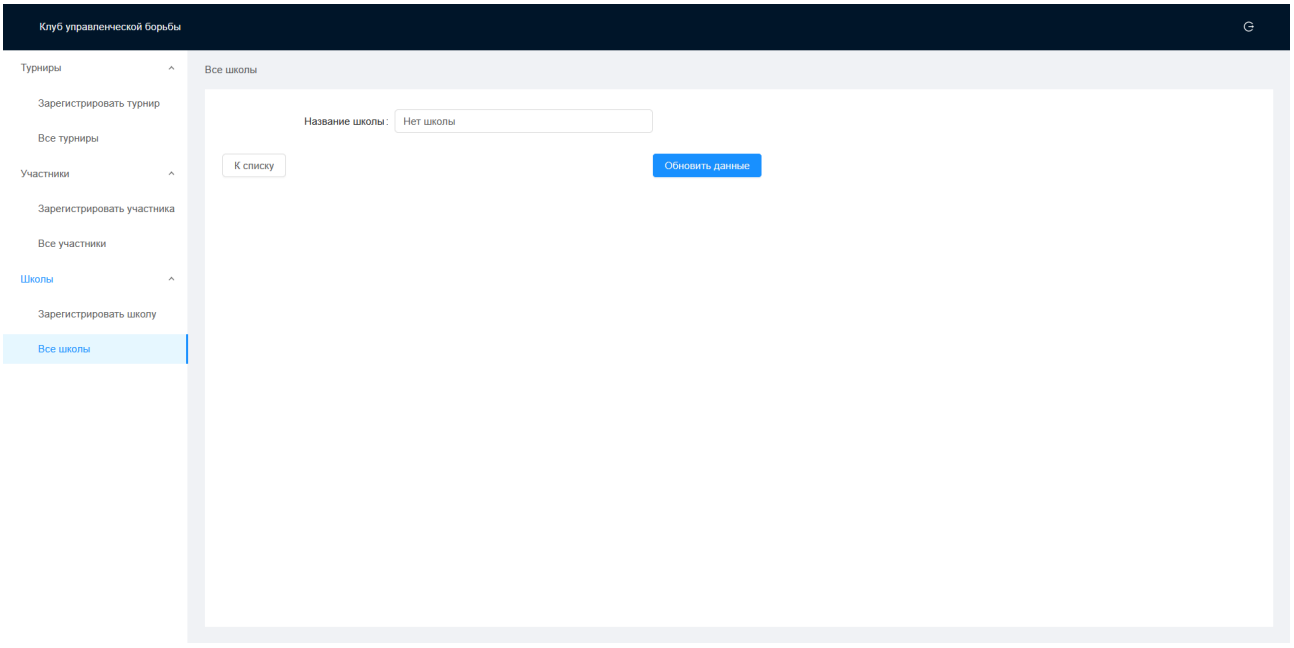

Fig. 24. School edititng form

ПРИЛОЖЕНИЕ Б

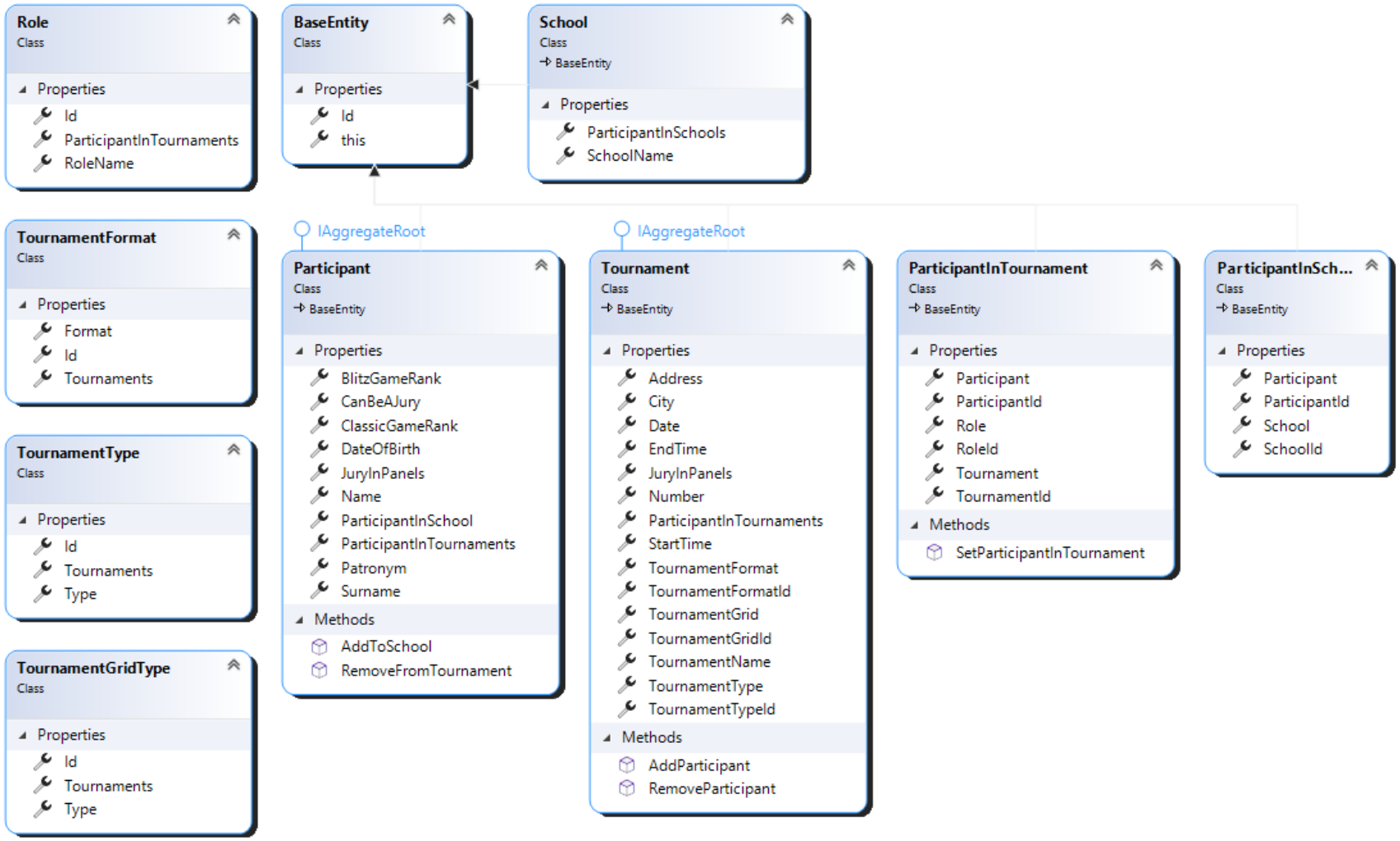

Рисунок 31. Диаграмма классов