Министерство науки и высшего образования Российской Федерации

Федеральное государственное автономное образовательное учреждение
высшего образования
«Уральский федеральный университет
имени первого Президента России Б.Н. Ельцина»

Институт радиоэлектроники и информационных технологий – РТФ

Школа бакалавриата

ОТЧЕТ

По проекту
«Разработка Web-сервиса для автоматизации проведения хоккейных турниров»

по дисциплине «Проектный практикум»

Заказчик: Полозов А.А.

Куратор: Шестеров М.А.

Студенты команды WebChads

Козлов С.Е.

Маликов И.И.

Мамонтов Н.В.

Ягунов А.А.

Екатеринбург, 2025

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

1 Цели и задачи проекта

1.1 Цели

Целью данного проекта является создание веб-сервиса для автоматизации проведения хоккейных турниров, включая:

а) регистрацию участников, формирования расписания и микроматчей;

б) расчёт динамического рейтинга игроков на основе результатов;

в) интеграция с SMS-оповещениями.

1.2 Задачи

Список задач включает в себя:

а) разработку интерфейсов для игроков, судей, администраторов;

б) реализацию алгоритма расчёта рейтинга;

в) обеспечение поддержки большого количества одновременно пользующихся приложением пользователей.

2 Актуальность проекта

В текущей системе существуют некоторые проблемы, а именно:

а) ручной учёт результатов рейтинга;

б) отсутствие централизированной платформы для анализа игроков;

в) сложность организации турниров из-за отсутствия автоматизации.

Решение данного проекта должно устранить эти проблемы, повысив эффективность отбора талантов.

3 Область применения

Изначально, приложение планировалось только для одного спортивного клуба. Но ничто не мешает в будущем расширить проект и для других клубов, или даже для других спортивных дисциплин, например киберспорт. Но основная область применения продукта – это спорт. А если ещё конкретнее – спортивные состязания.

4 Ожидаемые результаты

От продукта ожидаются следующие результаты:

а) сокращение времени организации турнира;

б) точный расчёт рейтинга для выявления сильнейших игроков.

ОСНОВНАЯ ЧАСТЬ

1 Информация о работе каждого участника в отдельности

1.1 Frontend-разработчик

1.1.1 Разработка пользовательских интерфейсов

От frontend-разработчика ожидалась реализация следующих интерфейсов:

а) вход и регистрация;

б) личный кабинет для разных типов пользователей;

в) просмотр списка турниров и подробной информации по каждому турниру;

г) административная панель для создания турнира,

д) панель судейства турнира для судьи,

е) панель «моё расписание» для игрока.

Лучше понять, какие интерфейсы нужны приложению, нужно составить диаграмму «User Path» (приложение А). Из списка выше реализованы только первые два пункта

1.1.2 Интеграция с API

Интеграция с API состоит из двух частей:

– запросы к backend-приложению (серверу), чтобы обеспечить данными интерфейсы из пункта выше;

– обработка серверных ошибок, суть которой заключается в выводе информации пользователю без «падения» приложения.

Из списка выше частично реализован первый пункт.

1.1.3 Адаптивный дизайн

Так как приложение преимущественно будет использоваться на телефоне, то необходимо внедрить адаптивный под это дизайн. Было принято решение использовать подход Mobile First, который заключается в том, чтобы изначально писать CSS-стили для телефонов, а потом адаптировать для компьютеров.

1.2 Backend-разработчик

1.2.1 Разработка RestfulApi и WebSocket-эндпоинтов

Разработка RestfulApi является основой для backend-разработки, так как является, по сути, «мостиком» между базой данных и frontend-приложением. Ниже приведён список эндпоинтов, которые обслуживают базовые бизнес-сущности:

– сущности User и Accounts для регистрации и авторизации;

– сущность Tournament;

– сущность Match.

Также, существуют промежуточные, но не менее важные эндпоинты. Эти эндпоинты обеспечивают взаимодействие между базовыми бизнес-сущностями. Функционал, который реализуют такие эндпоинты:

– расчёт рейтинга,

– сепарация рейтинг-листов,

– судейство турнира.

1.2.2 Интеграции

Чтобы реализовать одну из важнейших фич приложения: уведомление игроков, backend-приложение должно интегрироваться со сторонними сервисами, то есть интеграция с SMS-сервисом

1.2.3 Работа с данными

Работоспособность эндпоинтов строится на данных из БД. Поэтому работа с данным – одна из основных задач backend-разработчика. Для этого backend-разработчик строил ER-диаграммы, которые потом внедрить в БД. (приложение А)

1.2.4 Безопасность

В реальном процессе проведения турнира участвуют три основных вида участников: судья, игрок, администратор. Каждый из них имеет свой определённый набор функций, который может выполнять только данный тип. Из этого вытекают требование к безопасности, которое реализуется разграничением прав. Список бизнес-правил:

– судьи не могут создавать турниры;

– игроки видят только свои данные;

– только администраторы могут создавать турниры;

– только судьи могут вносить результаты в ходе турнира

1.3 UX/UI Дизайнер

Дизайнер отвечал за создание интуитивного, функционального и визуального приятного интерфейса, соответствующего требованиям ТЗ. ТЗ перерабатывалось в User path, который является «источником истины» для дизайнера (приложение А). Макеты страниц, которые реализовал дизайнер:

– вход/регистрация,

– личный кабинет пользователя,

– просмотр турниров,

– создание турниров,

– управление турнирами,

– просмотр расписания турнира,

– панель судейства турнира,

– загрузка личных данных и документов.

1.4 Тимлидер

1.4.1 Взаимодействие с заказчиком

Взаимодействие с заказчиком является основной обязанностью тимлида. Требования к продукту могут меняться, их нужно уточнять. В рамках текущего проекта тимлидер провёл 5 встреч, которые позволили составить расширенное техническое задание, а также документы, позволяющие членам команды понимать свои будущие задачи.

1.4.2 Составление документов

К этим документам относятся:

– PRD (Product Requirements Document),

– FCF (Flow Chart Functionality),

– User Path

Документы находятся в приложении (приложение А)

1.4.3 Организация работы

Суть данного пункта заключается в том, чтобы предоставить результат двух предыдущих участникам команды в понятной форме. Для этого:

– проведение регулярных созвзонов с командой для прояснения тонких моментов,

– декомпозиция и дробление содержимого документов, в результате чего формируется задачи для Kanban-доски;

– распределение этих задач на весь семестр, что приводит к формированию беклога.

2 Разбор требований заказчика и пользователей к программному продукту и составление плана действий

2.1 Анализ и систематизация требований

На основании проведенных интервью с заказчиком (ДЮСШ «Спартаковец») были выявлены следующие ключевые требования

2.1.1 Требования заказчика

Автоматизация процесса проведения турнира:

– поддержка формата микроматчей (1минута игры + 30 секунд паузы);

– автоматическое формирование пар с учетом рейтинга (разница меньше, чем 140 пунктов);

– реализация алгоритма расчета рейтинга по формуле

Интеграционные требования:

– SMS-оповещения участников.

Производительность:

– поддержка 600+ одновременных пользователей;

– Обновление рейтинга в реальном времени.

2.1.2 Требования пользователей

Для игроков:

– простая регистрация и подача заявок;

– доступ к просмотру персонального рейтинга;

– мобильный интерфейс для просмотра расписания.

Для судей/тренеров:

– удобный ввод результатов с мобильных устройств;

– возможность быстрой корректировки расписания;

Для администраторов:

– централизированное управление турнирами;

– модерация участников и документов;

– управление стоп-листом.

2.2 Приоритезация требований

Требования были классифицированы по методу MoSCoW.

2.2.1 Must have (обязательные)

К обязательным требованиям относятся:

– механизм регистрации и аутентификации;

– система расчета и отображения рейтинга;

– формирование расписания микроматчей;

– базовые интерфейсы для всех ролей;

– SMS-уведомления;

– мобильный интерфейс.

2.2.2 Should have (желательные)

К желательным требованиям относятся:

– расширенная аналитика для тренеров;

– детализированная статистика для матчей.

2.2.3 Could have (возможные) и Won’t have

Было принято решения не составлять требования для этой градации, так как проект довольно объёмный и первостепенной задачей является реализация обязательных требований.

2.3 Формирование беклога

На основании приоритеированных требований был составлен беклога, разбитый на 4 спринта по 2 недели.

2.3.1 Спринт 1. Базовый функционал

В первый спринт входит:

1) реализация системы регистрации и аутентификации;
2) разработка базовой части личного кабинета;
3) создание API для работы с пользовательскими данным;
4) настройка базовой инфраструктуры проекта.

2.3.2 Спринт 2. Управление турнирами

Во второй спринт входит:

1) механизм создания и настройки турниров;
2) алгоритм формирования микроматчей;
3) базовый интерфейс для панели судьи;
4) интеграция с SMS-сервисом.

2.3.3 Спринт 3. Интеграция и безопасность

В третий спринт входит:

1) нагрузочное тестирование API;

2) реализация системы ролей и прав доступа.

2.3.4 Спринт 4. Оптимизация и доработки

В четвёртый спринт входит:

1) реализация системы стоп-листа;
2) оптимизация алгоритма расчёта рейтинга;
3) финальное тестирование.

3 Анализ и сопоставление аналогов разрабатываемого продукта

Для выявления конкурентных преимуществ и недостатков разрабатываемой системы был проведен сравнительный анализ 3 популярных платформ для организации спортивных турниров. Критерии сравнения включали функционал для хоккея, систему рейтингов, поддержку микроматчей, мобильность и стоимость.

Таблица 3.1 – Анализ конкурентов

Сервис	Преимущества	Недостатки	Цена
ChallengeMode	широкая поддержка различных видов спорта; гибкая система настройки турниров; интеграция с социальными сетями.	нет специализации под хоккей; отсутствие формата микроматчей; сложная система расчета рейтинга.	200$ в месяц
LeagueApps	профессиональные инструменты для спортивных школ; поддержка мобильных устройств; система онлайн-платежей.	нет алгоритмов автоматического формирования пар; ограниченная аналитика; высокий порог входа.	300$ в месяц
SportsEngine	удобное расписание; система коммуникации.	нет поддержки микроматчей; устаревший интерфейс; слабая система рейтингов.	индивидуальный расчет

Таблица показала, что разрабатываемая система не имеет прямых аналогов на рынке, где бы сочетались специализация под хоккей, поддержка инновационного формата микроматчей, уникальная формула расчёта рейтинга. Это создает значительные преимущества для ДЮСШ «Спартаковец».

4 Обзор архитектуры программного продукта, описание компонентов и связей между ними, обоснование выбора архитектурного решения

4.1 Архитектурного программного продукта

Система реализована с использованием микросервисной архитектуры, что обеспечивает:

– высокую масштабируемость отдельных компонентов;

– независимое развёртывание сервисов;

– гибкость в выборе технологи для каждого модуля;

– устойчивость к отказам.

Основные технологические решения (стек):

– frontend: React + TypeScript;

– backend: Golang (микросервисы);

– дизайн: Figma (макеты и прототипы);

– документирование: Draw.io (архитектурные схемы).

4.2 Компонентная диаграмма системы

Ознакомится с диаграммой можно в приложении (приложение Б)

4.3 Описание ключевых компонентов

4.3.1 Клиентское приложение (Frontend)

Для реализации клиентского приложения был выбран React, TypeScript, MobX. Основные модули:

– авторизация (JWT),

– личный кабинет для всех ролей,

– панель администратора,

– интерфейс судьи.

4.3.2 API Gateway

Для реализации API в backend-приложении был выбран язык программирования Golang. Основной функционал:

– маршрутизация запросов,

– балансировка нагрузки,

– кеширование ответов.

4.3.3 Микросервисы (backend)

AuthService обеспечивает аутентификацию и авторизацию, управление ролями. TournamentService реализует создание турниров, формирование микроматчей управление расписанием. NotificationService обеспечивает отправку SMS-уведомлений.

4.4 Обоснование выбора архитектуры

4.4.1 Микросервисный подход

Данный подход был выбран по ряду причин:

– разные компоненты системы имеют различные требования к нагрузке;

– необходима независимая масштабируемость критичных сервисов;

– упрощает внедрение изменений без остановки всей системы.

4.4.2 Выбор стека разработки

Причины выбора React + TypeScript:

– система типов для снижения ошибок;

– богатая экосистема компонентов;

– поддержка сообщества.

Причины выбора Golang для backend:

– высокая производительность;

– эффективное управление памятью;

– поддержка конкурентости.

5 Описание методологии разработки, информация о процессе разработки, отчёт о результатах тестирования

5.1 Методология разработки

Для организации работы была выбрана гибридная модель Scrum + Kanban. Основные правила:

– итеративная разработка (4 спринта по 2 недели);

– ежедневные стендапы (15 минут);

– гибкое управление задачами через Yandex Tracker;

– регулярные ретроспективы (раз в 2 недели).

5.2 Процесс разработки

Реализация каждой фичи происходила по следующей схеме:

1) разбивка User Stories на технические задания;
2) ветвление по Git Flow;
3) документирование API через Swagger;
4) тестирование API;
5) разработка интерфейсов по макетам из figma;
6) тестирование интерфейсов;
7) связывание API и интерфейсов;
8) деплой.

5.3 Отчёт о результатах тестирования

5.3.1 Backend

Было принято решение не использовать Unit-тесты, а использовать ручной метод. На backend-стороне успешно были протестированы следующие сервисы:

– «AuthService»,

– «SmsService»,

– «AccountService»,

– «TournamentService».

Не был реализован и протестирован только сервис для судейства турнира. Ознакомится со всеми сервисами можно по ссылке: https://github.com/orgs/WebChads/repositories

5.3.2 Frontend

Для этой части также было принято решение использовать ручной метод. На frontend-стороне успешно были протестированы следующие модули:

– «Авторизация пользователей»,

– «Базовый личный кабинет».

Не было реализовано много чего. И эта проблема, которую планируется закрыть в оставшееся время до защиты. Ознакомится с репозиторием frontend-приложения можно в приложении (приложение А).

6 Информация о планировании деятельности в ходе разработки и распределении задач между участниками команды разработчиков

6.1 Организация рабочего процесса

Была использован гибридный формат Scrum + Kanban со следующими составляющими:

– итерации: 4 спринта по недели;

– артефакты, включающие backlog и sprint board.

Цикл планирования включает в себя:

1) планирование спринта с разбивкой на подзадачи;

2) ежедневные стендапы;

3) ретроспектива после спринта.

6.2 Распределение ролей и зон ответственности

Распределение обязанностей по каждой роли приведено на таблице 6.1.

Таблица 6.1 – Распределение ролей и зон ответственности

Роль	Технологии	Основные задачи
Frontend-разработчик	React, TypeScript	Реализация интерфейсов, интеграция с API, адаптивная верстка
Backend-разработчик	Golang, PostgreSQL	Разработка микросервисов, оптимизация запросов, интеграция с SMS-сервисов
Дизайнер	Figma	Прототипирование интерфейсов, адаптация под mobile,

Продолжение таблицы 6.1

Роль	Технологии	Основные задачи
Дизайнер	Figma	Прототипирование интерфейсов, адаптация под mobile, подготовка UI-кита
Тимлидер	Draw.io, Yandex Tracker	Координация команды, декомпозиция задач, взаимодействие с заказчиком

ЗАКЛЮЧЕНИЕ

7 Оценка соответствия продукту требованиям

По ключевым функциональным параметрам проект выполнен на 65-70%:

– созданы все макеты в figma;

– реализован базовый цикл турнира (регистрация, формирование пар, расчет рейтинга);

– достигнута требуемая производительность.

Со стороны backend-разработки проект практически готов. Осталось доработать сервис судейства и расписания для игроков. Проблемы возникли по части frontend. Тут реализовано всего 2 требования мало требований:

– регистрация и авторизация пользователя;

– базовый личный кабинет;

Требования со стороны frontend практически не выполнены. Нужны доработки пока ещё есть время до итоговой защиты

8 Оценка качества продукта на основе результатов тестирования

Как говорилось ранее, команда не использовала инструменты автоматического тестирования, поэтому сложно что-либо про это написать. Тестирование проводилось в ручном режиме, «чтобы работало».

9 Предложения по улучшению продукта

9.1 Объем разработки

Проект полностью реализован со стороны дизайнера. Почти полностью со стороны backend-разработки. Но со стороны frontend-разработки практически ничего не реализовано. Это связано с нагрузкой frontend-разработчика по другим предметам в вузе и специфики frontend-разработки в целом: она довольно нудная и трудоёмкая. Вывод тут только один: качественно вести разработку можно только в найме, когда ты полностью отдан одному делу и тебе платят за него деньги. Ожидается, что за время, которое есть у команды до итоговой защиты, fronted-приложение будет полностью реализовано на уровне MVP.

9.2 Автоматическое тестирование

Это важный пункт, который команде не удалось внедрить. Это требует значительных временных затрат, но позволяет избежать большого количества ошибок. В будущем, на более крупных проектах, использование инструментов автоматического тестирования будет является обязательной задачей

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. ГОСТ Р 56939-2022. Программное обеспечение. Требования к качеству и тестированию. – Москва : Стандартинформ, 2022. – 45 с.

2. Мартин Р. Чистая архитектура. Искусство разработки программного обеспечения. – Санкт-Петербург : Питер, 2021. – 352 с.

3. Фаулер М. Рефакторинг. Улучшение существующего кода. – Москва : Диалектика, 2019. – 448 с.

4. Таненбаум Э., Вудхалл А. Операционные системы. Разработка и реализация. – Санкт-Петербург : Питер, 2020. – 1120 с.

5. Документация React [Электронный ресурс]. – Режим доступа: https://react.dev.

6. Официальная документация Golang [Электронный ресурс]. – Режим доступа: https://go.dev/doc.

7. Twilio API Reference [Электронный ресурс]. – Режим доступа: https://www.twilio.com/docs.

8. PostgreSQL 15 Documentation [Электронный ресурс]. – Режим доступа: https://www.postgresql.org/docs/15.

9. Docker Documentation [Электронный ресурс]. – Режим доступа: https://docs.docker.com.

10. RabbitMQ Official Documentation [Электронный ресурс]. – Режим доступа: https://www.rabbitmq.com/documentation.html.

Приложение А

(обязательное)

Результаты каждого участника

1 Тимлидер

Ресурсы:

а) Расширенное техническое задание - https://docs.google.com/document/d/1w1UxyF9sRsgkmobSyBrA0uelQ3e4gQcS/edit?usp=sharing&ouid=114153666815274472365&rtpof=true&sd=true

б) FCF (Flow Chart Functionality) - https://drive.google.com/file/d/1x8wcWnKrwh4PsPio22H410aIIHb72IW4/view?usp=sharing

в) User Path - https://drive.google.com/file/d/14zGGoJRluLKe3SOXyG7vQKWHWVZfbilF/view?usp=sharing

г) Доска задач с беклогом - https://tracker.yandex.ru/agile/board/1

2 Дизайнер

Ресурсы:

а) Макеты в figma - https://www.figma.com/design/iH1v9iSh8bGdeErRMLGR2e/Hockey-App?node-id=43-15&p=f&t=C5uYbnBQLg2sKgZ8-0

3 Frontend-разработчик

Ресурсы:

а) Репозиторий на GitHub - https://github.com/WebChads/Frontend

4 Backend-разработчик

Ресурсы:

а) Репозитории для сервисов (все кроме «Frontend») - https://github.com/orgs/WebChads/repositories.

Приложение Б

(необязательное)

Компонентная диаграмма системы

Диаграмма представлена на Рисунке – 1

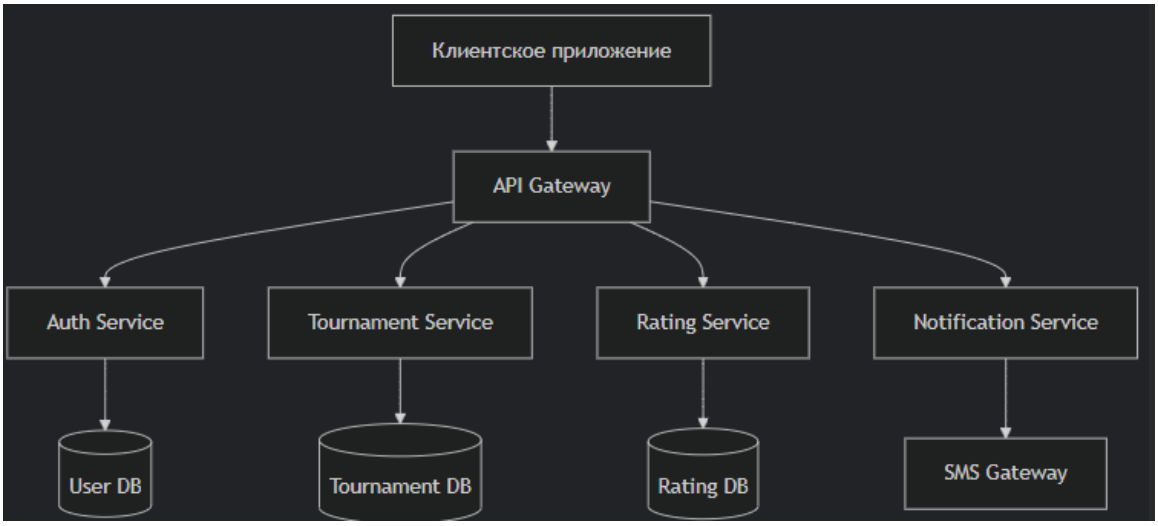

Рисунок 1 – Компонентная диаграмма системы