Национальный исследовательский университет
«Высшая школа экономики»

Факультет
компьютерных наук

Индивидуальный программный курсовой проект

Клиент-серверное приложение для системы управления проведением соревнований

Client-Server Application for the Competition Management System

Исполнитель:
Дробот Алексей Андреевич
студент группы БПИ228

Научный руководитель:
Сосновский Григорий Михайлович
внештатный преподаватель ДПИ ФКН

май 2025

Клиент-серверное приложение для системы управления проведением соревнований

Описание предметной области

>10 000
участников соревнований

Участники из
5 регионов

Необходимо
Повысить вовлеченность аудитории

Мероприятия
не реже 5 раз
в месяц

Общие ожидания от решения:

- Возможность быстрого и удобного доступа к результатам турниров для зрителей
- Возможность мониторинга событий в реальном времени
- Автоматизация работы с судейским корпусом по назначениям на матчи
- Повышение общей удовлетворенности участников, путем предоставления расширенной статистики

iOS

Мобильное приложение «ФХМ»

Архитектура
клиент-серверное приложение

Основная функция
Обзор результатов соревнований

+ назначения судей

5 функциональных блоков
результаты турниров и судейский раздел

Поддержка работы
С двумя базами данных

Актуальность работы

>80%

участников обращаются к ресурсам с мобильных устройств

~10 000 визитов

от уникальных посетителей приходит ежедневно

Отсутствует унифицированное решение от вышестоящих Федераций

Потенциал масштабирования на всех участников соревнований по виду спорта

?

Специфические требования к моделям данных из-за специфики вида спорта

Постоянное обращение к веб-платформе усложняет пользовательский путь

Текущее решение этой задачи федерацией является **не комплексной** и **не закрывают полный список потребностей пользователей**

Цели и задачи работы

Цель:

Разработать клиент-серверное приложение для системы управления проведением соревнований

Задачи:

- Выявить функциональные и нефункциональные требования
- Провести сравнительный анализ аналогов
- Спроектировать архитектуру и взаимодействия приложения
- Выбрать и обосновать средства разработки
  - Улучшить знания JavaScript и фреймворка ExpressJS
  - освоить работу с СУБД PostgreSQL и MariaDB
  - исследовать отличия UIKit и SwiftUI и изучить последний
- Определить структуру двух баз данных и отразить её на ERD диаграмме
- Разработать серверную часть приложения
- Определить принцип работы текущей статистической базы и способы взаимодействия с ней
- Разработать клиентскую часть приложения
- Сформировать сопроводительную документацию по ГОСТ в соответствии с ЕСПД

Клиент-серверное приложение для системы управления проведением соревнований

Сравнительный анализ аналогов

Название приложения	Существующие mobile app	Наличие API для работы	Возможность интеграции	Работа с назначениями судей	Возможность процессных доработок	Работа на российском рынке
Registry.fhr.ru	⛔	✅	⚠️	⚠️	⚠️	✅
cib.khl.ru	✅	✅	⛔	⛔	✅	⚠️
TeamSnap	⛔	✅	✅	⛔	✅	⛔
SportsEngine	✅	✅	⛔	✅	⚠️	⛔
GameSheet	✅	✅	⛔	⚠️	✅	⛔
MtGame	⛔	✅	⛔	⛔	✅	✅

Необходимо решение, отвечающее требованиям и существующим процессам внутри Федерации хоккея Москвы

Архитектура приложения

Клиентская (front-end) часть

Мобильное приложение

- Целевая ОС: iOS 18.4.1
- API взаимодействие с сервером
- MVVM – целевая архитектура

WEB

Объектное хранилище

Хранение статических данных

- Масштабируемость
- Возможность хранения различных типов данных
- Высокая доступность данных
- Простота интеграции
- Целостность информации

API

Серверная (back-end) часть

Набор API-endpoints

- Язык программирования, удобный для web-приложений
- Доступ по API
- Models + controller архитектура

SQL

База данных

Хранение статистических данных

SQL

База данных

Хранение данных приложения

Клиент-серверное приложение для системы управления проведением соревнований

Средства разработки и их обоснование

Версионный контроль

Объектное хранилище

Amazon S3

- Гибкая модель доступа и управления
- Низкие затраты на содержание
- Тройная репликация
- Доступ по API

timeweb

Клиентская (front-end) часть

Swift 5.7

- Простой в освоении
- Производительный
- Качественная документация
- Полная поддержка платформ Apple

Apple Xcode + Simulator

Серверная (back-end) часть

express

- Мощный и гибкий механизм обработки запросов
- Асинхронная модель программирования
- Широкий выбор пакетов
- Активное сообщество

JB IntellIJ IDEA + Postman + SwaggerUI

База данных

MariaDB

Имеющееся решение в инфраструктуре ФХМ

База данных

PostgreSQL

- Высокая надежность и целостность данных
- Расширяемость и производительность
- Поддержка JSON
- Реляционная структура

JB DataGrip

Клиент-серверное приложение для системы управления проведением соревнований

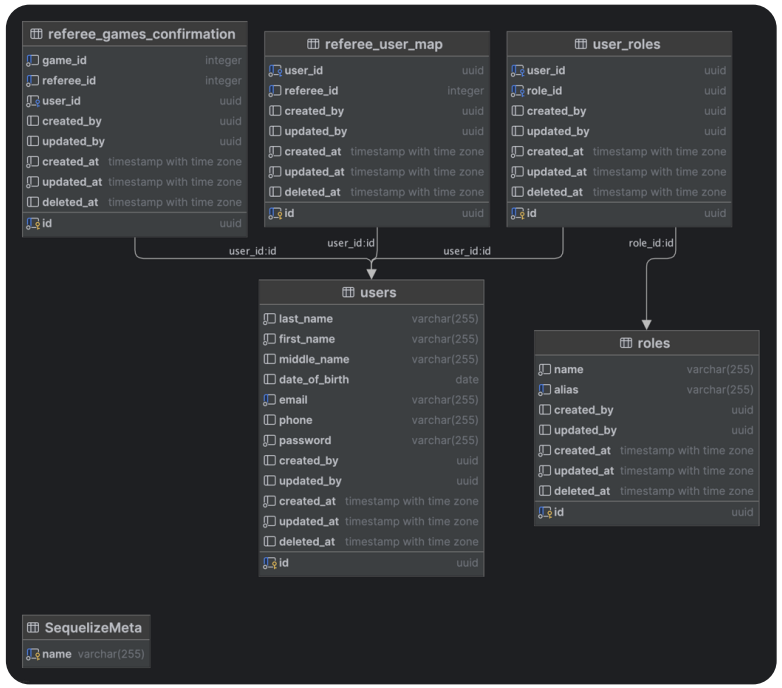
Структура базы данных и ERD-диаграмма

Статистическая БД

Содержит в себе >30 таблиц с информацией о статистике игроков и тренеров, принадлежности к клубам и с назначениями судей

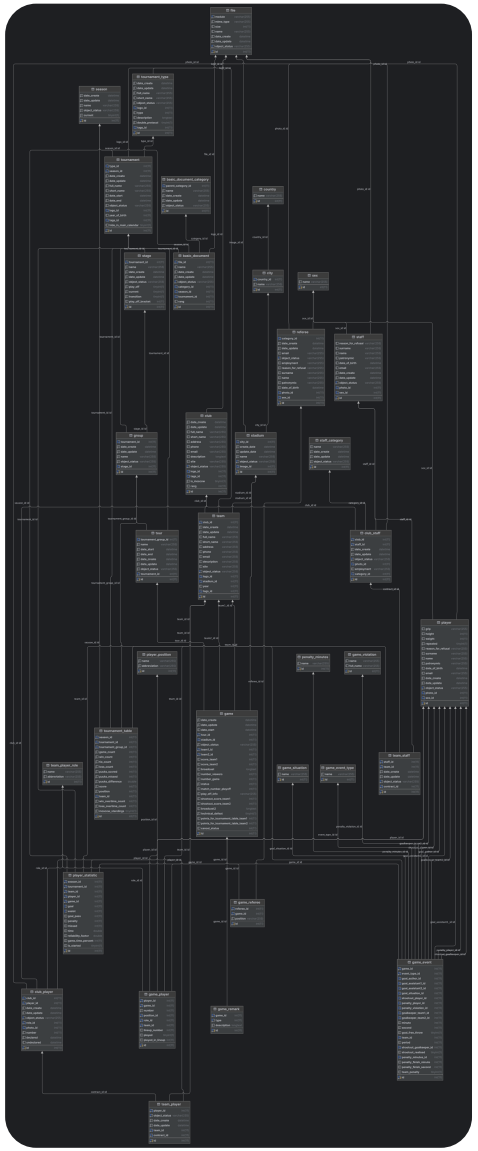
Особенности реализации

Так как используемая статистическая БД подключена к другим системам требуется особенно аккуратная поддержка и внесение изменений в модели и взаимодействия синхронно с релизом.
При реализации БД использовались лучшие практики, такие как «мягкое удаление», история внесения изменений и индексация (где это применимо)



БД приложения

Содержит в себе сведения о пользователях и ролях, а также об отношениях пользователь - судья



Настройка взаимосвязей через **SQL запросы**

Развёртывание СУБД **на выделенном сервере**

Использование идентификаторов **UUIDv4**

Применение **миграций и сидеров** при разработке

Полная схема базы данных с подробными взаимосвязями

Клиент-серверное приложение для системы управления проведением соревнований

Разработка серверной части

Фреймворк

express JS

Docker Compose

Взаимодействие с базой данных

NPM пакет: Sequelize

Преимущества:
- Упрощение синтаксиса кода
- Работа с отношениями между таблицами
- Валидация данных на уровне пакета

- Работа миграций и сидеров в отдельном контейнере после релиза
- Валидация данных перед каждым запросом
- Асинхронная работа
- Обработка ошибок

Применение отдельного контейнера миграций для базы данных приложения

Использование токена авторизации

NPM пакет: JWT

- Выдается при авторизации и используется при запросах к API
- Подписывается с помощью закрытого ключа

Хранение паролей пользователей

NPM пакет: BCrypt

- Помещение пароля для хэширования в файл .env
- Таблица с ограниченным доступом

Тестирование с помощью Swagger

NPM пакет: SwaggerUI

- Автоматическая документация и упрощение тестирования

models

Файлы моделей данных из БД для корректной работы пакета и валидации

routes

Файлы для аутентификации и навигации запросов к контроллерам данных

controllers

Файлы, обеспечивающие логику обработки запросов к приложению

services

Файлы, обеспечивающие основную бизнес-логику приложения и работу с данными

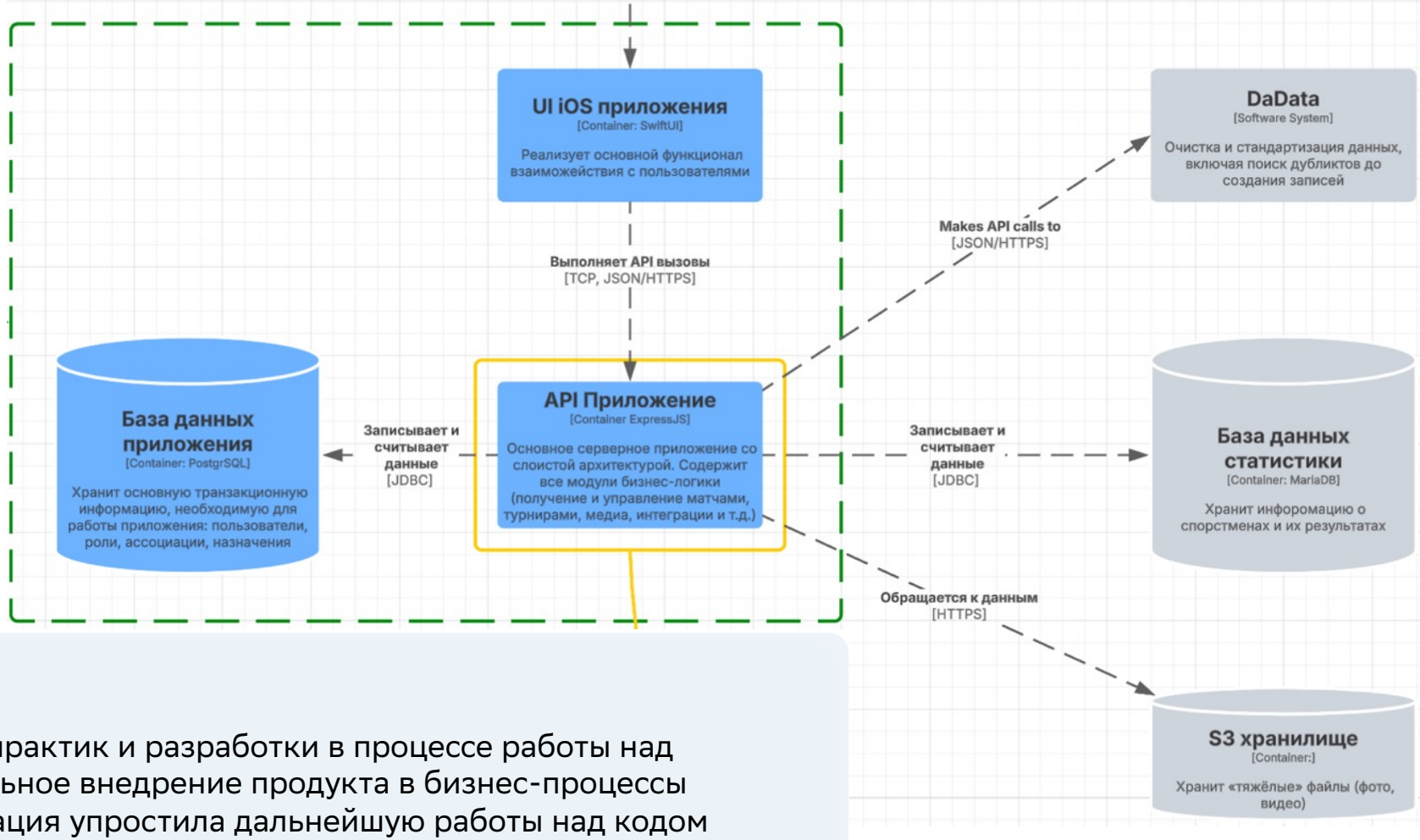
Архитектура серверной части

services

Инкапсулируют бизнес-логику
Вызываются из контроллеров
Используют модели и хелперы

controllers

Обрабатывают HTTP-запрос
Валидируют данные и права
Вызывают сервис / ORM
Формируют DTO-ответ
Логирование, errorHandler



Фокус – на качестве кода

Применение лучших промышленных практик и разработки в процессе работы над проектом позволило приблизить реальное внедрение продукта в бизнес-процессы заказчика. Автоматическая документация упростила дальнейшую работы над кодом

Подробное описание взаимосвязей файлов, конечных точек API и методов обращения к ним можно найти в сопроводительной документации в соответствии с ЕСПД

Клиент-серверное приложение для системы управления проведением соревнований

Разработка клиентской части

Фреймворк

SwiftUI

Целевая сборка: iOS 18.4.1

Без внешних библиотек

Только iPhone

Взаимодействие с сервером

- **Посредством API** запросов к заранее определённым точкам
- **Обработка проблем** с подключением к серверу
- **Обеспечение** краткосрочного **кэширования** данных и информации о пользователе
- **Использование Optional** значений в DTO для автоматического парсинга ответов
- Обработка пагинации и lazy-load запросов

Хранение информации о пользователе

- **Использование UserDefaults** для хранения статических данных о пользователе
- Сохранение **JWT в Apple KeyChain** в зашифрованном виде
- Возможность обновления данных пользователя при загрузке

Обработка исключений

- **Демонстрация предупреждений** пользователю с информацией, достаточной для идентификации проблемы
- UI предусматривает возможность **повторения запросов к серверу**, если предыдущие завершились неуспешно

Работа над интерфейсом велась без предварительного прототипирования и дорабатывалась динамически, исходя из гайдлайнов Apple

Архитектура

MVVM архитектура является целевой для проекта. Код разделен на файлы вида View, ViewModel и Model, каждый из которых отвечает за отображение информации на экране, работу с информацией, в том числе её запрос с сервера, и хранение соответственно.

Интерфейс

Используется динамическое построение интерфейса, которое обеспечивает совместимость с разными размерами экранов

Ключевые этапы пользовательского пути недоступны без подключения к интернету и блокируются при невозможности отправить информацию

Клиент-серверное приложение для системы управления проведением соревнований

Архитектура клиентской части

Services файлы

Представляют собой все необходимое для взаимодействия с API-бэкенда (включая DTO и вызовы API)

ViewModel файлы

Инкапсулируют логику обработки данных после получения для дальнейшего ее визуального представления пользователю

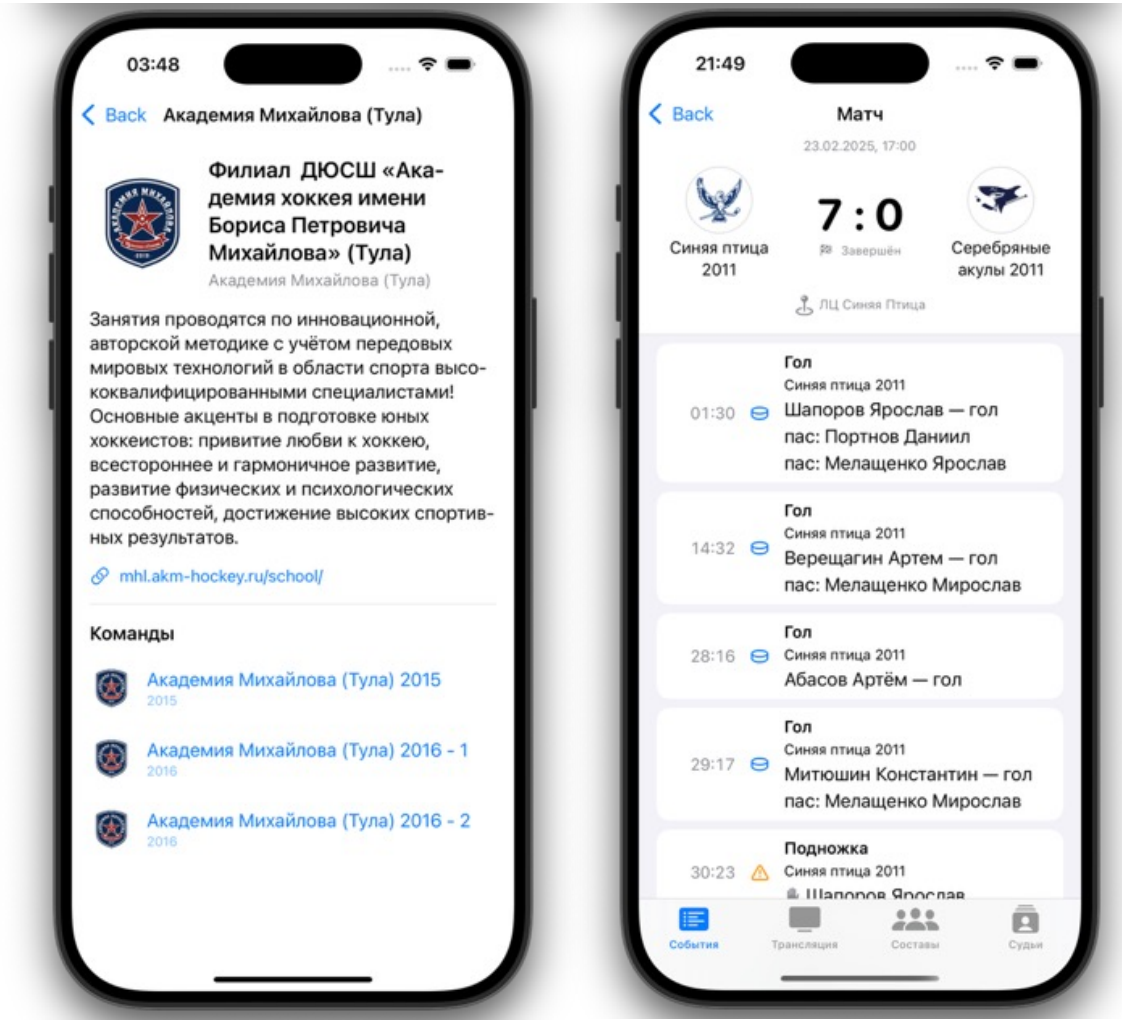
Views файлы

Отвечают за визуальные элементы, их расположение на экране и обработку пользовательских действий. Предполагают адаптивный интерфейс под разные размеры экранов, lazy-load, pull-to-refresh возможности для пользователя. Активно взаимодействует с ViewModel

Все дополнительные обработчики, включая авторизацию и логгеры вынесены в отдельные файлы для упрощения дальнейшего развития проекта в промышленной среде. Код снабжен необходимыми MARK.

Для упрощения дальнейшего поддержания кода элементы пользовательского интерфейса разделяются на отдельные классы и файлы, сгруппированные по смысловому назначению

В перспективе предполагается рефакторинг архитектуры с исправлением ошибок, улучшением читаемости кода



Выполненные цели и задачи

Цель достигнута.

Разработать клиент-серверное приложение для системы управления проведением соревнований

Задачи:

- Выявить функциональные и нефункциональные требования
- Провести сравнительный анализ аналогов
- Спроектировать архитектуру приложения
- Выбрать и обосновать средства разработки
  - Улучшить знания JavaScript и фреймворка ExpressJS
  - освоить работу с СУБД PostgreSQL и MariaDB
  - исследовать отличия UIKit и SwiftUI и изучить последний
- Определить структуру базы данных и отразить её на ERD диаграмме
- Разработать серверную часть приложения
- Определить принцип работы текущей статистической базы и способы взаимодействия с ней
- Разработать клиентскую часть приложения
- Сформировать сопроводительную документацию по ГОСТ в соответствии с ЕСПД

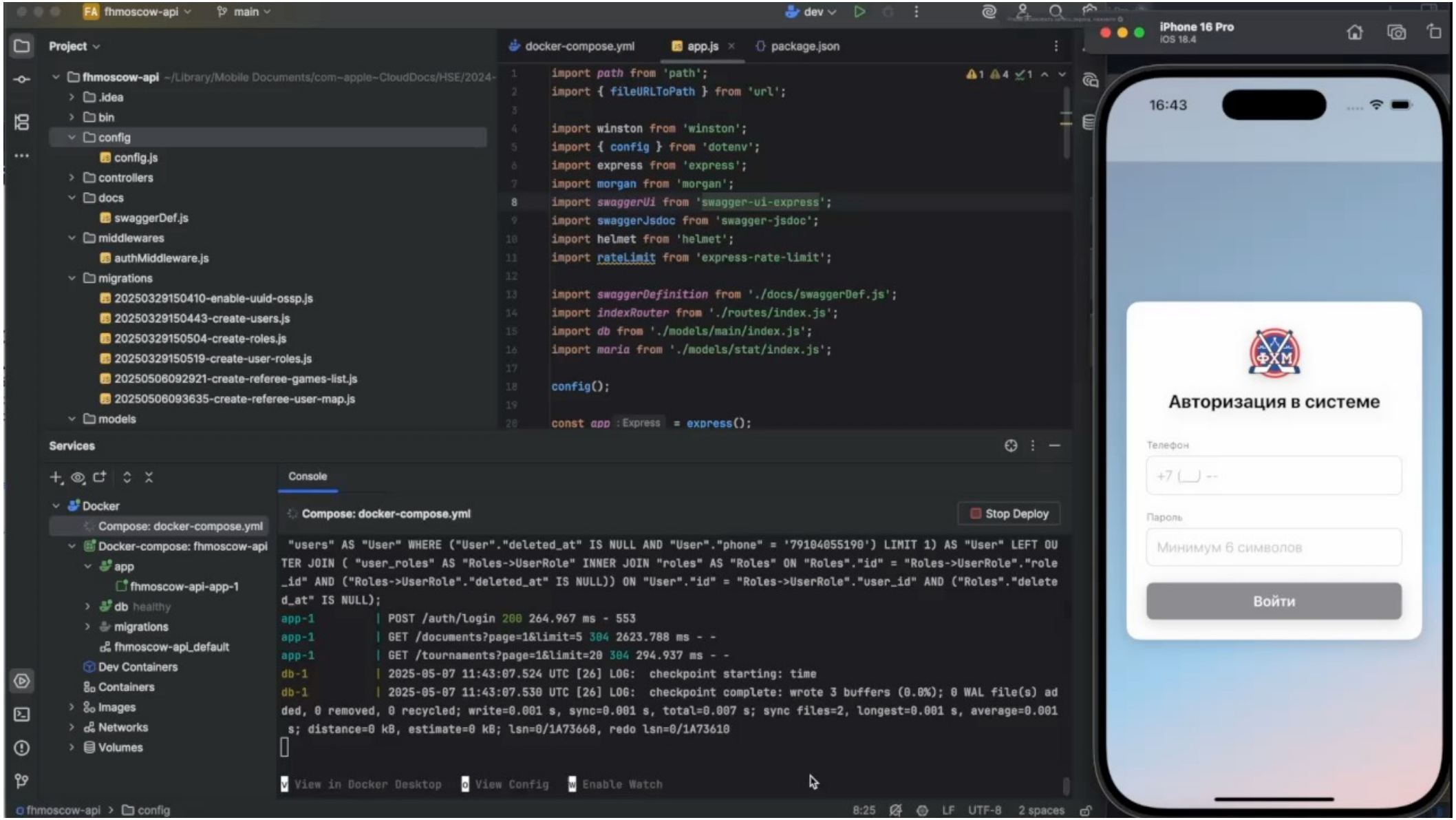
Планы на будущее

Задачи:

- Выполнить реструктуризацию кода клиентской части
- Провести тестирование существующих интеграций
- Доработать функционал в части более точного учета данных технических поражений и иных ситуаций
- Подготовить систему к дальнейшим интеграциям с возможными существующими базами заказчика
- Выполнить разработку веб-интерфейса администратора
- Настроить мониторинг критичных серверов и интеграций

Клиент-серверное приложение для системы управления проведением соревнований

Демонстрация работы проекта



Благодарю за внимание

Буду рад ответить на Ваши вопросы

Национальный исследовательский университет
«Высшая школа экономики»

Факультет
компьютерных наук

Индивидуальный программный курсовой проект

Клиент-серверное приложение для системы управления проведением соревнований

Client-Server Application for the Competition Management System

Исполнитель:
Дробот Алексей Андреевич
студент группы БПИ228

Научный руководитель:
Сосновский Григорий Михайлович
внештатный преподаватель ДПИ ФКН

май 2025