

Санкт-Петербургский государственный университет

ГАБДРАХМАНОВ Азат Райнурович

Выпускная квалификационная работа

Разработка веб-приложения для шахматной школы

Уровень образования: бакалавриат

Направление *02.03.03 «Математическое обеспечение и администрирование
информационных систем»*

Основная образовательная программа *СВ.5162.2021 «Технологии программирования»*

Научный руководитель:
доцент кафедры системного программирования, к. ф.-м. н., Луцив Д. В.

Рецензент:
Старший разработчик, АО «Таймвеб» Лукашук Д. А.

Санкт-Петербург
2025

Saint Petersburg State University

Azat Gabdrakhmanov

Bachelor's Thesis

Development of a web application for a chess school

Education level: bachelor

Speciality *02.03.03 "Software and Administration of Information Systems"*

Programme *CB.5162.2021 "Programming Technologies"*

Scientific supervisor:
C.Sc., docent D. V. Lutsiv

Reviewer:
senior developer at "Timeweb" D. A. Lukashuk

Saint Petersburg
2025

Оглавление

Введение	4
1. Постановка задачи	6
2. Обзор	7
2.1. Обзор существующих решений	7
2.2. Обзор используемых технологий	10
2.3. Требования к разрабатываемой системе	12
3. Описание решения	15
3.1. Архитектура системы	15
3.2. Внутренняя архитектура Backend	17
3.3. Функциональность тренера	18
3.4. Функциональность для ученика	20
3.5. Система учёта и анализа решений	21
3.6. Архитектура развёртывания	22
3.7. Инфраструктура мониторинга и системы логирования .	23
3.8. Интеграция и безопасность	23
4. Апробация	24
4.1. Методика проведения	24
4.2. Участники исследования	24
4.3. Результаты SUS	25
4.4. Результаты функционального опроса	25
4.5. Оценка родителей	25
4.6. Оценка независимых частных тренеров	26
4.7. Метрики использования	27
4.8. Выводы	27
Заключение	28
Список литературы	29

Введение

Шахматы, как один из наиболее популярных интеллектуальных видов спорта в мире, требует от игроков систематической, тщательно организованной тренировки. В процессе обучения шахматистов ключевую роль играет решение специально подобранных позиций, развивающих тактическое зрение и расчет вариантов. Для юных шахматистов и их тренеров особенно важен подбор задач, соответствующих уровню подготовки и направленных на совершенствование конкретных навыков [16].

Существующие системы обучения шахматам предлагают ограниченный набор возможностей для взаимодействия между тренером и учеником. Большинство платформ либо предоставляют общий доступ к задачам без возможности контроля тренером процесса их решения, либо позволяют только выбирать из существующих наборов задач без возможности формирования индивидуальной программы для каждого ученика. У тренера отсутствует возможность анализа результатов ученика в решении задач, что затрудняет выявление проблемных областей в подготовке шахматиста и корректировку тренировочного процесса. Также тренеру при работе с детьми важно на ранних стадиях выявить у ученика предрасположенность к решению задач наугад, что в будущем будет сложно искоренять, а инструменты для отслеживания хода действий ученика во время решения задач сильно ограничены.

В связи с этим возникает потребность в специализированной системе, которая позволит тренерам создавать наборы шахматных задач, контролировать процесс их выполнения, анализировать результаты и выстраивать индивидуальные тренировочные планы для учеников. Такая система должна предоставлять интерфейс для формирования домашних заданий с заданными параметрами сложности и тематики, а также инструменты для мониторинга выполнения задач обучающимся.

Данная работа посвящена разработке веб-приложения Chess Dream, предназначенного для поддержки тренировочного процесса шахматистов. Приложение реализует возможность создания персонализированных наборов задач, генерации домашних заданий в соответствии с па-

раметрами сложности и тематики, отслеживания выполнения заданий и анализа результатов. В отличие от существующих решений, Chess Dream обеспечивает возможность тесного взаимодействия между тренером и учеником, возможность тренеру видеть все успехи ученика и корректировать индивидуальный план.

1. Постановка задачи

Целью работы является разработка веб-приложения для поддержки тренировочного процесса шахматистов, обеспечивающего взаимодействие между тренерами и учениками. Для её выполнения были поставлены следующие задачи:

1. Провести обзор существующих решений для обучения шахматам и выявить их ограничения.
2. Сформировать требования для приложения.
3. Спроектировать архитектуру приложения Chess Dream, включающую клиентскую часть, серверную часть и базу данных.
4. Разработать интерфейс тренера с возможностями создания и назначения персонализированных домашних заданий на основе заданных параметров сложности и тематики.
5. Создать интерфейс ученика для решения назначенных задач и получения обратной связи.
6. Разработать интерфейс для анализа результатов и прогресса учеников, предоставляющий тренерам информацию для корректировки обучения.
7. Провести апробацию приложения Chess Dream с участием шахматных тренеров и их учеников.

2. Обзор

Современным шахматным платформам свойственна либо широта функциональности (турниры, игры, анализ партий), либо возможность решать задачи, но при этом отсутствует инструмент, который позволял бы тренерам массово формировать задания и сразу получать результаты о времени решения, количестве попыток и типах ошибок. Есть разные школы и методики преподавания [2], например, автор данной работы и тренеры шахматной школы UfaChess — сторонники усиленного изучения тактики — столкнулись с отсутствием узкоспециализированных платформ для решения тактических задач и их анализа.

В данном разделе представлен обзор существующих решений для обучения шахматам, а также технологий, используемых для разработки системы Chess Dream. Цель обзора — выявить достоинства и недостатки существующих решений, определить выбор технологического стека и ключевые требования к разрабатываемой системе.

2.1. Обзор существующих решений

В настоящее время существует значительное количество платформ и систем, ориентированных на обучение шахматам.

2.1.1. Lichess

Lichess¹ — бесплатная шахматная платформа с открытым исходным кодом, имеющая раздел тактических задач.

Преимущества:

- Открытый исходный код и бесплатное использование.
- Большая база шахматных задач (более 1 000 000).
- Рейтинговая система для задач и онлайн-игр.
- Возможность создания командных турниров.

¹Lichess: <https://lichess.org> (дата обращения: 25 мая 2025 г.)

- Большое сообщество и хорошие возможности для шахматного развития.

Недостатки:

- Ограниченные возможности анализа результатов обучающихся.
- Основной практикой для решения задач стало создание студий², где создаются вручную задачи, но создать таким образом большое количество задач занимает много времени.

На сегодняшний день в региональных шахматных школах Lichess является наиболее популярной платформой: благодаря открытой лицензии и активному сообществу она остаётся доступной и не подвержена блокировкам³.

При принятии решения о создании Chess Dream встал вопрос: имеет ли смысл разрабатывать собственную платформу с нуля, а не «патчить» Lichess? После анализа кода и архитектуры Lichess было принято решение использовать лишь отдельные модули этого проекта в качестве библиотек, а не модифицировать его целиком. Основные аргументы против полного форка Lichess:

- **Сложность онбординга.** Объём кода Lichess превышает несколько десятков тысяч строк, а многоуровневая архитектура требует значительных усилий для понимания и поддержки.
- **Избыточная функциональность.** Большая часть функциональности Lichess ориентирована на проведение турниров, онлайн-игру и анализ партий — эти модули избыточны для системы, ориентированной только на решение задач и их анализ.
- **Стек технологий.** Основной язык разработки — Scala. Найти опытных Scala-разработчиков сложнее и дороже, чем специалистов по C# и JavaScript/TypeScript, используемым в Chess Dream.

²Студии Lichess - инструмент, позволяющий создавать наборы позиций для анализа или решения. Подробнее: <https://lichess.org/study> (дата обращения: 25 мая 2025 г.)

³Исходный код Lichess: <https://github.com/lichess-org/lichess> (дата обращения: 25 мая 2025 г.)

2.1.2. Chess.com

Chess.com⁴ — крупнейший шахматный портал, предлагающий раздел с тактическими головоломками и обучающим контентом.

Преимущества:

- Обширная база задач различной сложности.
- Структурированные уроки по тематическим разделам.
- Система для анализа партий.
- Наличие мобильного приложения.

Недостатки:

- В РФ не работает без VPN.
- Большинство обучающих функций доступны только в платной версии.
- Отсутствие специализированного интерфейса для тренеров.
- Ограниченные возможности для создания персонализированных заданий.

2.1.3. ChessKid

ChessKid⁵ — платформа, ориентированная на обучение детей шахматам.

Преимущества:

- Адаптированный для детей интерфейс.
- Игровые элементы в обучении.
- Безопасная среда для детей.

⁴Chess.com: <https://chess.com> (дата обращения: 25 мая 2025 г.)

⁵ChessKid: <https://chesskid.com> (дата обращения: 12 мая 2025 г.)

- Наличие функций родительского контроля.

Недостатки:

- Ограниченная база шахматных задач в бесплатной версии.
- Невозможность создания персонализированных наборов заданий тренерами.
- Ограниченные аналитические возможности.

2.1.4. StepChess

StepChess — платформа, ориентированная на работу тренеров и групповые занятия. Позволяет:

- Создавать домашние задания на основе задач из базы Lichess.
- Отслеживать для каждого ученика точное время решения позиции и количество попыток.
- Просматривать историю совершённых ходов и анализировать ошибки.

Недостатки:

- Нельзя видеть какие именно попытки делал ученик во время решения задач.
- Нельзя импортировать из Личесса сразу несколько десятков задач по определенной теме, чтобы создать из них домашнее задание.

2.2. Обзор используемых технологий

Для обеспечения производительности, надёжности и масштабируемости приложения были выбраны следующие инструменты:

- **React** — компонентный фреймворк для построения динамического интерфейса; обеспечивает быстрый рендеринг и удобное разделение логики представления, также имеет готовые библиотеки для работы с шахматными досками[14].
- **ASP.NET Core Web API** — кроссплатформенная серверная платформа с низкой задержкой, встроенной поддержкой DI и надёжной системой маршрутизации[4].
- **Entity Framework Core** — обеспечивает автоматическое отображение моделей на схему PostgreSQL, упрощает работу с данными и миграциями[6].
- **PostgreSQL** — высокопроизводительная реляционная СУБД с богатым набором типов и масштабируемыми возможностями репликации[12].
- **OAuth2 через Lichess** — стандартный протокол авторизации, позволяющий использовать готовую инфраструктуру учётных записей и исключить хранение паролей[3].
- **Nginx** — высокопроизводительный веб-сервер и балансировщик нагрузки, обеспечивает распределение запросов между экземплярами приложения[10].
- **Docker** — контейнеризация сервисов для единообразного окружения разработки, тестирования и деплоя[7].
- **GitLab CI/CD** — автоматизация сборки, тестирования и доставки кода; гарантирует воспроизводимость процессов[8].
- **Prometheus & Grafana** — сбор и визуализация метрик производительности и доступности, позволяет оперативно выявлять узкие места[13], [9].
- **Serilog** — структурированное логирование с возможностью экспорта в OpenSearch для последующего анализа[15].

- **OpenSearch** — распределённая платформа поиска и аналитики логов, обеспечивает быстрый поиск и агрегацию больших объёмов данных[11].

2.3. Требования к разрабатываемой системе

На основании поставленных задач (см. раздел 1) сформулированы следующие требования к системе Chess Dream.

2.3.1. Функциональные требования

1. Тренер должен иметь возможность создавать домашние задания, выбрав:
 - тему (тематическая группа задач),
 - диапазон рейтинга задач,
 - количество задач.
2. Система должна автоматически подбирать задачи по заданным параметрам и исключать повторение ранее назначенных заданий.
3. Тренер должен получать детализированную статистику по каждому заданию: время решения, число ошибок и попыток учеников в режиме реального времени.
4. Ученик должен иметь личный кабинет с отображением назначенных домашних заданий, статуса их выполнения и истории результатов.
5. Ученик должен иметь возможность решать задачи внутри веб-интерфейса с сохранением всех ходов и времени выполнения для последующего анализа.
6. Система должна предоставлять интерфейс для детального анализа попыток решения задач с возможностью воспроизведения последовательности ходов ученика.

7. Пользовательская авторизация должна осуществляться с использованием учётных записей Lichess.

2.3.2. Нефункциональные требования

- Система должна поддерживать одновременную работу не менее 100 активных пользователей без деградации производительности для обеспечения групповых занятий в клубах.
- Интерфейс должен корректно отображаться и функционировать в последних версиях Chrome, Firefox и Safari.
- Интерфейс ученика должен быть максимально простым и понятным для детской аудитории.
- Все данные пользователей и их результаты должны надежно сохраняться с регулярным резервным копированием.

2.3.3. Пользовательские сценарии

Система Chess Dream разработана для поддержки следующих пользовательских сценариев в практике шахматного обучения:

Сценарий 1: Работа в шахматном клубе

Во время групповых занятий в клубе тренер организует работу учеников следующим образом: половина группы играет партии, а половина решает тактические задачи по пройденной теме на Chess Dream. Тренер после решения задач одной из групп проверяет их и выявляет ошибки учеников. После смены активности (игроки переходят к решению задач, а решавшие задачи начинают играть) тренер может создать дополнительный набор задач для учеников, показавших слабые результаты, используя это как мотивирующий фактор для ученика.

Сценарий 2: Домашние задания и онлайн-анализ

Тренер назначает ученику набор домашних заданий через систему, которые ученик самостоятельно решает дома в удобное время. На следующем онлайн-занятии тренер открывает интерфейс анализа результатов с демонстрацией экрана и разбирает с учеником допущенные ошибки.

ки, анализирует неудачные попытки решения и объясняет правильные варианты, перемещая фигуры на интерактивной доске для наглядности.

Сценарий 3: Предтурнирная подготовка

Перед участием в турнире тренер назначает ученику большой объем относительно простых тактических задач (обычно около 100 задач) для «разминки» и настройки на игру. Ученик решает эти задачи утром в день турнира или вечером накануне, что, согласно практическому опыту тренеров UfaChess, значительно улучшает игровые результаты учеников.

3. Описание решения

3.1. Архитектура системы

Система реализована по многослойной клиент–серверной модели, обеспечивающей разделение ответственности и упрощение поддержки:

- **Уровень представления (Presentation):** React-приложение, формирующее интерфейсы тренера и ученика.
- **Прикладной уровень (Application):** ASP.NET Core Web API, обеспечивающее обработку HTTP-запросов и выполнение бизнес-логики.
- **Уровень данных (Data):** доступ к PostgreSQL через Entity Framework Core.

Каждый слой взаимодействует с соседними через чётко определённые интерфейсы, что облегчает тестирование и масштабирование.

Для визуализации архитектуры использована модель C4:

3.1.1. Контекстная диаграмма

Показывает границы системы, основные роли и внешние системы.



Рис. 1: Контекстная диаграмма системы Chess Dream

3.1.2. Контейнерная диаграмма

Демонстрирует основные контейнеры (клиент, сервер, база данных) и их взаимодействие.

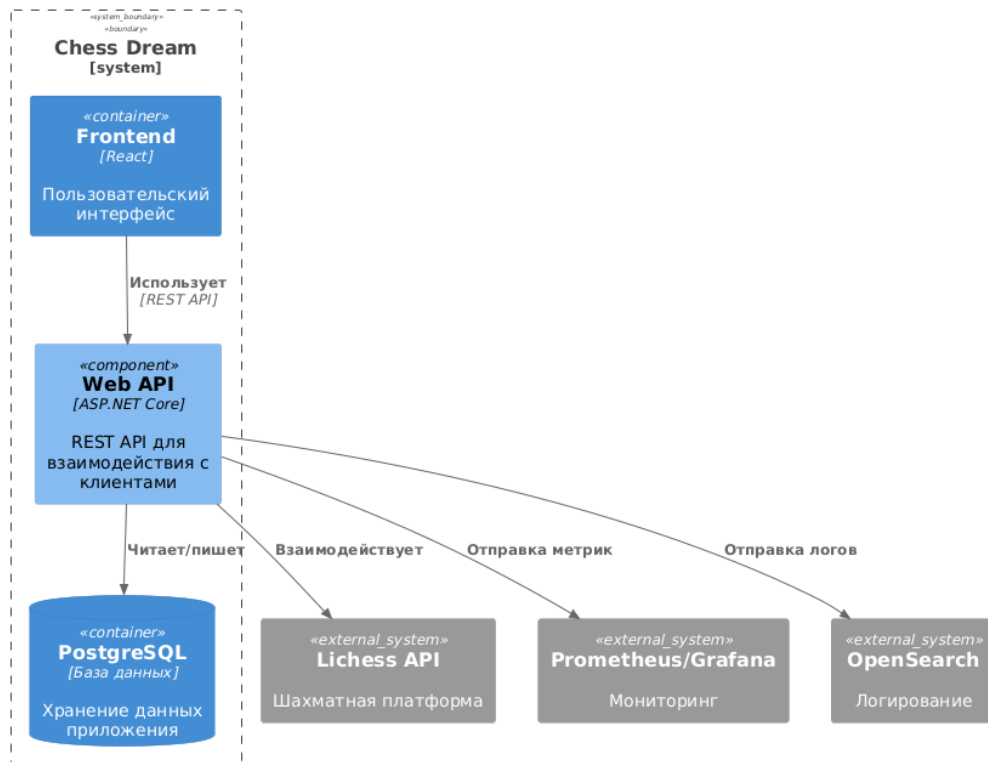


Рис. 2: Контейнерная диаграмма системы Chess Dream

3.2. Внутренняя архитектура Backend

Серверная часть реализована в виде монолитного ASP.NET Core приложения и организована по следующей схеме:

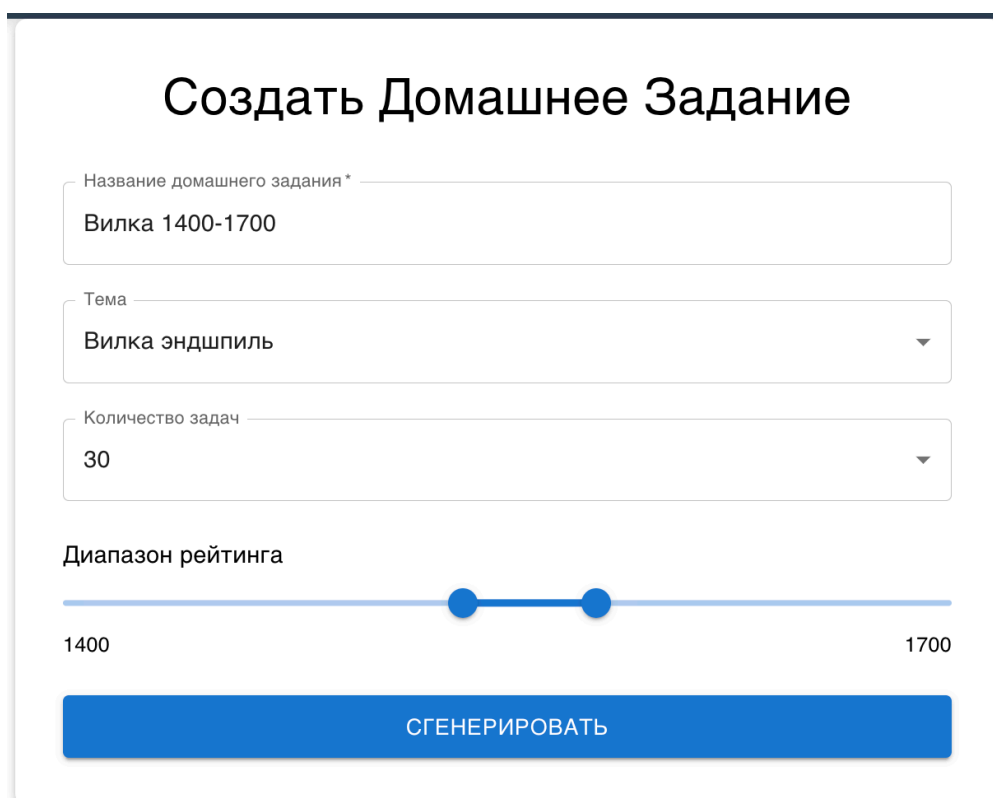
- **Controllers** Принимают HTTP(S)-запросы, выполняют аутентификацию и базовую валидацию данных.
- **Business Logic** Реализуют основную логику работы приложения: подбор и генерацию заданий, сбор и агрегацию статистики, применение правил доступа.
- **Data Access** Отвечают за взаимодействие с базой данных (PostgreSQL + Entity Framework Core), обеспечивая выполнение CRUD-операций и транзакций.

Дополнительно имеются обёртки для интеграции с внешними сервисами (например, OAuth2-клиент для Lichess), встроенные механизмы логирования и сбора метрик.

3.3. Функциональность тренера

Тренер взаимодействует с системой через веб-интерфейс, который предоставляет следующие возможности:

1. **Создание домашнего задания:** Интерфейс позволяет указать название, тему, количество задач и диапазон рейтинга для автоматической генерации домашнего задания. Система самостоятельно подбирает подходящие задачи из базы данных, учитывая заданные параметры.



Создать Домашнее Задание

Название домашнего задания *
Вилка 1400-1700

Тема
Вилка эндшпиль

Количество задач
30

Диапазон рейтинга

1400 1700

СГЕНЕРИРОВАТЬ

Рис. 3: Интерфейс создания задач

2. **Просмотр статистики:** Доступен детальный анализ прогресса по каждому ученику, включая время, затраченное на решение задач, количество попыток и типичные ошибки.

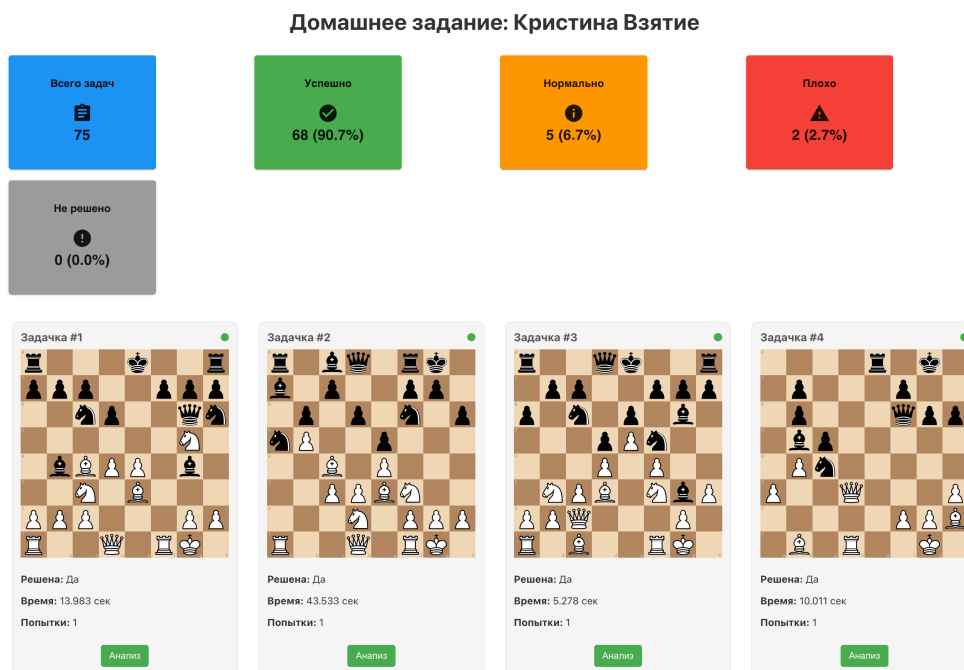


Рис. 4: Интерфейс просмотра домашнего задания ученика

3. Просмотр ходов ученика: Тренер имеет возможность увидеть неудачные попытки ученика при решении задачи.

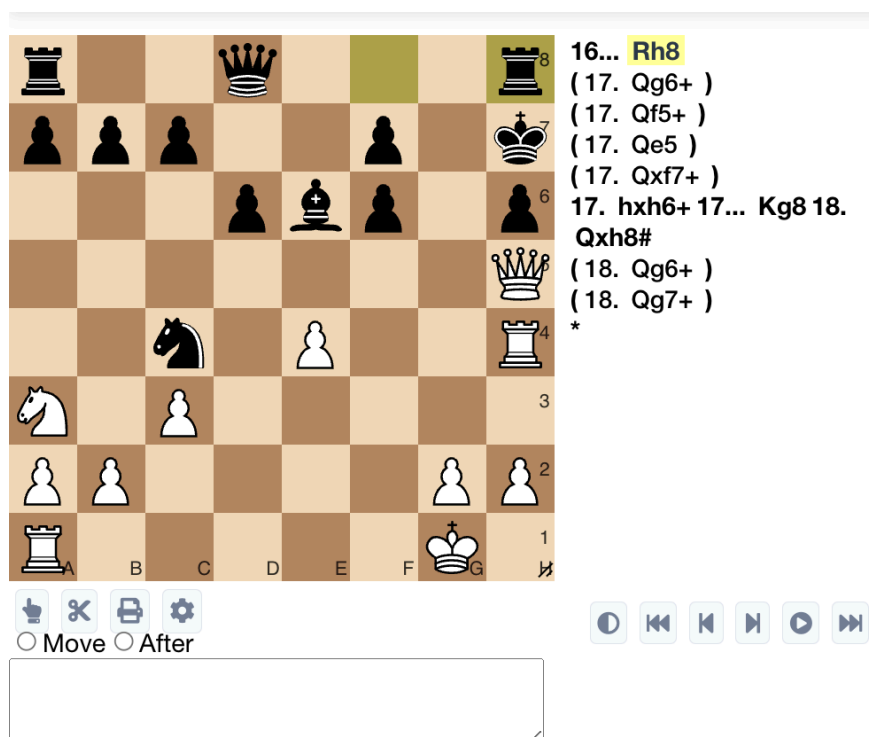


Рис. 5: Интерфейс анализа ходов ученика

3.3.1. Алгоритм подбора задач

Для генерации домашнего задания используется следующий подход:

1. Выбираются задачи с рейтингом в интервале $[\text{minElo}, \text{maxElo}]$.
2. Исключаются уже назначенные тренером задачи.
3. Из оставшегося набора происходит псевдослучайная выборка.

3.4. Функциональность для ученика

Ученику доступны следующие возможности:

- **Личный кабинет:** Интерфейс отображает список назначенных домашних заданий, прогресс их выполнения и общую статистику.

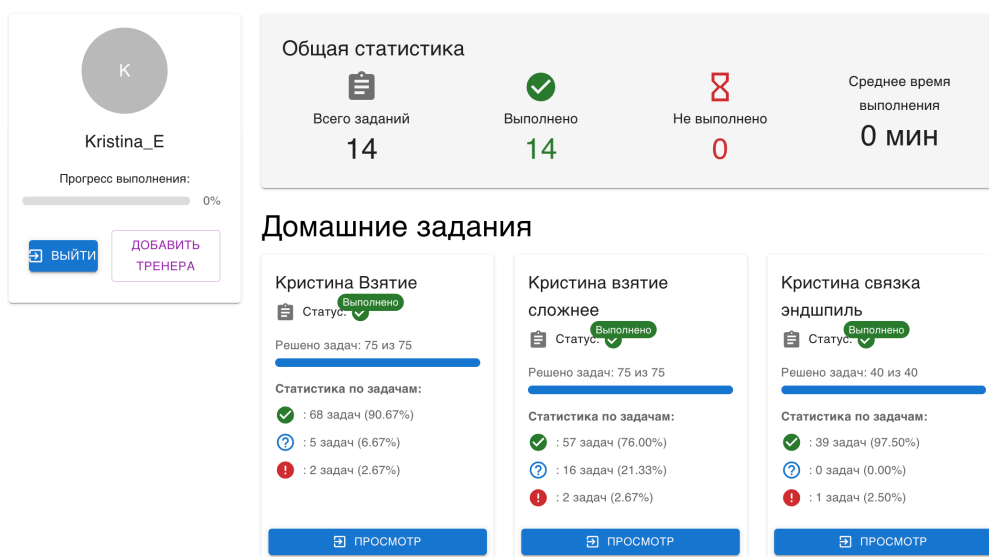


Рис. 6: Интерфейс личного кабинета ученика

- **Интерфейс решения задач:** Интерактивная шахматная доска позволяет решать задачи, перемещая фигуры.

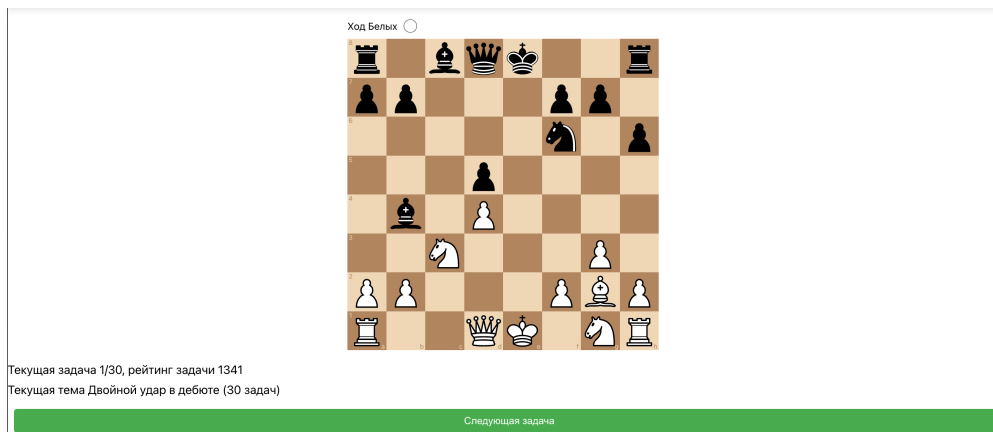


Рис. 7: Интерфейс решения задач ученика

- **Анализ решений:** После выполнения задания ученик видит свою краткую статистику, показывающую правильно и ошибочно решённые задачи.

3.5. Система учёта и анализа решений

Для обеспечения детального анализа прогресса ученика реализована система отслеживания действий при решении задач:

1. **Запись каждого хода:** При каждом ходе ученика система фиксирует сам ход, его порядковый номер и правильность.
2. **Отслеживание времени:** Система записывает время, затраченное на каждый ход и на решение задачи в целом.
3. **Учёт решённых задач:** После успешного решения задача логируется в таблице `solvedpuzzle` с данными о времени и попытках.
4. **Агрегация статистики:** На основе собранных данных формируются отчёты для тренера по таблицам `solvedpuzzle` и `attemptpuzzle`.

Основные сущности базы данных:

- `puzzle` — задачи (FEN, рейтинг, тема, решение).
- `homework` — домашние задания (название, владелец, параметры).

- `homeworkpuzzle` — связь заданий и задач с порядком.
- `solvedpuzzle` — успешные решения с метриками.
- `attemptpuzzle` — детализация каждого хода.

3.6. Архитектура развёртывания

Для обеспечения высокой доступности и отказоустойчивости системы реализована современная инфраструктура развёртывания:

1. **Контейнеризация:** Все компоненты (frontend, backend, БД, мониторинг) упакованы в Docker-контейнеры.
2. **Горизонтальное масштабирование:** Запущены две реплики API и одна реплика БД.
3. **Балансировка нагрузки:** NGINX, настроенный на алгоритм round-robin с health checks, распределяет запросы между репликами.
4. **CI/CD:** GitLab CI прогоняет линтеры, запускает модульные и интеграционные тесты, затем автоматически деплоит в staging или production.
5. **Автоматическое восстановление:** Docker Compose перезапускает сбойные контейнеры без ручного вмешательства.
6. **Резервное копирование:** Настроены регулярные бэкапы состояния базы данных, хранящиеся на других удаленных серверах для обеспечения сохранности данных.

Такая архитектура обеспечивает:

- высокую доступность при сбоях отдельных компонентов;
- гибкость масштабирования при росте нагрузки;
- упрощённое развёртывание одной командой;
- надёжное восстановление без простоя.

3.7. Инфраструктура мониторинга и системы логирования

Для наблюдаемости и оперативного обнаружения проблем реализована система мониторинга и логирования:

- **Serilog**: структурированное логирование в формате JSON.
- **OpenSearch**: централизованное хранение и анализ логов.
- **Prometheus**: сбор метрик производительности и бизнес-метрик.
- **Grafana**: визуализация ключевых показателей и настройка алертов.

3.8. Интеграция и безопасность

OAuth2 (Lichess): интеграция с Lichess для аутентификации пользователей через протокол OAuth2, позволяющая избежать хранения паролей в системе.

4. Апробация

4.1. Методика проведения

Для оценки удобства использования и качества системы Chess Dream были использованы:

- **System Usability Scale (SUS):** 10 утверждений с пятибалльной шкалой для количественного измерения юзабилити.
- **Функциональный опрос:** 7 утверждений по ключевым возможностям системы с оценкой от 1 до 5.
- **Открытые вопросы:** сбор качественной обратной связи о сильных и слабых сторонах работы.

SUS была выбрана как основной инструмент оценки удобства использования системы, поскольку представляет собой надежный метод измерения оценки удобства использования системы. Исследования показывают, что SUS эффективно различает удобные и неудобные системы и высоко коррелирует с другими инструментами оценки систем[1]. Средним показателем SUS считается 68 баллов, результаты выше этого значения считаются выше среднего[5].

Опрос проводился после практического использования платформы в учебном процессе. Данные собирались в формате анкеты.

4.2. Участники исследования

В исследовании приняли участие:

- **Основной заказчик:** шахматная школа UfaChess
- **Тренеры UfaChess:** 10 человек
- **Независимые частные тренеры:** 7 человек
- **Ученики:** 15 человек
- **Родители:** 7 человек (для оценки внешнего восприятия)

4.3. Результаты SUS

SUS-опрос проводился только среди тренеров:

- Тренеры UfaChess (10 человек): 78.5 баллов из 100
- Независимые частные тренеры (7 человек): 82.3 баллов из 100

4.4. Результаты функционального опроса

4.4.1. Интерфейс тренера

В таблице 1 приведены средние оценки тренеров по ключевым критериям.

Таблица 1: Средние оценки тренеров по функциональным критериям

Критерий	Средняя оценка (1–5)
Интуитивность интерфейса создания заданий	4.6
Автоматическая генерация заданий	4.4
Просмотр статистики	4.5
Удобство навигации в личном кабинете	4.2
Общая удовлетворённость	4.5

4.4.2. Интерфейс ученика

Таблица 2: Средние оценки учеников по функциональным критериям

Критерий	Средняя оценка (1–5)
Удобство решения задач	4.3
Понятность навигации	4.0
Общая удовлетворённость	4.2

4.5. Оценка родителей

Родители оценили систему с точки зрения внешнего восприятия:

- Средняя оценка (1–5): 4.3

- **Что понравилось:**

- Простота восприятия результатов ребенка
- Наглядность представленной статистики

- **Что не понравилось:**

- Отсутствие мобильной версии

Родители имеют желание получать краткие отчеты по результатам решения задач своих детей в мессенджерах раз в неделю/месяц. Данная функциональность находится в разработке.

4.6. Оценка независимых частных тренеров

Опрос среди независимых частных тренеров показал следующие результаты:

- **Средняя SUS-оценка (0–100):** 82.3 баллов из 100

- **Средняя оценка функциональных критериев (1–5):** 4.4

- **Что понравилось:**

- Удобство создания и настройки заданий
- Подробная аналитика ошибок и попыток

- **Что не понравилось:**

- Отсутствие возможности задавать конкретный таймер ученику

Частные тренеры зачастую имеют большое количество занятий в группах, поэтому у них есть желание задавать домашнее задание с определенным таймером, который будет отображаться ученику во время решения домашнего задания. Данная функциональность также была интересна для заказчика, но не была приоритетной. Разработка планируется вне рамок данной дипломной работы.

4.7. Метрики использования

В процессе использования системы на момент апробации были достигнуты следующие показатели:

- **Зарегистрировано пользователей:** 150 человек
- **Количество решённых задач:** 15400 задач
- **Общее число сделанных ходов:** 44300 попыток

4.8. Выводы

На основе результатов опроса можно заключить, что система Chess Dream получила высокую оценку удобства и функциональности среди всех ключевых групп пользователей, включая заказчика (UfaChess), независимых частных тренеров, учеников и родителей.

На основе опроса пользователей были сформированы приоритетные направления для дальнейшего развития.

Заключение

В ходе выполнения выпускной квалификационной работы по разработке веб-приложения Chess Dream были достигнуты следующие результаты:

- Выполнен обзор существующих решений с анализом их ограничений.
- Сформированы требования к приложению на основе обсуждения с тренерами шахматных школ и анализа пользовательских сценариев.
- Спроектирована архитектура приложения.
- Реализован интерфейс тренера: создание и автоматическая генерация домашних заданий по теме и диапазону рейтинга, назначение домашнего задания ученику и просмотр детализированной статистики выполнения.
- Реализован интерфейс ученика: личный кабинет, динамическое решение шахматных задач с фиксацией ходов и времени, анализ ошибок.
- Реализован интерфейс для анализа результатов и прогресса учеников.
- Проведена апробация платформы среди основных групп пользователей.

Исходный код проекта является закрытым.

Направления дальнейшей работы:

- Расширение геймификации и рейтинговой системы учеников.
- Интеграция модуля загрузки и анализа PGN-файлов партий.
- Разработка сервиса уведомлений для родителей и тренеров.

Список литературы

- [1] Bangor Aaron, Kortum Philip T., Miller James T. An Empirical Evaluation of the System Usability Scale // International Journal of Human-Computer Interaction. — 2008. — Vol. 24, no. 6. — P. 574–594. — URL: <https://www.tandfonline.com/doi/abs/10.1080/10447310802205776> (дата обращения: 25 мая 2025 г.).
- [2] Polgar Susan, Truong Paul. Chess Tactics for Champions: A step-by-step guide to using tactics and combinations the Polgar way. — URL: <https://www.amazon.com/Chess-Tactics-Champions-step-step/dp/081293671X> (дата обращения: 25 мая 2025 г.).
- [3] Richer Justin, Sanso Antonio. OAuth 2 in Action. — URL: <https://www.manning.com/books/oauth-2-in-action> (дата обращения: 25 мая 2025 г.).
- [4] Sanctis Valerio De. Building Web APIs with ASP.NET Core. — URL: <https://www.manning.com/books/building-web-apis-with-asp-net-core> (дата обращения: 25 мая 2025 г.).
- [5] Sauro Jeff. Measuring Usability with the System Usability Scale (SUS). — URL: <https://measuringu.com/sus/> (дата обращения: 25 мая 2025 г.).
- [6] Smith Jon P. Entity Framework Core in Action. — URL: <https://www.manning.com/books/entity-framework-core-in-action> (дата обращения: 25 мая 2025 г.).
- [7] Документация Docker. — URL: <https://docs.docker.com/> (дата обращения: 25 мая 2025 г.).
- [8] Документация GitLab CI/CD. — URL: <https://docs.gitlab.com/ee/ci/> (дата обращения: 25 мая 2025 г.).

- [9] Документация Grafana. — URL: <https://grafana.com/docs/grafana/latest/> (дата обращения: 25 мая 2025 г.).
- [10] Документация Nginx. — URL: <https://nginx.org/ru/docs/> (дата обращения: 25 мая 2025 г.).
- [11] Документация OpenSearch. — URL: <https://opensearch.org/docs/> (дата обращения: 25 мая 2025 г.).
- [12] Документация PostgreSQL. — URL: <https://www.postgresql.org/docs/current/> (дата обращения: 25 мая 2025 г.).
- [13] Документация Prometheus. — URL: <https://prometheus.io/docs/introduction/overview/> (дата обращения: 25 мая 2025 г.).
- [14] Документация React. — URL: <https://reactjs.org/docs/getting-started.html> (дата обращения: 25 мая 2025 г.).
- [15] Документация Serilog. — URL: <https://serilog.net/> (дата обращения: 25 мая 2025 г.).
- [16] Костенюк Александра, Костенюк Наталия. Как научить шахматам. — URL: <https://irorb.ru/wp-content/uploads/2019/12/kostenyuk-kak-nauchit-shahmatam1.pdf> (дата обращения: 25 мая 2025 г.).