1. Информация о проекте

1.1. Задача

Создание интерфейса онлайн-платформы для подготовки к олимпиаде по спортивному программированию.

1.2. Что удалось реализовать

1. Лобби
    a. Онлайн статус участников команды
    b. Возможность выбрать контест для тренировки
2. Тренировка
    a. Онлайн статус участников команды и управляющий редактором кода
    b. Кнопка для передачи управления между участниками команды
    c. Время до окончания соревнования
    d. Список задач с функцией назначения их определённым участникам
    e. Статус задачи (решена верно, решена неверно, не отправлена на проверку)
    f. Чат с комментариями для обсуждения и обмена идеями внутри команды
    g. Вкладка с результатами отправленных решений
    h. Редактор кода с подсветкой синтаксиса для разных языков программирования
3. Лидерборд
    a. Счёт и статусы по задачам других команд
    b. Фильтр по названию команды / логин участника

2. Архитектура и технологи

2.1. Структура приложения

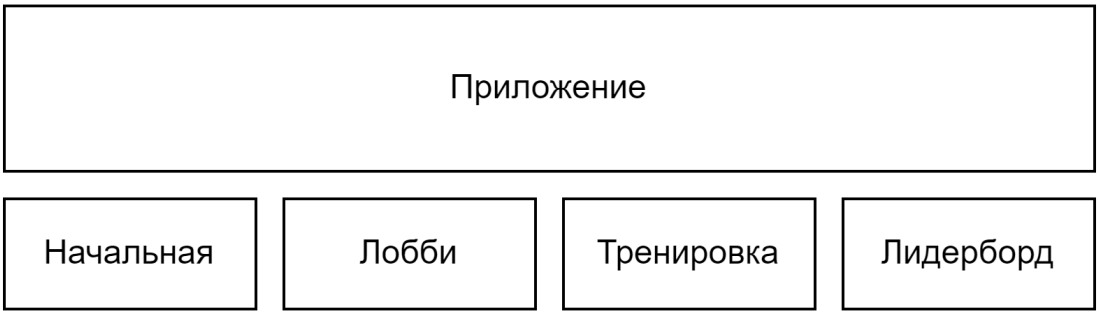
Приложение состоит из **страниц** - элементов, отвечающих за определенный пользовательский **сценарий**.

Страницы состоят из **виджетов** - элементов, отвечающих за определенный **функционал** страницы.

Виджеты состоят из **компонентов** - элементов, отвечающих за определенную **функцию** виджета.

2.2. Страницы

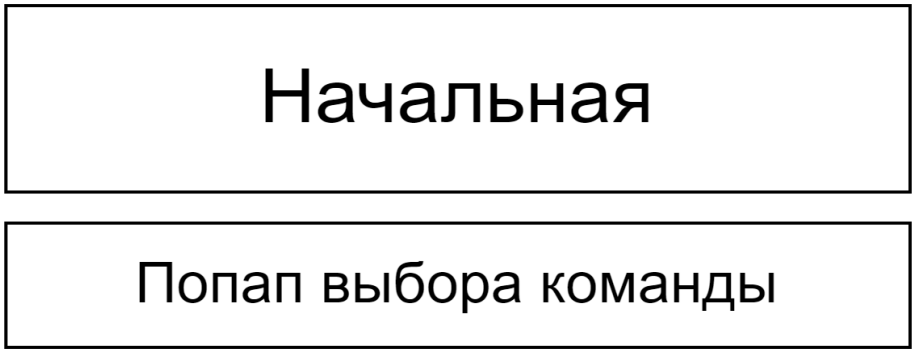
Страница	**Сценарий**
Начальная	Выбор команды
	Переход в режим групповых тренировок
Лобби	Выбор контеста
	Инициация тренировки
Лидерборд	Командное решение задач выбранного контеста
Лидерборд	Просмотр рейтинга команд выбранного контеста на текущее время от начала контеста



2.3. Виджеты

Начальная

Виджет	**Функционал**
Попап выбора команды	Выбор команды для участия в тренировке



Лобби

Виджет	**Функционал**
Выбор контеста	Выбор контеста для тренировки
	Инициация тренировки
Переход в чат	Переход в телеграмм-чат **Сообщество CodeRun // чат для общения**
Текущая команда	Отображение названия текущей команды, выбранной на странице **Начальная**
Участники онлайн	Динамической отображение аватаров участников команды, зашедших в **Лобби**

Тренировка + Лидерборд

Виджет	**Функционал**
Таб	Переключение между страницами Тренировка и Лидерборд
Участники онлайн	Динамической отображение аватаров участников команды, зашедших в Тренировку
	Индикация наличия управления у каждого пользователя
Взятие управления	Передача управления пользователю по нажатии на кнопку
	Индикация принадлежности управления текущему пользователю
Таймер	Индикация принадлежности управления текущему пользователю
Завершение тренировки	Завершение текущей сессии
	Редирект в Лобби

Тренировка

Виджет	**Функционал**
Список задач	Вывод названий задач
	Переключение между задачами
	Назначение ответственного на задачу
	Индикация ответственного на задачу
	Индикация статуса решения задачи *(не решена, решена неверно, решена верно)*
	Индикация количества попыток решения задачи
Комментарии к задаче	Вывод комментариев, добавленных к выбранной задаче
	Добавление комментария
	Удаление комментария
Инфомация о задаче	Вывод условия выбранной задачи
	Вывод списка посылок выбранной задачи
	Вывод детализации выбранной посылки
	Индикация статуса решения выбранной задачи *(не решена, решена неверно, решена верно)*
Редактор	Выбор компилятора
	Написание кода - решения задачи
	Отправка кода - решения задачи
	Индикация отправки уже отправленного ранее кода

Лидерборд

Виджет	**Функционал**
Таблица результатов	Отображение таблицы-рейтинга команд, принявших участие в данном контесте
	Поиск команды по названию и списку участников

2.4. Связь с бекендом

http -> socket

Отправка данных, требующих сохранение в базу данных, одним клиентом по **http**.

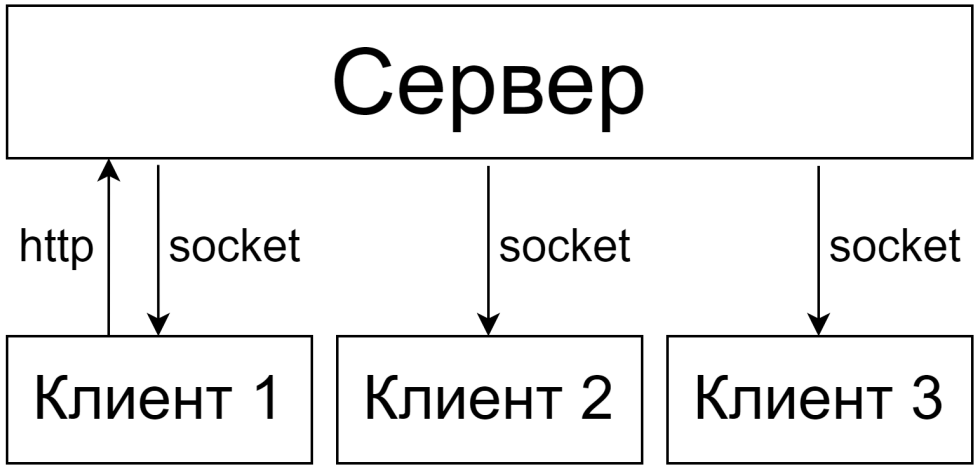
Получение данных остальными клиентами по socket.

Лобби

http out	**socket in**
Отправка данных для создания сессии	Получение информации о созданной сессии

Тренировка

http out	**socket in**
Отправка комментария к задач	Получение добавленного комментария к задаче
Отправка id комментария для удаления	Получение id удаленного комментария
Отправка кода на проверку	Получение статуса проверки решения
	Получение обновленного статуса решения задачи
	Получение обновленного числа попыток решения задачи



socket -> socket

Отправка данных, требующих сохранение в redis, одним клиентом по **socket**.

Получение данных остальными клиентами по **socket**.

Лобби + Тренировка + Лидерборд

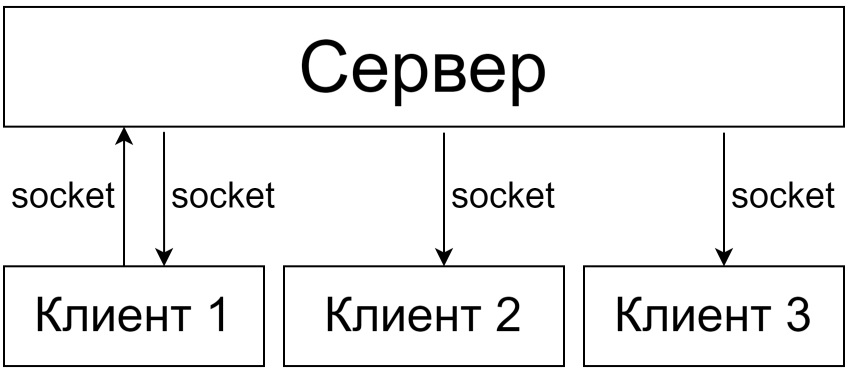
socket out	**socket in**
Отправка данных текущего пользователя при подключении	Получение данных подключившегося пользователя

Лобби

socket out	**socket in**
Отправка id выбранного контеста	Получение id выбранного контеста

Тренировка

socket out	**socket in**
Отправка кода в редакторе	Получение кода в редакторе
Отправка id пользователя, взявшего управление	Получение id пользователя, взявшего управление
Отправка пользователя, назначенного на задачу	Получение пользователя, назначенного на задачу
Отправка выбранного компилятора к задаче	Получение выбранного компилятора к задаче



http -> http

Запрос текущего состояния при загрузке страницы или при переключении задачи отдельно каждым клиентом.

Начальная

- Получение списка команд текущего пользователя
- Получение открытой сессии выбранной команды

Лобби

- Получение списка контестов
- Получение текущих онлайн участников
- Получение id выбранного контеста

Тренировка + Лидерборд

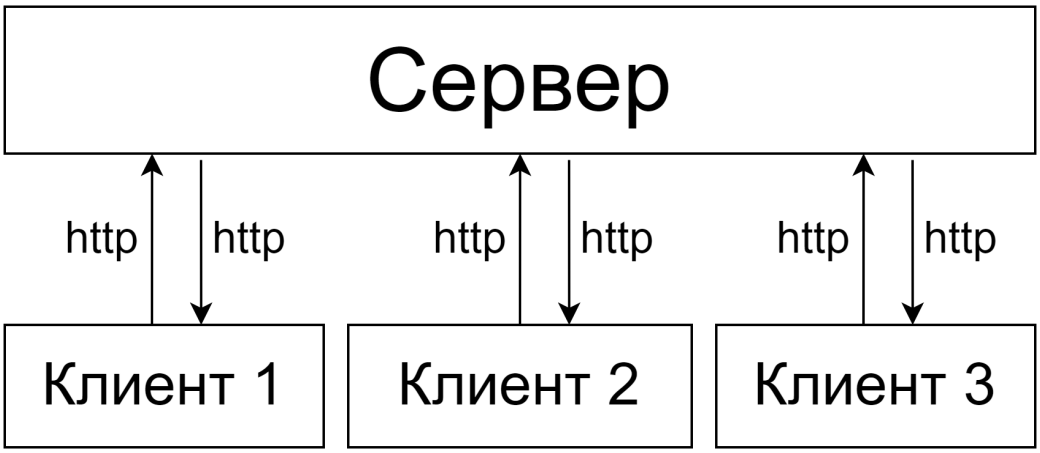
- Получение информации о текущем авторизованном пользователе из Яндекс-паспорта
- Получение id текущего пользователя, управляющего редактором
- Получение текущих онлайн участников
- Получение времени до конца контеста

Тренировка

- Получение списка задач выбранного контеста
- Получение условия выбранной задачи
- Получение списка посылок к выбранной задаче
- Получение текущего состояния кода в редакторе к выбранной задаче
- Получение текущего выбранного компилятора к задаче
- Получение комментариев к выбранной задаче
- Получение текущего статуса решения выбранной задачи
- Получение подробной информации выбранной посылки к задаче

Лидерборд

- Получение статистики по командам в выбранном контесте



http

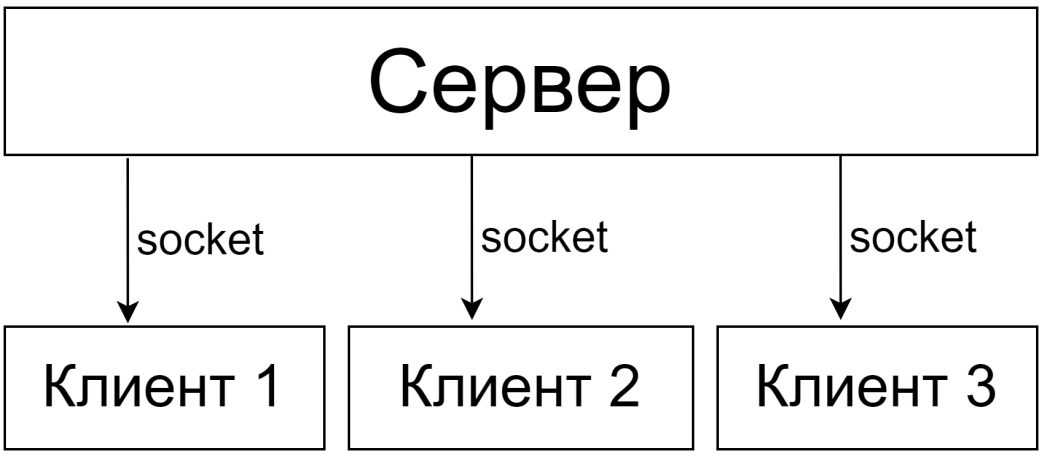
Получение информации по сокету всеми клиентами с сервера без предварительной отправки одним из клиентов.

Лобби + Тренировка + Лидерборд

- Получение id пользователя, вышедшего из online

Тренировка + Лидерборд

- Получение информации о завершении контеста



2.5. Технологии

Технология	**Аргументы**
React без Next.js	Отсутствие необходимости индексации
	Отсутствие необходимости переноса нагрузки на сервер
TS	Типизации данных
Axios	Запросы динамических данных без кэширования
	Подстановка токена в заголовок запроса в интерсепторе
RTKQuery	Запросы нединамических данных с кэшированием
Браузерный WebSocket	Актуализация данных, общих для всех клиентов
CSS модули	Снижение риска пересечения названия классов
	Удобство работы

3. Инфраструктура

3.1. Сборка и линтеры

1) Инструменты:

В проекте используется следующий набор инструментов:

Сборка:

В проекте используется Create React App, что автоматически конфигурирует Webpack и другие инструменты для сборки и разработки приложения.

Линтеры:

ESLint (версия 8.3.0) с конфигурацией "react-app/jest", и Stylelint (версия 15.10.2) для стилевых файлов.

Форматирование кода:

Prettier (версия 2.8.8) с настройками, включая сортировку импортов.

Транспиляция:

Babel с пресетом "react-app".

2) Способы запуска:

Вотчинг:

Для наблюдения за изменениями файлов и автоматической перезагрузки используется webpack-dev-server.

Прекоммит:

Отсутствует

Ручной запуск:

В scripts нашего package.json присутствуют команды для ручного запуска сборки, тестирования и работы с линтерами (например, **"style": "prettier --write ./src && stylelint --fix ./src/*_/_.css"**).

3) Конфигурация:

ESLint:

Настроен на расширение "react-app" и "react-app/jest".

Stylelint:

Используется порядок свойств и селекторов из stylelint-semantic-groups, настроены правила для сортировки.

Prettier:

Настройки включают отключение точек с запятой, настройку длины строки, типа кавычек и другие детали, которые обеспечивают единообразный стиль кода.

TypeScript:

Файл конфигурации включает строгий режим, настройки для работы с модулями CSS, а также алиасы для удобной работы с импортами.

Эти инструменты и конфигурации обеспечивают согласованность кода и облегчают разработку, позволяя легко масштабировать проект и поддерживать его в долгосрочной перспективе.

3.2. CI и деплой

1) Решение:

У нас отсутствует CI и мы внедрили возможность ручного деплоя из любой ветки,так как надо было разводить разные версии приложения: для бета-тестирования и для дальнейшей разработки.

2) Ручной деплой для разных версий приложения:

Команда **"deploy": "npm run build && rsync -av --delete--progress ./build/ student@158.160.111.208:/usr/share/nginx/html --rsync-path=\"sudo rsync\""**

выполняет следующие шаги:

Сборка проекта:

Сначала происходит сборка проекта с помощью команды npm run build. Это генерирует оптимизированные и минимизированные файлы проекта, обычно помещаемые в директорию **./build/**.

Синхронизация с удаленным сервером:

После сборки происходит синхронизация локальной директории **./build/** с директорией **/usr/share/nginx/html** на удаленном сервере по адресу **158.160.111.208**. Это делается с помощью утилиты **rsync**, с опциями:

-av для архивации и вывода подробного прогресса.

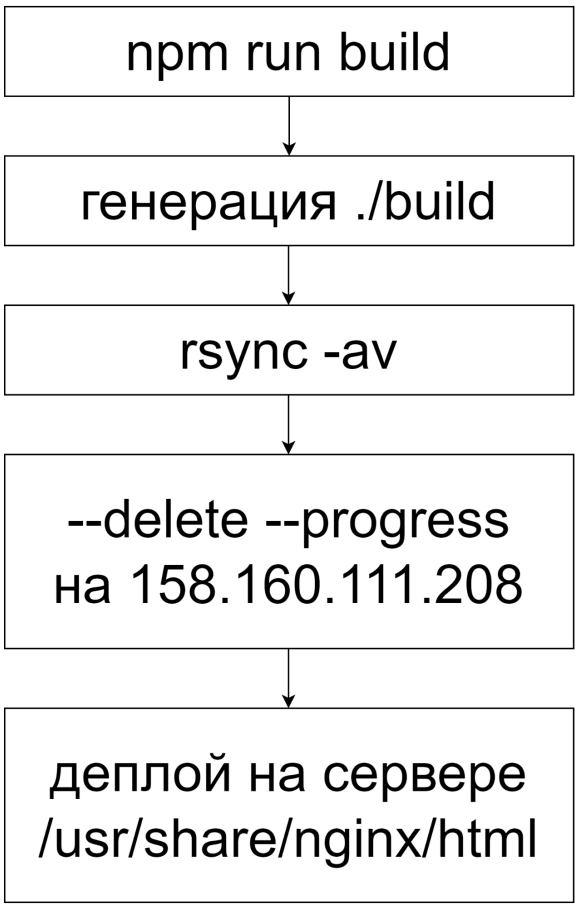
--delete для удаления файлов на сервере, которых нет в локальной директории.

--progress для вывода информации о ходе процесса.

--rsync-path=\"sudo rsync\" для выполнения rsync с правами суперпользователя на удаленной машине.

3) Визуализация процесса:

Этап	Описание
npm run build	Сборка проекта
Генерация ./build/	Создание оптимизированных и минимизированных файлов
rsync -av	Синхронизация с удаленным сервером
--delete --progress на 158.160.111.208	Удаление лишних файлов и отображение прогресса на указанном сервере
Деплой на сервере в /usr/share/nginx/html	Завершающий этап, где файлы размещаются на сервере



3.3. Тестирование (ручное и автоматическое)

1) Автоматическое тестирование:

Отсутствует

2) Ручное тестирование:

Функциональное тестирование:

Проверка различных функций приложения, таких как регистрация, вход, выбор тренировок и задач, выполнение и отправка решений, получение вердиктов и отслеживание прогресса в лидерборде.

Тестирование направлено на обеспечение соответствия всем требованиям и ожиданиям пользователей.

Интеграционное тестирование:

Тестирование взаимодействия различных частей приложения, таких как взаимодействие между клиентской и серверной частями(Websocket и HTTP запросы).

Интеграция с внешними сервисами (Яндекс Айди).

Приемочное тестирование:

Проведение всесторонней проверки перед выпуском новой версии приложения, чтобы убедиться, что оно готово к использованию конечными пользователями.

Это включало в себя тестирование различных сценариев тренировки, работы с задачами и управление процессом.

Тестирование совместимости:

Проверка приложения на различных браузерах.

Регрессионное тестирование:

Регулярная проверка основных функций приложения после внесения изменений или обновлений, чтобы убедиться, что существующая функциональность остается неповрежденной.

Это включало в себя проверку решения задач, получения вердиктов, отображение в лидерборде и других ключевых аспектов приложения.

3.4. Документация

Самодокументирующийся код:

Имена переменных, функций и классов выбирались таким образом, чтобы они явно и понятно описывали свою роль и функцию, уменьшая необходимость в дополнительной документации.

Регулярные встречи и обсуждения:

Вместо письменных материалов, команда проводила регулярные встречи для обсуждения и объяснения кода и архитектуры проекта.

4. Командная работа

4.1. Итерации

Ввиду небольшой команды было принято решение организовать работу по методологии Scrum. Рабочий процесс был разбит на недельные итерации, состоящие из трех мини-спринтов:

1. Вторник - пятница.
2. Суббота.
3. Воскресенье.

4.2. Планирование

В начале каждого спринта проводилось планирование, на котором формулировались и приоритезировались задачи на будущий стринт. Для удобного трекинга был использован такой инструмент, как Jira.

4.3. Распределение задач

При наличии параллельных задач, они распределялись на всех членом команды, если же стояла задача, требующая сложного решения одной проблемы, практиковалось совместное программирование.

4.4. Флоу

На каждую задачу выделялась отдельная ветка от основной ветки. Код-ревью проводилось редко ввиду нехватки времени и приоритета на скорость разработки. По возможности проводился рефакторинг.

Пример MR

4.5. Роли

Разработчик	Роли
Ярослав	Реализация WebSocket-менеджера (функционал подписки\отписки компонентов на события сокета)
	Обсуждение формата взаимодействия с бэкендом
	Рефактор кода для обеспечения расширяемости
Руслан	Деплой
	Подключение компонентов к логике взаимодействия с сервером
	Презентация проекта на демо
Илья	Комплексная реализация новых компонентов (вертска + подключение к api, socket)
	Реализация api-менеджера (функционал получения токена OAuth и подстановки в заголовки запросов)
Даяна	Верстка сложных компонентов
	Взаимодействие с дизайнером

4.6. Взаимодействие с менеджером и дизайнером

У дизайнера запрашивались варианты той или иной части интерфейса, из которых в рамках обсуждения с командой и менеджером выбирался рабочий.

Основное взаимодействие с менеджером осуществлялось в рамках общего планирования и синхронизации текущего состояния выполнения задач.

5. Чему научились

5.1. Личные достижения разработчиков

Илья

- Ознакомился с технологией авторизации через Яндекс.Паспорт впервые и успешно разобрался в ней. Теперь клиенты могут входить в систему, используя свои учетные записи Яндекса.
- Изучил принципы работы вебсокетов и успешно применил их на проекте. Это позволило улучшить взаимодействие клиент-сервер и обеспечивает главный функционал нашего приложения.
- Научился нести ответственность за свою часть работы. Работая в команде, приобрел навыки коллективного взаимодействия.
- Прошел через процесс рефакторинга кода, что улучшило его читаемость и эффективность. Научился находить и исправлять ошибки в коде.
- Освоил работу с pull request'ами и ветками в git, что сделало процесс разработки более организованным.

Руслан

- В ходе работы над проектом Я значительно расширил свои профессиональные навыки и компетенции. В частности:
- Публичные выступления: Усовершенствовал навык публичных выступлений, активно участвуя в презентациях и демонстрациях проекта.
- Командная разработка: Улучшил навык командной разработки, успешно сотрудничая с коллегами и достигая совместных целей.
- Чистота кода: Сосредоточил внимание на качестве и чистоте кода, применяя лучшие практики и стандарты.

- Дебаггинг: Улучшил навык дебаггинга, успешно находя и устраняя ошибки в коде.
- Работа в команде: Прокачал навык работы в команде, налаживая эффективное взаимодействие с коллегами.
- Взаимодействие с бекендерами: Укрепил сотрудничество с бэкендерами, обеспечивая согласованность и качество взаимодействия между клиентской и серверной частью.
- Архитектура фронтенд-приложений: Углубил знания в области архитектуры фронтенд-приложений, применяя современные подходы и технологии.
- Авторизация: Научился реализовывать авторизацию, обеспечивая безопасность и доступ к функциональности приложения.
- Работа с реальными API: Улучшил навык работы с реальными API, интегрируя внешние сервисы и обеспечивая надежную связь с сервером.
- Изучение работы с WebSockets: Освоил работу с WebSockets, обеспечивая реализацию реального времени в приложении.
- Освоение деплоя приложения: Научился настраивать ручной деплой для разных версий приложения
- Эти достижения не только повысили мой уровень экспертизы в различных областях разработки, но и сделали меня более гибким и адаптивным участником команды.

Ярослав

- Получил опыт выстраивания логики взаимодействия клиентской части приложения с серверной.
- Впервые поучаствовал в командной разработке. Ознакомился с процессами общего планирования, приоритезации, распределения задач.
- Научился оценивать время выполнения собственных задач и время выполнения задач участников своей команды для корректной синхронизации работы.

- Усилил навык оценки необходимости рефактора кода для обеспечения дальнейшей расширяемости проекта.

Даяна

- Улучшила стрессоустойчивость, научилась одновременно выполнять задачи по работе и проекту, параллельно выходя замуж.
- Приобрела опыт работы с React, разрабатывала UI компоненты, использовала современные веб-технологии.
- Попрактиковала умение разрешать merge конфликты, что повысило мой профессионализм в этой области.
- Усовершенствовала навыки оценки времени и планирования, это помогла сдавать задачи по проекту в срок.
- Развила навыки межличностного общения, наладила эффективную коммуникацию с коллегами по команде.

5.2. Топ три сложных задач

Авторизация

Процесс авторизации был сложным, особенно при использовании яндекс паспорта. Вначале не было ясности, как это реализовать на клиентской стороне. Решение пришло с идеей хранения токена аутентификации в куки и последующей передачи его на бэкенд для получения данных о пользователе.

Вебсокеты

Работа с вебсокетами изначально вызывала затруднения из-за нехватки знаний о технологии. Однако с течением времени было решенно создать класс для сокета и написать методы для подписки на конкретные события сокета. Это позволило интегрировать вебсокеты в проект и реализовать задуманный функционал.

Таймер

Сложность в реализации таймера заключалась в необходимости показывать время до конца тренировки без постоянных запросов к бэкенду и синхронизации времени для всех участников. В итоге мы договорились с командой бэкенда что таймер теперь работает на их стороне, а фронтенд делает запрос текущего времени на таймере один раз и запускает его на своей стороне. Это решение обеспечило правильную работу и синхронизацию таймера для всех пользователей.

5.3. Что планируется сделать позже

В планах команды ряд улучшений и нововведений направленных на повышение удобства и функциональности платформы. Среди них:

- Добавление адаптивности для окон, аналогично leetcode, что обеспечит более гибкую настройку интерфейса.

- Внедрение возможности изменения параметров таймера, что даст команде больше контроля над временем тренировки.

- Функция паузы тренировки.

- Реализация отправки решений через файл, добавление горячих клавиш для удобной навигации, в том числе переключения между табами и лидербордом.

- Возможность создания нескольких табов редактора кода для оптимизации решений.

- Возможность написания своих тест-кейсов прямо рядом с редактором кода.

Эти улучшения сделают платформу еще более удобной и эффективной для участников.

6. Результаты работы

6.1. Ссылки на видео

6.2. Ссылка на сервис

логин: contest

пароль: shbr2023

6.3. Ссылка на репозиторий

6.4. Ссылка на Merge Requests