**МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ**

**федеральное государственное бюджетное образовательное учреждение**

**высшего образования**

**«УЛЬЯНОВСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ» ФАКУЛЬТЕТ СРЕДНЕГО ПРОФЕССИОНАЛЬНОГО ОБРАЗОВАНИЯ –**

**КОЛЛЕДЖ ЭКОНОМИКИ И ИНФОРМАТИКИ ИМ А.Н. АФАНАСЬЕВА**

Специальность 09.02.03 Программирование в компьютерных системах

К защите допустить «\_\_\_»\_\_\_\_\_\_\_\_\_\_ 2023 г.

Декан ФСПО С.Ю. Прохорова

(подпись)

**ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА**

**(дипломный проект)**

**Тема** Мобильное приложение для фитнес-клуба

**Студент:** / Д.С. Львова /

подпись инициалы, фамилия

**Руководитель:** / О.Б. Ларионова /

подпись инициалы, фамилия

**Нормоконтролёр:** / Ю.В. Рассадина /

подпись инициалы, фамилия

Ульяновск, 2023**МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ**

**федеральное государственное бюджетное образовательное учреждение**

**высшего образования**

**«УЛЬЯНОВСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»**

**ФАКУЛЬТЕТ СРЕДНЕГО ПРОФЕССИОНАЛЬНОГО ОБРАЗОВАНИЯ –**

**КОЛЛЕДЖ ЭКОНОМИКИ И ИНФОРМАТИКИ ИМ А.Н. АФАНАСЬЕВА**

Специальность 09.02.03 Программирование в компьютерных системах

|  |  |
| --- | --- |
| Рассмотрено и одобрено на заседании  предметной (цикловой) комиссии  Вычислительная техника и программирование  Протокол № 6 от « 01 » февраля 2023 года  Председатель П(Ц)К \_\_\_\_\_\_\_\_\_\_ Е.В. Графова  (подпись) (инициалы, фамилия) | УТВЕРЖДАЮ:  Декан \_\_\_\_\_\_\_\_\_\_\_ С.Ю. Прохорова  (подпись) (инициалы, фамилия)  «\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 2023 г. |

**ЗАДАНИЕ**

**на выпускную квалификационную работу студента**

группы: Пдо-41

Львова Диана Сергеевна

(Фамилия, имя, отчество)

тип ВКР дипломный проект

1. Тема ВКР Мобильное приложение для фитнес-клуба

утверждена приказом по университету № 554 от 15.03.2023

2. Срок сдачи студентом дипломного проекта 16.06.2023

3. Исходные данные к работе

Задание, методические указания, ГОСТ, учебная и справочная литература

4. Содержание дипломного проекта:

4.1 Введение (определение целей, постановка задачи)

4.2 Аналитическая часть. Описание предметной области. Сравнительный анализ аналогов

4.3. Теоретическая часть. Обзор инструментальных средств создания продукта. Обоснование выбора средств реализации.

4.4. Практическая часть – реализация программного продукта.\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

4.5 Экономическая часть\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

4.6. Заключение (выводы, значимость проделанных результатов)

4.7. Список использованных источников

4.8. Приложение (в том числе в электронном виде)

5. План-график выполнения дипломного проекта

|  |  |  |  |
| --- | --- | --- | --- |
| Наименование этапов ВКР | Консультант | Дата, подпись | |
| Задание выдал | Задание принял |
| 1. Обоснование темы и оформление задания на выпускную квалификационную работу (далее - ВКР), составление предварительного плана работы. | Ларионова О.Б. | 25.03.2023 | 07.04.2023 |
| 2. Подбор материалов для ВКР, изучение ис­точников. | Ларионова О.Б. | 07.04.2023 | 21.04.2023 |
| 3. Составление плана ВКР, анализ предмет­ной области, разработка проекта содержа­тельной части ВКР. Написание введения. | Ларионова О.Б. | 22.04.2023 | 28.04.2023 |
| 4. Реализация программных модулей | Ларионова О.Б. | 28.04.2023 | 21.05.2023 |
| 5. Экономический раздел | Рябоволова Т.А. | 23.05.2022 | 24.05.2022 |
| 6. Организационно-заключительная оценка степени практической значимости ВКР | Ларионова О.Б. | 25.05.2022 | 28.05.2022 |
| 7. Оформление списка используемых источ­ников | Ларионова О.Б. | 25.05.2022 | 28.05.2022 |
| 8. Оформление работы, нормоконтроль, согласование с консультантами по отдельным частям, получение отзыва руководителя. | Ларионова О.Б.  Рассадина Ю.В. | 25.05.2022 | 03.06.2022 |
| 9. Исправление замечаний по результатам предзащиты | Ларионова О.Б. | 03.06.2022 | 16.06.2022 |

Дата выдачи задания 25.03.2023

Руководитель ВКР / О.Б. Ларионова, замдекана УЛГТУ /

(подпись) (инициалы, фамилия, должность)

Задание принял к исполнению / Д.С. Львова /

(подпись) (инициалы, фамилия)

СОДЕРЖАНИЕ

[ВВЕДЕНИЕ 5](#_Toc137680471)

[1 Теоретическая часть 7](#_Toc137680472)

[1.1 Актуальность онлайн фитнес – приложений в России 7](#_Toc137680473)

[1.2 Сравнительный анализ аналогов 13](#_Toc137680474)

[1.3 Постановка задачи 16](#_Toc137680475)

[1.4 Средства разработки мобильного приложения 16](#_Toc137680476)

[1.4.1 HTML 5 16](#_Toc137680477)

[1.4.2 CSS3 17](#_Toc137680478)

[1.4.3 JavaScript 18](#_Toc137680479)

[1.4.4 PHP 19](#_Toc137680480)

[1.5 Фреймворки 20](#_Toc137680481)

[1.5.1 Vue.js 20](#_Toc137680482)

[1.5.2 React.js 21](#_Toc137680483)

[2 Проектирование и разработка программного продукта 24](#_Toc137680484)

[2.1 Диаграмма вариантов использования 24](#_Toc137680485)

[2.2 Диаграмма компонентов 25](#_Toc137680486)

[2.3 Диаграмма деятельности 26](#_Toc137680487)

[2.4 Диаграмма последовательности 28](#_Toc137680488)

[2.5 Разработка приложения 29](#_Toc137680489)

[3 Руководство пользователю 64](#_Toc137680490)

[4 Экономическая часть 76](#_Toc137680491)

[4.1 Расчет трудоемкости разработки программного продукта 76](#_Toc137680492)

[4.2 Расчет себестоимости и цены разработки программного продукта 77](#_Toc137680493)

[4.3 Методика расчета статей калькуляции 77](#_Toc137680494)

[4.3.1 Расчет материальных затрат 77](#_Toc137680495)

[4.3.2 Определение цены программного продукта 81](#_Toc137680496)

[ЗАКЛЮЧЕНИЕ 83](#_Toc137680497)

[СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 84](#_Toc137680498)

[ПРИЛОЖЕНИЕ А 86](#_Toc137680499)

[ПРИЛОЖЕНИЕ Б 88](#_Toc137680500)

[ПРИЛОЖЕНИЕ В 90](#_Toc137680501)

[ПРИЛОЖЕНИЕ Г 95](#_Toc137680502)

[ПРИЛОЖЕНИЕ Д 98](#_Toc137680503)

[ПРИЛОЖЕНИЕ Е 100](#_Toc137680504)

[ПРИЛОЖЕНИЕ Ж 101](#_Toc137680505)

# ВВЕДЕНИЕ

Современный образ жизни ставит перед нами все более высокие требования к поддержанию здоровья и физической формы. С ростом осознания важности активного образа жизни и регулярных физических нагрузок, растет и спрос на фитнес-клубы и тренировки. Однако, чтобы привлечь и удержать клиентов в современной конкурентной среде, фитнес-клубам необходимо предложить не только качественные тренировки, но и инновационные решения.

В рамках данного дипломного проекта представляется разработка мобильного приложения для фитнес-клуба, с целью улучшить взаимодействие между клубом и его клиентами, обеспечить удобство и комфорт при посещении клуба, а также повысить мотивацию клиентов для достижения своих фитнес-целей.

Мобильные приложения стали неотъемлемой частью нашей повседневной жизни и играют важную роль в обеспечении доступа к различным услугам и информации. Фитнес-клубы, осознавая это, стремятся внедрить собственные мобильные приложения для улучшения взаимодействия с клиентами. Такие приложения позволяют клиентам получать доступ к расписанию тренировок, следить за своими достижениями и прогрессом, получать персонализированные рекомендации и советы по тренировкам и питанию.

Целью данного дипломного проекта является разработка мобильного приложения для фитнес-клуба, которое будет удовлетворять потребности клиентов и создавать удобную платформу для взаимодействия. Анализ существующих решений на рынке и опросы среди потенциальных пользователей позволят выявить ключевые функциональные возможности и особенности, которые будут реализованы в разрабатываемом приложении.

Ожидается, что разработка мобильного приложения для фитнес-клуба приведет к повышению удовлетворенности клиентов, улучшению их результатов тренировок и увеличению активности посещений клуба. Кроме того, приложение будет способствовать укреплению связи между тренерами и клиентами, обеспечивая персонализированные рекомендации и консультации.

Для разработки мобильного приложения будет использован современный инструментарий и технологии, обеспечивающие стабильную работу приложения на различных устройствах и операционных системах. Будет уделено особое внимание дизайну пользовательского интерфейса, чтобы создать интуитивно понятный и привлекательный пользовательский опыт.

В рамках работы над дипломным проектом будут проведены исследования и анализ требований клиентов, определены основные функциональные возможности приложения, разработана архитектура системы и реализованы соответствующие модули и компоненты. Также будет осуществлено тестирование приложения для обеспечения его стабильной работы и соответствия требованиям.

В заключение, разработка мобильного приложения для фитнес-клуба представляет собой актуальную и перспективную задачу, направленную на улучшение качества обслуживания клиентов и повышение эффективности работы клуба. Ожидается, что создание удобного и функционального приложения позволит привлечь новых клиентов, удержать существующих и повысить их мотивацию для достижения своих фитнес-целей.

# 1 Теоретическая часть

## 1.1 Актуальность онлайн фитнес – приложений в России

Современный мир характеризуется повсеместной цифровизацией, технологическими достижениями, которые изменили жизнь всего мирового сообщества. Из-за стремительного развития информационных технологий, возникло такое явление, как гиподинамия или малоподвижный образ жизни у детей и взрослых.

За последние десятилетия уровень физических нагрузок неуклонно снижается, в первую очередь у детей и подростков, из- за чего поддержание их физической активности и здоровья становится непростой задачей. Снижение физической нагрузки неизбежно ведет к проблемам со здоровьем уже в молодом возрасте, к развитию хронических заболеваний, нарушениям работы организма в целом, и к низкому уровню физической подготовленности.

Важнейшей задачей современных образовательных учреждений (школа, институт) является формирование навыков здорового образа жизни, прививание любви к спорту и формирование здоровых привычек. При этом одних занятий в образовательных учреждениях недостаточно. Самостоятельные занятия ученика, студенты играют даже большую роль в физическом воспитании, чем занятия в школе или институте.

Согласно современным исследования ученых, максимальная активность мозга ребенка, подростка наблюдается при самостоятельной деятельности, в то время как во время обычного группового занятия уровень мозговой активности такой же, как и во время сна. В связи с чем, на взгляд автора применение онлайн фитнес приложений позволяет решить сразу три задачи:

повысить физическую активность ребенка, взрослого;

привить дисциплинированность, самостоятельность и ответственность в занятиях физической культурой и спортом, а также развивать творческий подход в своих занятиях.

Таким образом, актуальность исследования не оставляет сомнений**.**

Обзор онлайн фитнес приложений для занятий физической культурой и спортом и для ведения здорового образа жизни в России

В наше время количество фитнес приложений растёт изо дня в день. Так, в период с 2016 по 2018 рост скачиваний онлайн фитнес-приложений вырос на 22%. В 2015 году количество пользователей составляло 133,8 млн. человек, в 2019 году достигло 302,4 млн. человек. В 2020 году наблюдался значительный рост скачиваний фитнес-приложений, в том числе из-за пандемии коронавируса. Количество пользователей в 2020, в 2021 году превысило 400 млн. человек, а в 2022 году уже составило 650 млн.

Интересна статистика по онлайн приложениям:

– к концу 2023 года мировой доход от приложений для фитнеса достигнет 6 млрд. долларов, а к 2028 году – 2028 млрд. долларов;

– на июль 2022 года пришлось наибольшее количество скачиваний приложений – 656 млн. раз, при этом 150 млн. скачиваний приходились на приложения для здоровья;

– в пандемию 2019–2022 годов около 40% пользователей хотя бы раз скачали фитнес-приложение;

– из 84 миллионов активных пользователей в 2022 году лишь 22% – те, кто пользовался приложением до пандемии;

по итогам 2022 года число пользователей, открывающих приложение ежедневно увеличилось на 24%.

На рисунке 1 представлен обзор наиболее распространенных приложений для фитнеса в России.

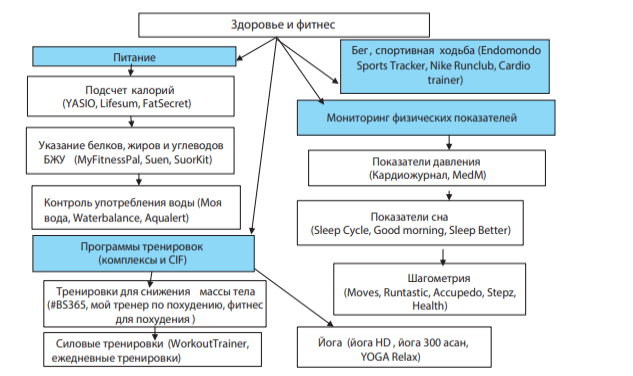


Рисунок 1 - Мобильные приложения по онлайн фитнесу и здоровому образу жизни в России

Наиболее высокие оценки пользователей получили такие приложения как Nike+, Training club, Workout trainer, Fitprosport, Jefit. Согласно представленной классификации, все онлайн приложения для фитнеса условно делятся на 4 категории: питание, программы тренировок, бег и ходьба, мониторинг физических показателей

Категория питание представлена счетчиками калорий, приложениями для подсчета жиров, белков и углеводов, а также приложениями по контролю воды. Приложения по питанию представлены блюдами, меню и анализаторами калорийности; есть счетчики отдельных блюд и продуктов.

Мониторинги физических показателей включают показатели давления, анализаторы сна, анализаторы количества шагов, приложения по ходьбе и бегу. Мобильные приложения с планами тренировок содержат: тренировки для снижения массы тела, силовые программы и йогу.

На рисунке 2 представлено процентное соотношение использования разных видов приложений российскими пользователями.



Рисунок 2 - Используемые мобильные приложения российскими пользователями, %

В результате проведенного исследования современных приложений для фитнеса, были выделены отдельные категории и виды приложений, была сформирована их классификация. Также были сгруппированы приложения по следующим категориям: мобильные приложения, носимые технологии с программным обеспечением системы тестирования с вводом и обработкой данных в режиме онлайн.

Мобильные приложения удобны в использовании. Они могут синхронизироваться с различными мобильными устройствами: часами, телефоном, пульсометром и т.д. Наиболее популярными приложениями по отслеживанию и анализу показателей физической активности являются: Polar Flow, RunKeeper, Strava, Adidas Running, Nike Run Club.

Polar Flow – современное онлайнприложение с большим набором функций. Приложение предназначено для планирования и проведения тренировок, для контроля активности и сна. Приложение Polar Flow способно синхронизироваться с часами Polar Vantage для дальнейшего отслеживания таких показателей как: калории, продолжительность тренировок, частота сердечных сокращений, темп, скорость, дистанция, мощность, максимальное потребление кислорода, максимальная аэробная нагрузка и т.д. Приложение позволяет устанавливать индивидуальные настройки для разных видов спорта.

Онлайн приложения, которые предназначены для отслеживания физической активности через системы GPS и ГЛОНАСС называются фитнес-трекерами. Приложение RunKeeper позволяет через спутниковую связь предоставлять такие функции, как отслеживание расстояния, времени движения, пульса, скорости и т.д. Пользователи могут пользоваться пульсометром, создавать многодневные планы тренировок и использовать другие функции.

Следующее довольно популярное онлайн приложение – Strava. Srava – фитнестрекер, синхронизирующийся с большинством устройств. Это приложение позволяет строить собственные маршруты или пользоваться маршрутами друзей на телефоне или навигаторе. Приложение позволяет использовать большое количество показателей стандартных и нестандартных: расстояние, скорость, время, пульс, мощность общую и средневзвешенную и др.

Adidas Running также фитнес-трекер, который позволят отслеживать большое количество различных показателей: маршрут, калории, время, расстояние, скорость и т.д. В приложении есть трекер для бега с построением планов тренировок.

Nike Run Club – фитнес-трекер для бега с большинством необходимых функций (темп, пульс, время калории и т.д.) Приложение отображает отчеты по каждой тренировке, также есть возможность сравнения результатов с предыдущими тренировками.

Актуальность использования онлайн-приложений для фитнеса в российских ВУЗах

Быстрое развитие цифровых технологий на российском рынке, стремительное внедрение проектов цифровизации в российское образование, обусловили необходимость использования информационных технологий в процессе образования российских студентов. Цифровые технологии используются для автоматизации рабочего процесса, позволяют создавать принципиально новые модели профессиональной деятельности, решать задачи нового высшего уровня.

Основные возможности информационных технологий заключаются в возможностях применения технологий мультимедиа, интерактивности, интеллектуальности, производительности и коммуникативности. Цифровизация на сегодня – существенный фактор, который позволяет существенно преобразовывать процесс обучения в целом.

Необходимость применения цифровых технологий сегодня сложно отрицать. Все российское общество стремится к цифровизации различных сфер деятельности. Следует отметить, что в первую очередь, проекты цифровизации поддерживаются и активно развиваются со стороны российской государственной власти. Так, для внедрения цифровых технологий в различные сферы жизнедеятельности общества, были разработаны федеральные проекты «Цифровая экономика», «Современная цифровая образовательная среда в РФ», «Устранение цифрового неравенства», информационная система «Цифровая школа». Все эти проекты направлены на внедрение информационных технологий, в первую очередь, в экономику и образование. В рамках этих проектов образовательные процессы строятся на базе информационных технологий.

В наше время на базе ВУЗов были созданы такие платформы, как «Universarium», «Stepik». На данных платформах представлены образовательные программы, которые в том числе содержат курсы по физической культуре. Сервис «Udemy» предоставляет возможность использовать целую категорию курсов по направлению здоровье и фитнес. В них представлены курсы по обучению медитации, питанию; различные тренировки и др. Также используются сервисы «lektorium», «eliademy», «eduardo.studio». Сегодня перед российскими ВУЗами стоит задачи построения 3500-4000 онлайн-курсов, в том числе и по физической культуре и спорту. Образовательные организации только начинают работу в этом направлении, однако, уже имеющиеся сервисы набирают большое количество пользователей.

Итак, онлайн приложения для фитнеса, в России и в мире сегодня обладают большим потенциалом. Они способны формировать:

– нацеленность обучающихся на получение знаний;

– формирование навыков использования электронных устройств;

– снижение уровня тревожности, застенчивости и нерешительности;

– развитие стойкого интереса, положительной мотивации и способностей к занятиям физической культурой и спортом;

– развитие целеустремленности и желания стать успешным;

– развитие коммуникативных навыков, навыков сотрудничества;

– самореализация, самосовершенствование;

– ориентация на общечеловеческие ценности, ценности здорового образа жизни; формирование здоровых привычек.

Использование онлайн приложений для фитнеса сегодня актуально как никогда. Разумно подобранные тренировки, доступность в любое время, низкие финансовые расходы на использование приложений в совокупности с трендами, ориентированными на здоровый образ жизни, позволяет говорить о необходимости развития этой сферы применения информационных технологий.

Любой школьник, студент, взрослый может использовать мобильные приложения, фитнес-трекеры для повышения общей физической подготовленности и контроля здоровья. Полезными являются и дополнительные приложения по питанию, анализу питания, калорийности и т.д., в совокупности все эти приложения упрощают внедрение здорового образа жизни для учащихся и взрослого населения России.

## 1.2 Сравнительный анализ аналогов

Проведем анализ сопоставимых приложений, выделив некоторые критерии для оценки. Удобство пользовательского интерфейса является одним из главных факторов, определяющих комфортность работы с системой и уровень взаимодействия пользователя с необходимыми функциями. Очевидно, что этот критерий также влияет на скорость работы приложения.

При выборе подходящей платформы для разработки мобильного приложения, одной из наиболее популярных является операционная система Android, что предоставляет пользователям широкий доступ к различным приложениям.

В настоящее время программы лояльности часто являются неотъемлемой частью бизнеса, направленной на управление клиентским опытом и удержание клиентской базы. При анализе окружающих брендов можно обнаружить множество успешных и малоэффективных программ.

Существующие аналоги предлагают бонусные системы, которые функционируют следующим образом: клиент выполняет определенные целевые действия и получает за них баллы, которые затем можно использовать по-разному.

Включение расписания в мобильное приложение является одним из наиболее удобных нововведений для фитнес-клубов. Клиенты получают доступ к информации о графике работы клуба и тренеров, которая регулярно обновляется.

Для оперативного получения уведомлений об изменениях в расписании, отменах занятий, акциях и других событиях клуба в приложении предусмотрена удобная функция системы уведомлений, которая позволяет клиентам быть в курсе всех событий.

Интеграция приложения с социальными сетями, то есть его синхронизация со страницами в популярных социальных сетях, позволяет клиентам получить больше актуальной и достоверной информации о фитнес-клубе.

Исходя из этих критериев была составлена таблица для оценки сравниваемых аналогов данного проекта.

Таблица 1 – Оценка мобильных приложений фитнес – клуба

|  |  |  |  |
| --- | --- | --- | --- |
| Критерии оценивания/Приложения | Hulk.fit | Bright fit | Прайм – фитнес |
| Удобство пользовательского интерфейса | Легко | Легко | Легко |
| Платформа | Android/IOS | Android/IOS | Android/IOS |
| Наличие сайта | Есть | Есть | Есть |
| Система уведомлений | Есть | Есть | Есть |
| Программа лояльности | Нет | Нет | Есть |
| Наличие расписания | Есть | Есть | Есть |
| Интеграция с соц.сетями | Доступно | Доступно | Доступно |

При разработке данного мобильного приложения были определены следующие критерии:

Удобство пользовательского интерфейса.

Выбор подходящей платформы.

Наличие веб-сайта.

Функционал системы уведомлений.

Программа лояльности.

Реализация расписания.

Интеграция с социальными сетями.

## Постановка задачи

В выпускной квалификационной работе требуется разработать мобильное приложение для фитнес - клуба, основная функция, которого заключается в отслеживание физических данных, получении рекомендаций по тренировкам и питанию, доступ к расписанию занятий. Данная функция поможет достичь желаемых результатов, следить за своим здоровьем и сохранить отличную форму.

К проекту выставлены следующие требования:

1. Приложение должно иметь удобный интерфейс и приятный дизайн;
2. Оптимизацию под мобильное устройство;
3. Возможность просмотра расписания;
4. Возможность введения параметров для отслеживания прогресса;
5. Возможность просмотра физической активности;

## 1.4 Средства разработки мобильного приложения

### 1.4.1 HTML 5

HTML5 представляет собой мощный инструмент для структурирования веб-контента. Он упрощает проектирование и разработку веб-сайтов, предоставляя стандартизированный и интуитивно понятный пользовательский интерфейс. С помощью HTML5 разработчикам доступны средства для организации и сегментации веб-страниц, а также создания независимых компонентов, которые обеспечивают логическую структуру сайта и возможности синдикации. Основываясь на принципах отображения информации, HTML5 позволяет разделить и пометить информацию для удобства использования и понимания. Благодаря этому, HTML5 обладает значительной семантической и эстетической ценностью.

HTML5 предоставляет дизайнерам и разработчикам различных уровней возможности представления разнообразного контента - от простых текстов до интерактивных материалов с мультимедийным наполнением. Он упрощает разработку кросс-браузерных веб-приложений и мобильных приложений, а также способствует развитию мобильных сервисов на основе облачных вычислений. Гибкость HTML5 проявляется в возможности создания впечатляющих и интерактивных веб-сайтов. Также HTML5 предлагает новые теги и улучшения, такие как элегантная структура, элементы управления формами, API-интерфейсы, мультимедийные функции, поддержка баз данных и повышенная скорость обработки.

Одной из ключевых особенностей HTML5 является использование названий тегов, которые явно описывают их назначение и использование. В предыдущих версиях HTML использовались менее конкретные названия тегов. В HTML5 используются описательные и интуитивно понятные названия, которые точно определяют предназначение соответствующего контента. Новые теги, такие как <section>, <article>, <video>, <audio>, <canvas> и <figure>, обеспечивают более точное описание различных типов контента.

HTML5 предоставляет разнообразные возможности, включая описательные названия тегов, улучшенные сетевые коммуникации, хранение данных, фоновые процессы, постоянное соединение с сервером, улучшенное извлечение данных, ускоренную загрузку страниц, поддержку CSS3, улучшенную обработку форм, API-интерфейс баз данных, возможность добавления графики и видео без сторонних модулей, геолокационные возможности и более эффективную связь между мобильными устройствами и облачными серверами.

### 1.4.2 CSS3

CSS3 представляет собой новый стандарт для стилизации HTML документов, который значительно расширяет возможности предыдущего стандарта CSS2.1.

В CSS3 появилась возможность легко достигать многих функций, которые ранее были сложнодоступны в CSS2.1 и требовали использования внешних программ, скриптов или специальных приемов. Это достигается за счет внедрения новых свойств оформления.

В CSS3 можно:

Создавать элементы с гладкими углами.

Создавать линейные и радиальные градиенты.

Гибко настраивать фоновые изображения элементов.

Добавлять тени к элементам и тексту.

Использовать нестандартные шрифты без опасений, что они не будут поддерживаться браузером пользователя.

Создавать анимацию и различные эффекты переходов.

Определять цвета с использованием новых способов и многое другое.

### 1.4.3 JavaScript

JavaScript был разработан в 1995 году в компании Netscape как язык сценариев для браузера Netscape Navigator 2. Изначально он назывался LiveScript, но в связи с популярностью языка Java в то время, его переименовали в JavaScript. Однако, это часто приводит к путанице, и некоторые новички ошибочно считают, что Java и JavaScript - это одно и то же. На самом деле, это два разных языка, связанных только названием.

Первоначально JavaScript имел ограниченные возможности и использовался преимущественно для добавления некоторого поведения на веб-страницы. Например, обработки событий, связанных с кнопками и другими элементами управления.

Однако, с развитием веб-технологий, появлением HTML5 и технологии Node.js, JavaScript получил гораздо больше возможностей. Сегодня JavaScript продолжает использоваться для создания веб-сайтов, но предоставляет гораздо более широкий спектр функциональности.

Он также используется в качестве серверного языка программирования. Ранее для серверной разработки приходилось использовать другие языки, такие как PHP, ASP.NET, Ruby или Java, в то время как JavaScript применялся только на стороне клиента. Однако благодаря Node.js сейчас JavaScript может обрабатывать запросы на сервере.

В последнее время JavaScript также стал популярным для мобильной разработки. Увеличение вычислительной мощности мобильных устройств и широкое распространение стандарта HTML5 позволяют использовать JavaScript для создания приложений для смартфонов и планшетов.

JavaScript также перешагнул границы веб-браузера, для которого он изначально был создан. С появлением новых операционных систем Windows 8 / 8.1 / 10, JavaScript стал использоваться для разработки приложений для этих платформ.

Кроме того, JavaScript может быть использован для программирования устройств Интернета вещей (IoT). Это открывает новые возможности для создания "умных" устройств, взаимодействующих с Интернетом.

Таким образом, JavaScript находит применение практически везде. Он является одним из самых популярных языков программирования сегодня, и его популярность продолжает расти.

Отметим также, что существует стандарт языка JavaScript, разрабатываемый организацией ECMA. Наиболее распространенным стандартом на данный момент является ECMAScript 2015 (ES6), однако его полная поддержка в популярных браузерах еще не завершена. В данном руководстве рассматриваются возможности стандарта ES5, которые уже доступны во всех основных браузерах.

JavaScript является интерпретируемым языком, что означает, что его код выполняется с помощью интерпретатора. Интерпретатор получает инструкции на языке JavaScript, определенные на веб-странице, и выполняет их.

### 1.4.4 PHP

PHP - это язык программирования, специально разработанный для создания web-приложений, которые выполняются на сервере. Синтаксис PHP во многом основан на синтаксисе C, Java и Perl, что делает его знакомым для опытных программистов. В то же время он доступен и для веб-мастеров, которые не имеют опыта в других языках программирования.

Одним из основных преимуществ PHP по сравнению, например, с JavaScript, является то, что PHP-скрипты выполняются на стороне сервера. Это означает, что скорость компьютера пользователя или его браузера не влияют на выполнение PHP-скриптов. Пользователь может получать результат выполнения скрипта, не осознавая, что получает необычный HTML-файл. Скрипты на PHP могут быть выполнены на сервере как отдельные файлы или интегрированы в HTML-страницы. PHP также может генерировать и преобразовывать различные форматы данных, включая изображения (JPEG, GIF, PNG), PDF-файлы и FLASH. Он способен работать с данными в любом текстовом формате, включая XHTML и XML.

PHP является кроссплатформенной технологией, поддерживаемой на большинстве операционных систем, таких как Linux, Unix, Microsoft Windows, Mac OS и других. Он также поддерживается на большинстве веб-серверов, включая Apache, Microsoft Internet Information Server (IIS) и Microsoft Personal Web Server. PHP может работать как модуль или как CGI препроцессор на большинстве серверов. Он поддерживает работу с различными базами данных, такими как MySQL, MSQL, Oracle, PostgreSQL, SQLite и другими.

Язык программирования PHP, особенно в сочетании с базой данных MySQL, является оптимальным выбором для создания веб-сайтов различной сложности.

PHP постоянно развивается, и ему обеспечено долгое господство в области web-программирования.

## 1.5 Фреймворки

### 1.5.1 Vue.js

Vue.js - это фреймворк JavaScript, разработанный в 2013 году, который идеально подходит для создания гибких пользовательских интерфейсов и сложных одностраничных приложений.

Преимущества Vue.js:

Улучшенный HTML: Vue.js обладает множеством функций, похожих на Angular, что помогает оптимизировать HTML-блоки с использованием компонентов.

Подробная документация: Vue.js предлагает подробную документацию, которая упрощает процесс обучения разработчиков и сокращает время разработки приложений, используя базовые знания HTML и JavaScript.

Гибкость: Благодаря своей архитектуре и дизайну, Vue.js обеспечивает быстрый переход от других фреймворков, таких как Angular и React.

Превосходная интеграция: Vue.js может использоваться для создания как одностраничных приложений, так и сложных веб-интерфейсов. Он легко интегрируется с существующей инфраструктурой без негативного влияния.

Масштабируемость: Vue.js облегчает разработку крупных многоразовых шаблонов, что позволяет сэкономить время по сравнению с более простыми подходами.

Малый размер: С размером около 20 КБ Vue.js остается быстрым и гибким, обеспечивая лучшую производительность по сравнению с другими платформами.

Недостатки Vue.js:

Ограниченное количество ресурсов: В настоящее время Vue.js занимает меньшую долю рынка по сравнению с React или Angular, что означает, что обмен знаниями в сообществе разработчиков находится на начальной стадии.

Риск чрезмерной гибкости: Иногда Vue.js может столкнуться с проблемами при интеграции в крупные проекты, и пока еще не существует широкого опыта и решений, но с течением времени они будут разработаны.

Некоторые компании, использующие Vue.js, включают Xiaomi, Alibaba, WizzAir, EuroNews, Grammarly, GitLab и Laracasts, а также Adobe, Behance, Codeship и Reuters.

### 1.5.2 React.js

React - это библиотека JavaScript, разработанная Facebook в 2013 году, которая идеально подходит для создания современных одностраничных приложений любого размера и масштаба.

Преимущества React:

Легкость изучения: Благодаря простому дизайну, использованию JSX (синтаксис, похожий на HTML) для шаблонов и подробной документации, React легко осваивается разработчиками. Они могут сосредоточиться на написании современного JavaScript, не беспокоясь о специфичных для фреймворка аспектах.

Высокая производительность: Благодаря использованию виртуального DOM и различным оптимизациям рендеринга, React обеспечивает быструю работу приложений.

Поддержка серверного рендеринга: React предоставляет отличную поддержку серверного рендеринга, что делает его мощной платформой для контент-ориентированных приложений.

Поддержка Progressive Web App (PWA): С помощью инструмента create-react-app React легко поддерживает разработку прогрессивных веб-приложений.

Односторонняя привязка данных: React использует одностороннюю привязку данных, что помогает избежать нежелательных побочных эффектов.

Поддержка Redux: Redux - популярная платформа для управления состоянием приложений в React, которая легко осваивается и используется.

Функциональное программирование: React внедряет концепции функционального программирования (FP), что делает код простым в тестировании и повторном использовании.

Поддержка TypeScript и Flow: React позволяет создавать приложения с использованием TypeScript или Flow и обладает встроенной поддержкой JSX.

Простой переход между версиями: Facebook предоставляет "кодовые модули", которые упрощают процесс перехода между версиями React.

Возможность применения навыков React к React Native: Навыки, приобретенные в React, могут быть использованы при разработке на React Native.

Недостатки React:

Неоднозначность: React предоставляет разработчикам возможность выбирать наилучший способ развития, что требует хорошего лидерства проекта и установленных процессов.

Разделение в сообществе: Существует разделение в сообществе относительно подходов к написанию CSS в React, включая использование традиционных таблиц стилей (CSS Modules) или CSS-in-JS (таких как Emotion и Styled Components).

Отход от классовых компонентов: React движется в сторону функциональных компонентов, что может вызывать проблемы у разработчиков, привыкших к объектно-ориентированному программированию (ООП).

Смешивание шаблонов с логикой: Некоторые разработчики могут быть сбиты с толку из-за смешивания шаблонов (JSX) с логикой при первом знакомстве с React.

Некоторые компании, использующие React, включают Facebook, Instagram, Netflix, New York Times, Yahoo, Khan Academy, WhatsApp, Codecademy, Dropbox, Airbnb, Asana, Atlassian, Intercom, Microsoft, Slack, Storybook и многие другие.

# 2 Проектирование и разработка программного продукта

## 2.1 Диаграмма вариантов использования

Диаграмма вариантов использования отражает отношения между актерами и прецедентами системы и позволяет описать систему на концептуальном уровне.

Прецедент – возможность моделируемой системы, часть ее функциональности, благодаря которой актер может получить конкретный, измеримый и нужный ему результат. Прецедент соответствует отдельному сервису системы, определяет один из вариантов ее использования и описывает типичный способ взаимодействия пользователя с системой. Диаграмма прецедентов обычно применяется для спецификации внешних требований к системе.

В ходе проектирования был выделен следующий актер.

Пользователь – пользователь приложения, которому доступна возможность использования всех его функций.

На основе функциональных требований к мобильному приложению была создана диаграмма прецедентов, которая представлена на рисунке 3.

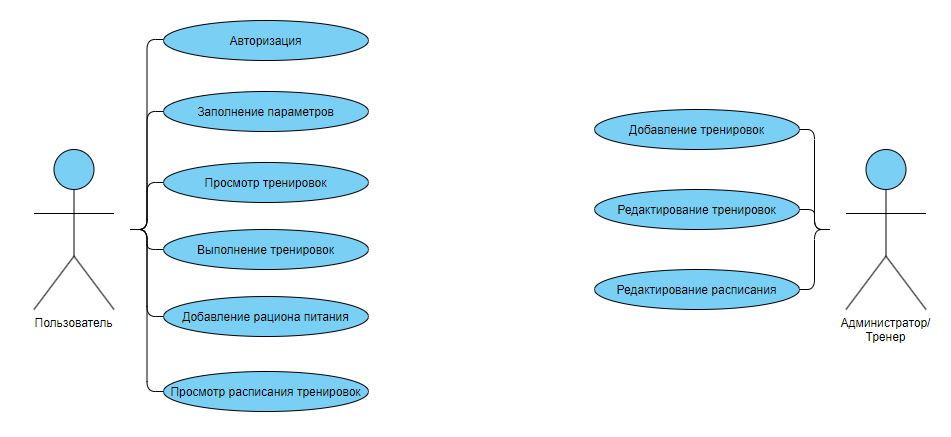


Рисунок 3 - Диаграмма вариантов использования

## 2.2 Диаграмма компонентов

Диаграмма компонентов (рисунок 4), в отличие от ранее рассмотренных диаграмм, описывает особенности физического представления системы. Диаграмма компонентов позволяет определить архитектуру разрабатываемой системы, установив зависимости между программными компонентами, в роли которых может выступать исходный, бинарный и исполняемый код. Во многих средах разработки модуль или компонент соответствует файлу. Пунктирные стрелки, соединяющие модули, показывают отношения взаимозависимости, аналогичные тем, которые имеют место при компиляции исходных текстов программ. Основными графическими элементами диаграммы компонентов являются компоненты, интерфейсы и зависимости между ними.

Компонент (component) — физически существующая часть системы, которая обеспечивает реализацию классов и отношений, а также функционального поведения моделируемой программной системы. Компонент предназначен для представления физической организации ассоциированных с ним элементов модели. Дополнительно компонент может иметь текстовый стереотип и помеченные значения, а некоторые компоненты – собственное графическое представление. Компонентом может быть исполняемый код отдельного модуля, командные файлы или файлы, содержащие интерпретируемые скрипты.

Модуль (module) — часть программной системы, требующая памяти для своего хранения и процессора для исполнения.

Важным этапом проектирования является построение диаграммы компонентов, представленной на следующем рисунке.

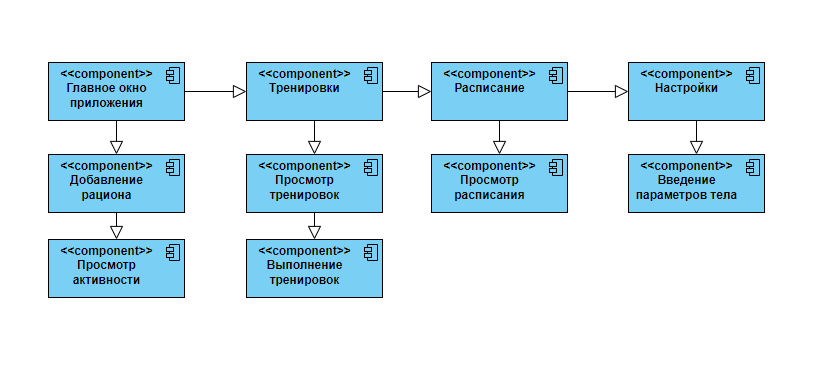


Рисунок 4 – Диаграмма компонентов

## 2.3 Диаграмма деятельности

Для моделирования процесса выполнения операций в языке UML используются диаграммы деятельности. Применяемая в них графическая нотация во многом похожа на нотацию диаграммы состояний, поскольку на диаграммах деятельности также присутствуют обозначения состояний и переходов. Отличие заключается в семантике состояний, которые используются для представления деятельности и действий, а также в отсутствии на переходах сигнатуры событий. Каждое состояние на диаграмме деятельности соответствует выполнению некой операции, а переход в следующее состояние происходит только после завершения выполнения этой операции. Диаграмма деятельности представляется в форме графа деятельности, вершинами которого являются состояния действия или деятельности, а дугами - переходы от одного состояния действия к другому.

Диаграммы деятельности на рисунке 5 – частный случай диаграмм состояний. Они позволяют реализовать в языке UML особенности процедурного и синхронного управления, обусловленного завершением внутренних действий и деятельности. Основным направлением использования диаграмм деятельности является визуализация особенностей реализации операций классов, когда необходимо представить алгоритмы их выполнения. При этом каждое состояние может являться выполнением операции определенного класса либо ее части, позволяя использовать диаграммы деятельности для описания реакций на внутренние события системы.

В контексте языка UML деятельность представляет собой совокупность отдельных вычислений, выполняемых автоматом. При этом отдельные элементарные вычисления могут приводить к результату или действию. На диаграмме деятельности отображается логика или последовательность перехода от одной деятельности к другой, при этом внимание фиксируется на результате деятельности. Сам же результат может привести к изменению состояния системы или возвращению некоторого значения. Диаграмма деятельности предназначена для моделирования поведения систем, хотя время в явном виде отсутствует на этой диаграмме.

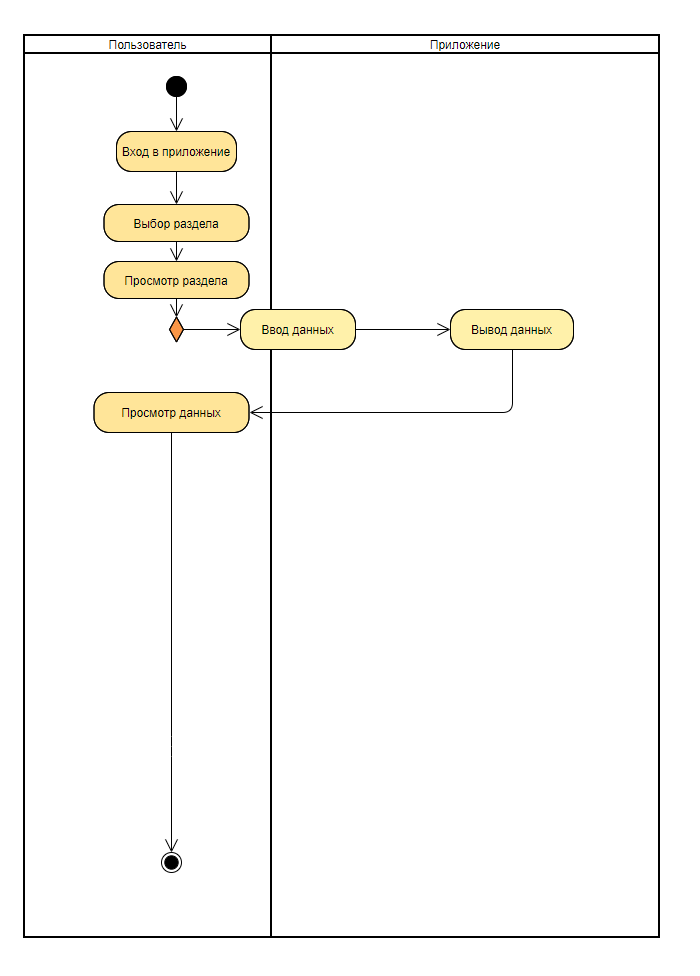


Рисунок 5 – Диаграмма деятельности

## 2.4 Диаграмма последовательности

Диаграмма последовательности (sequence diagram) – UML-диаграмма, которая представляет взаимодействие между линиями жизни как упорядоченную последовательность событий.

На рисунке 6 представлена диаграмма последовательности, описывающая процесс пользования приложением.

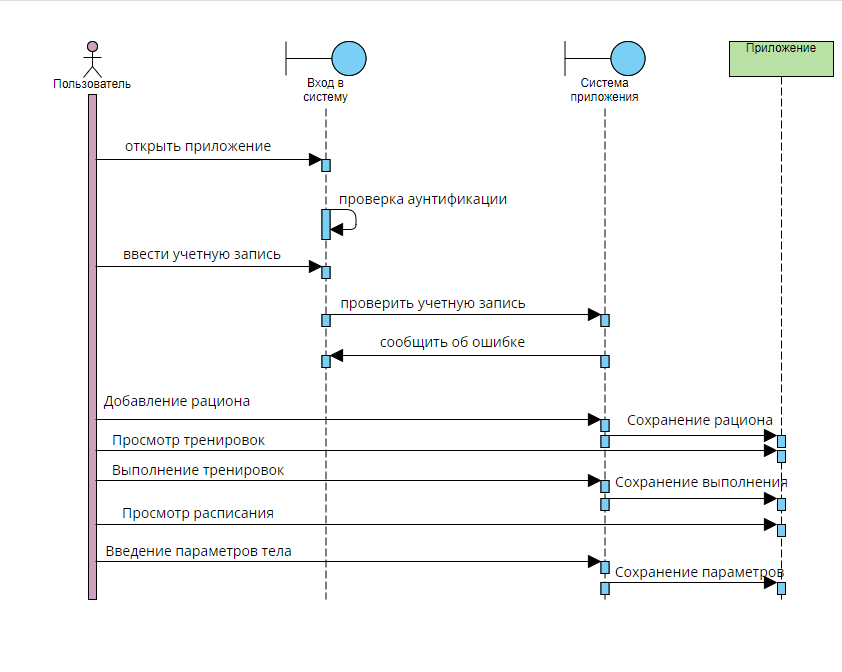


Рисунок 6 - Диаграмма последовательности

## 2.5 Разработка приложения

Инструменты, которые понадобятся в процессе разработки: Node.js, Vue.js, Firebase, редактор кода Visual Studio Code и так же терминал, в котором будут прописываться все команды Windows Terminal

Для начала нужно установить Windows Termina (оф. сайт: <https://www.microsoft.com/store/productId/9N0DX20HK701>)

Далее установка VS Code (оф. сайт: <https://code.visualstudio.com/>)

Далее Node.js с официального сайта <https://nodejs.org/en/>.

После того как Node.js установлен на компьютер, следует установить Vue CLI для сборки проектов на Vue. Процесс установки подробно описан на официальном сайте Vue.js: <https://cli.vuejs.org/ru/guide/installation.html>

В текущем случае установка производится пакетным менеджером npm.

Команда для установки Vue CLI: npm install -g @vue/cli

Как все необходимые инструменты установлены, следуя инструкции с оф. сайта Vue идет сборка проекта, для этого необходимо написать в Windows Terminal команду vue ui

Данная команда запустит UI интерфейс для сборки проекта (рисунок 7)

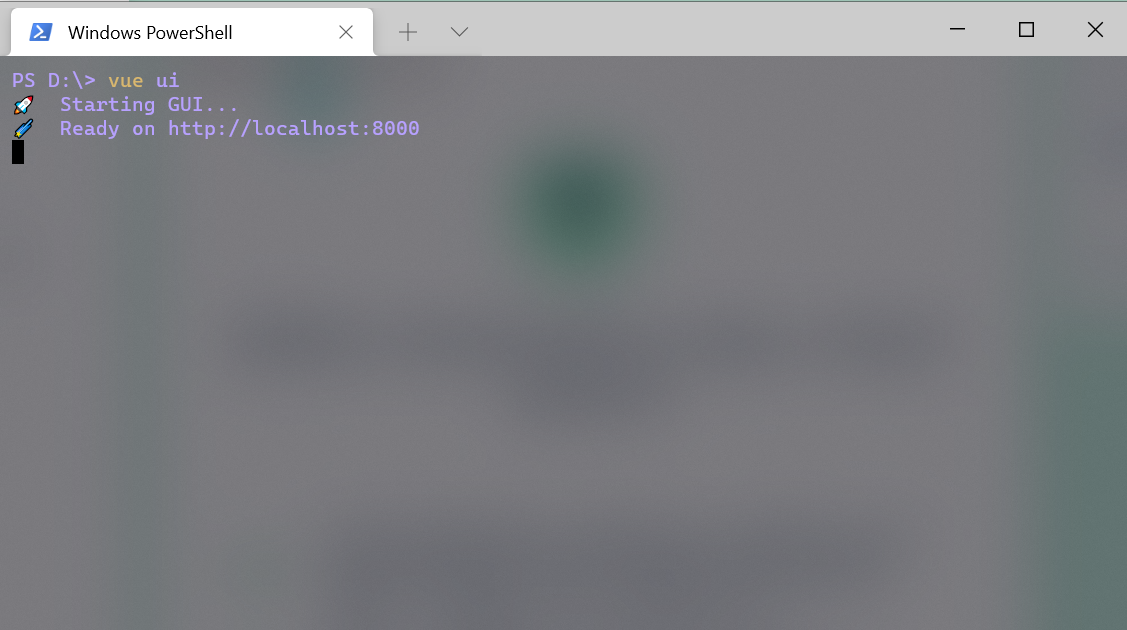


Рисунок 7 – командная строка

После успешного выполнения команды vue ui в терминале выводится сообщение “Ready on <http://localhost:8000>” и в браузере откроется окно «Менеджер проектов Vue» (рисунок 8).

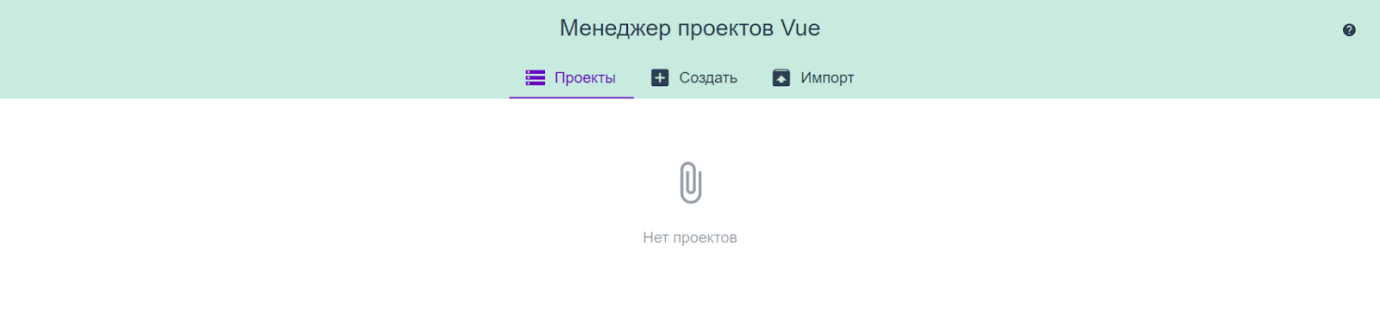


Рисунок 8 – менеджер проектов

Затем следует нажать кнопку «Создать», указать путь к проекту и его название (рисунок 9).

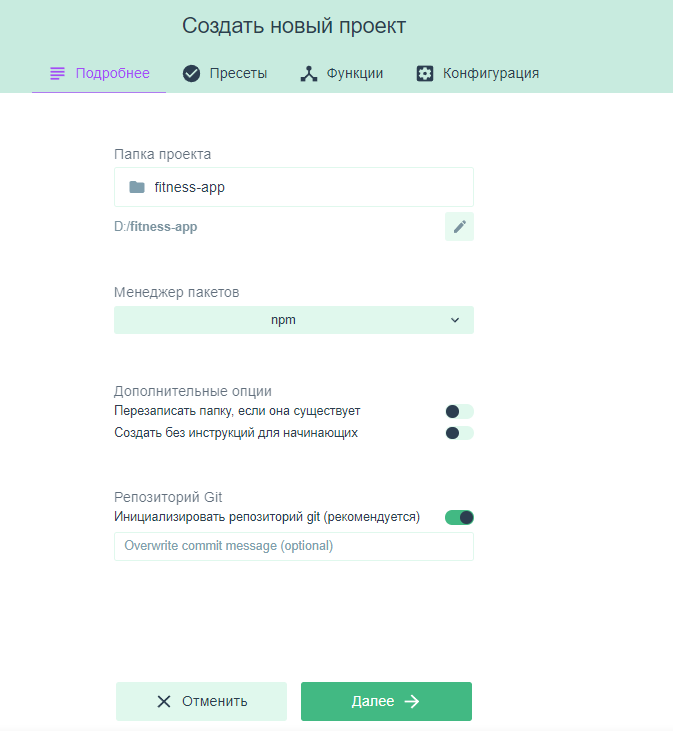


Рисунок 9 – создать проект

После того как название было указано нужно нажать кнопку «Далее», произойдет переход на страницу «Пресеты» (рисунок 10), там необходимо выбрать «Вручную», т.к. нужно установить несколько необходимых в процессе разработки функций

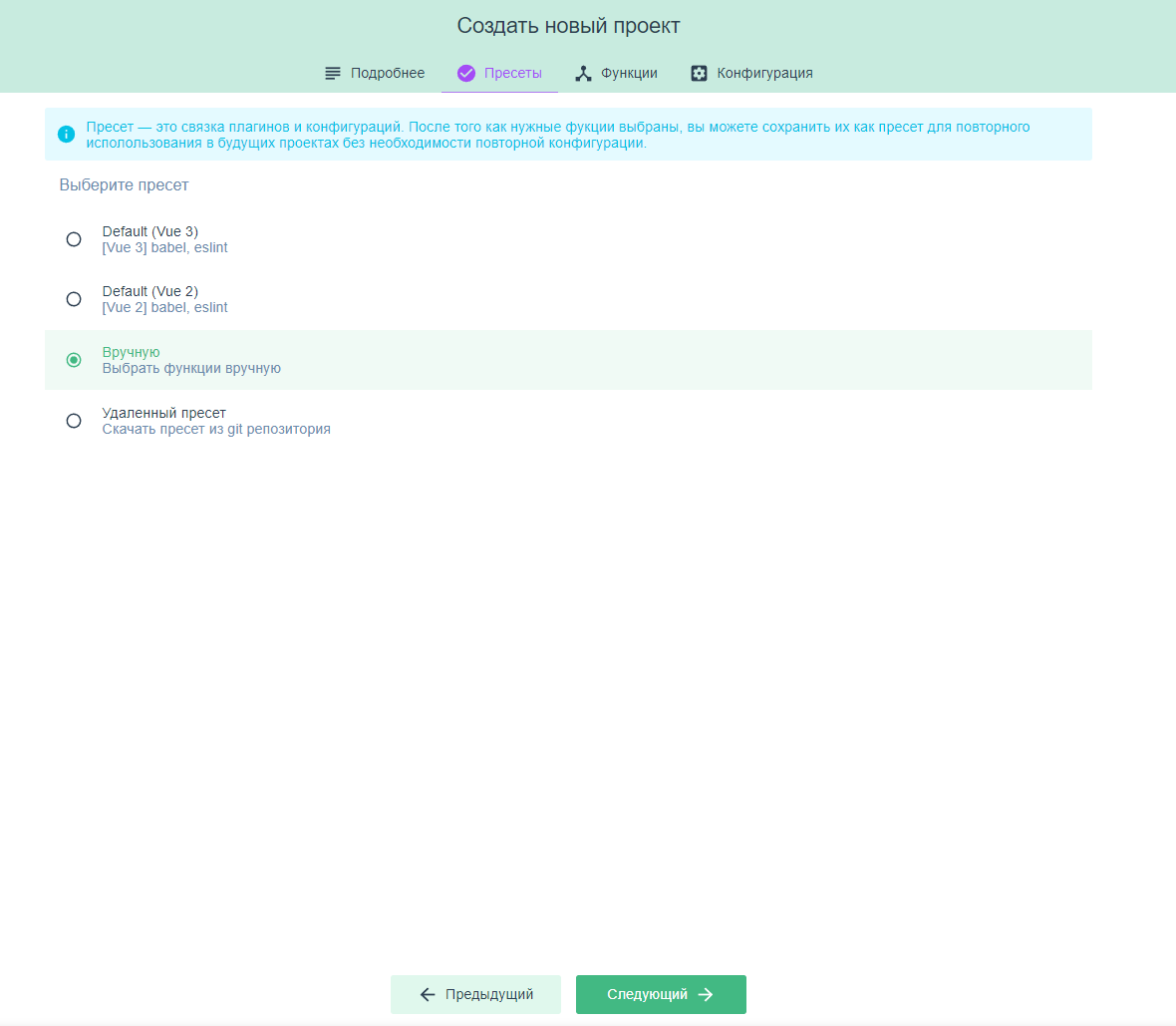


Рисунок 10 – пресеты

На следующей странице выбираем нужные функции (рисунок 11), к ним относятся: Chose Vue version(выбор версии Vue), Babel, Router, Vuex, Linter

Babel

Babel - это транспайлер, который переписывает код современного стандарта Javascript (ES2015) на более поздний.

Транспайлер - это программа, позволяющая менять исходный код одной программы на эквивалентный исходный код на другом языке. В случае с Babel, он переписывает современный Javascript на старый.

Vue Router

Официальная библиотека маршрутизации для Vue. js . Она глубоко интегрируется с Vue. js и позволяет легко создавать SPA-приложения.

Vuex

В центре любого Vuex-приложения находится хранилище. «Хранилище» — это контейнер, в котором хранится состояние вашего приложения. Два момента отличают хранилище Vuex от простого глобального объекта:

Хранилище Vuex реактивно. Когда компоненты Vue полагаются на его состояние, то они будут реактивно и эффективно обновляться, если состояние хранилища изменяется.

Нельзя напрямую изменять состояние хранилища. Единственный способ внести изменения — явно вызвать мутацию. Это гарантирует, что любое изменение состояния оставляет след и позволяет использовать инструментарий, чтобы лучше понимать ход работы приложения.

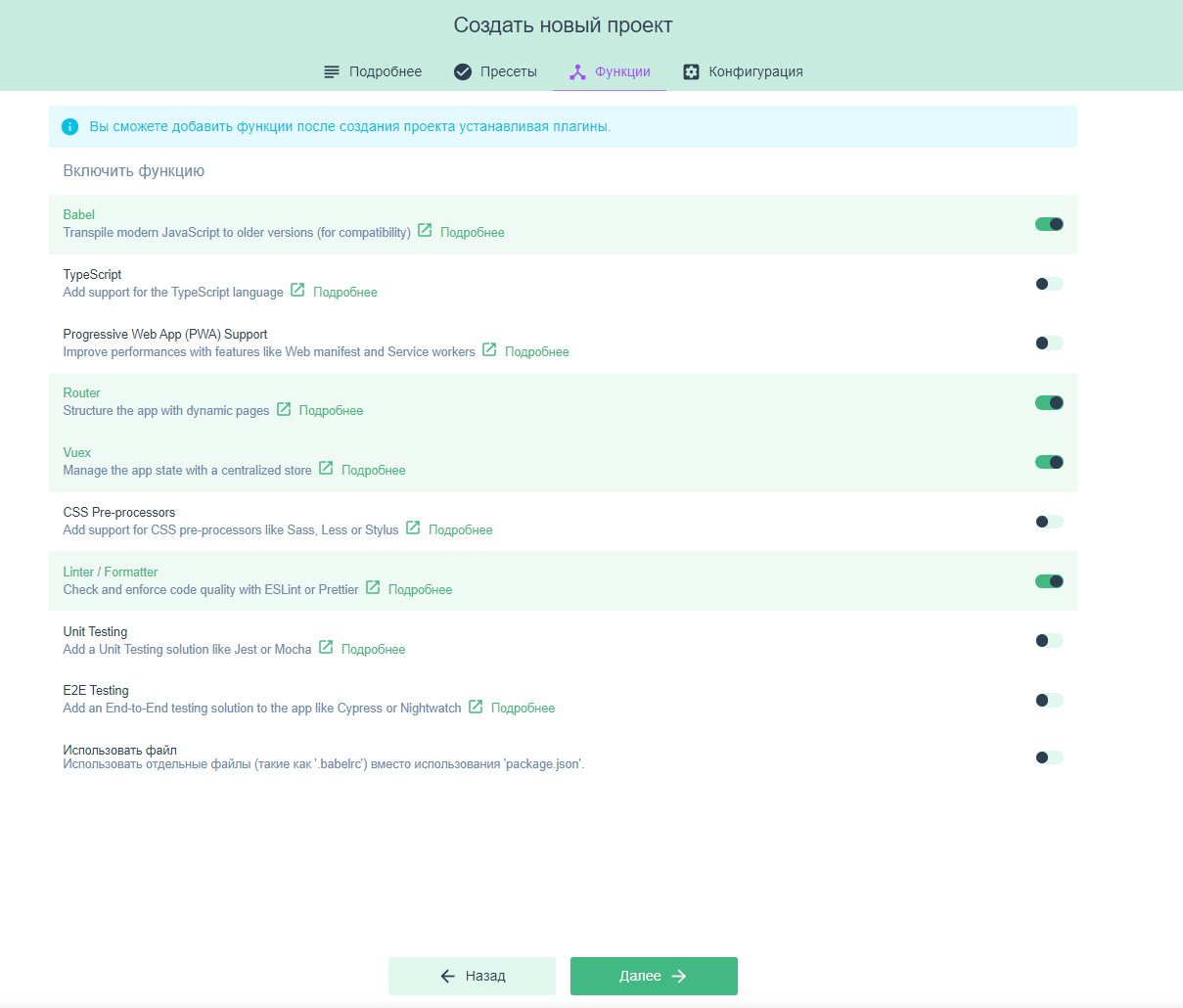


Рисунок 11 – функции

И в конце нужно выбрать нужную версию Vue. Установить в положение включено использование history mode в Router

В Linter необходимо выбрать ESLint with error prevention only - это будет выводить только предупреждения об ошибках в коде (рисунок 12).

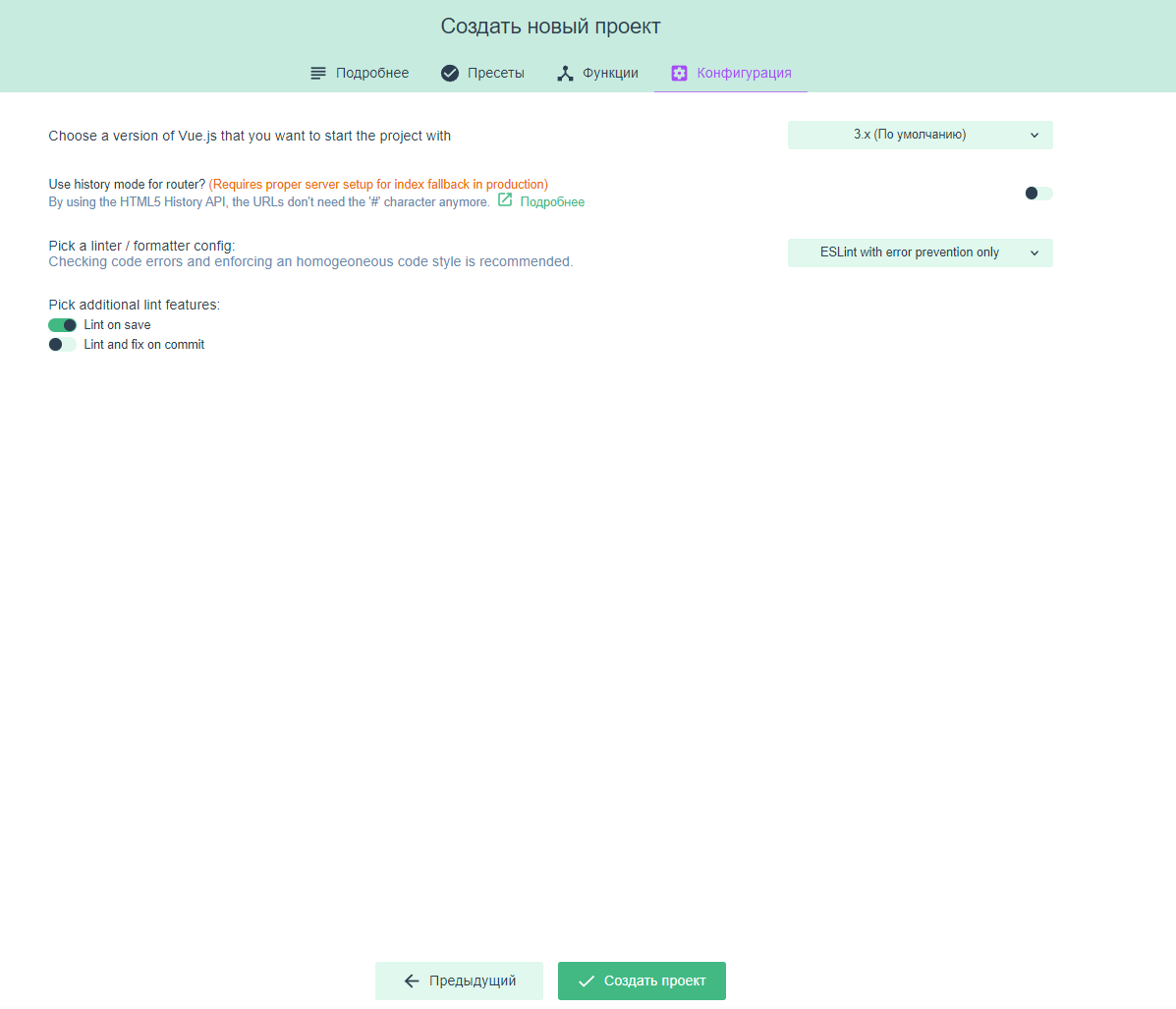


Рисунок 12 – конфигурации

После нажатия на кнопку запустится сборка проекта (рисунок 13).

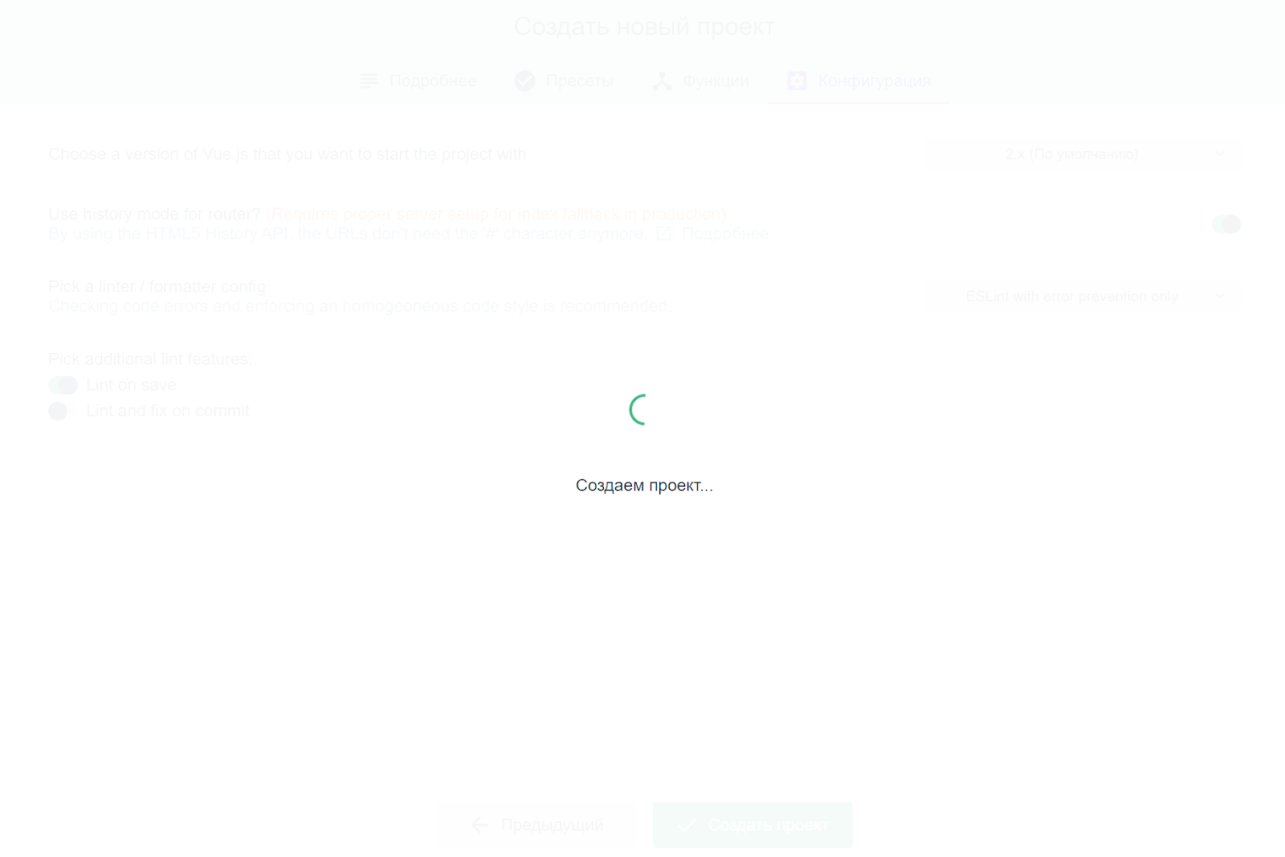


Рисунок 13 – сборка проекта

Ждем окончания установки, затем останавливаем Vue CLI в командной строке с помощью сочетания клавиш CTRL + C и пишем в консоль команду cd fitness-app; code .; npm run serve (рисунок 14);

Данная команда переместится в папку fitness-app запустит редактор кода VS Code и сервер с проектом.

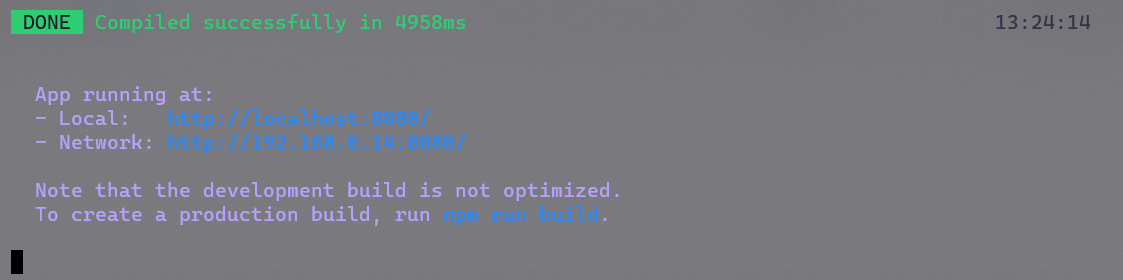


Рисунок 14 – запуск сервера

Также в процессе разработки потребуется Bootstrap (официальный сайт: <https://bootstrap-vue.org/docs>) для упрощения верстки, чтобы его установить необходимо выполнить команду в директории проекта:

npm install bootstrap bootstrap-vue --save

После того как процесс установки завершится нужно добавить в main.js следующий код:

import { BootstrapVue, IconsPlugin } from 'bootstrap-vue'

import 'bootstrap/dist/css/bootstrap.css'

import 'bootstrap-vue/dist/bootstrap-vue.css'

Vue.use(BootstrapVue)

Vue.use(IconsPlugin)

Данный код подключает компоненты Bootstrap для Vue и так же стили Bootstrap,

после чего добавляет на глобальном уровне компоненты BootstrapVue и IconsPlugin.

Создание базы данных в консоли Firebase (https://console.firebase.google.com/)

Чтобы создать базу данных необходимо перейти в консоль Firebase нажать кнопку Create a project и следовать инструкциям (рисунок 15).

Первым делом нужно ввести название проекта

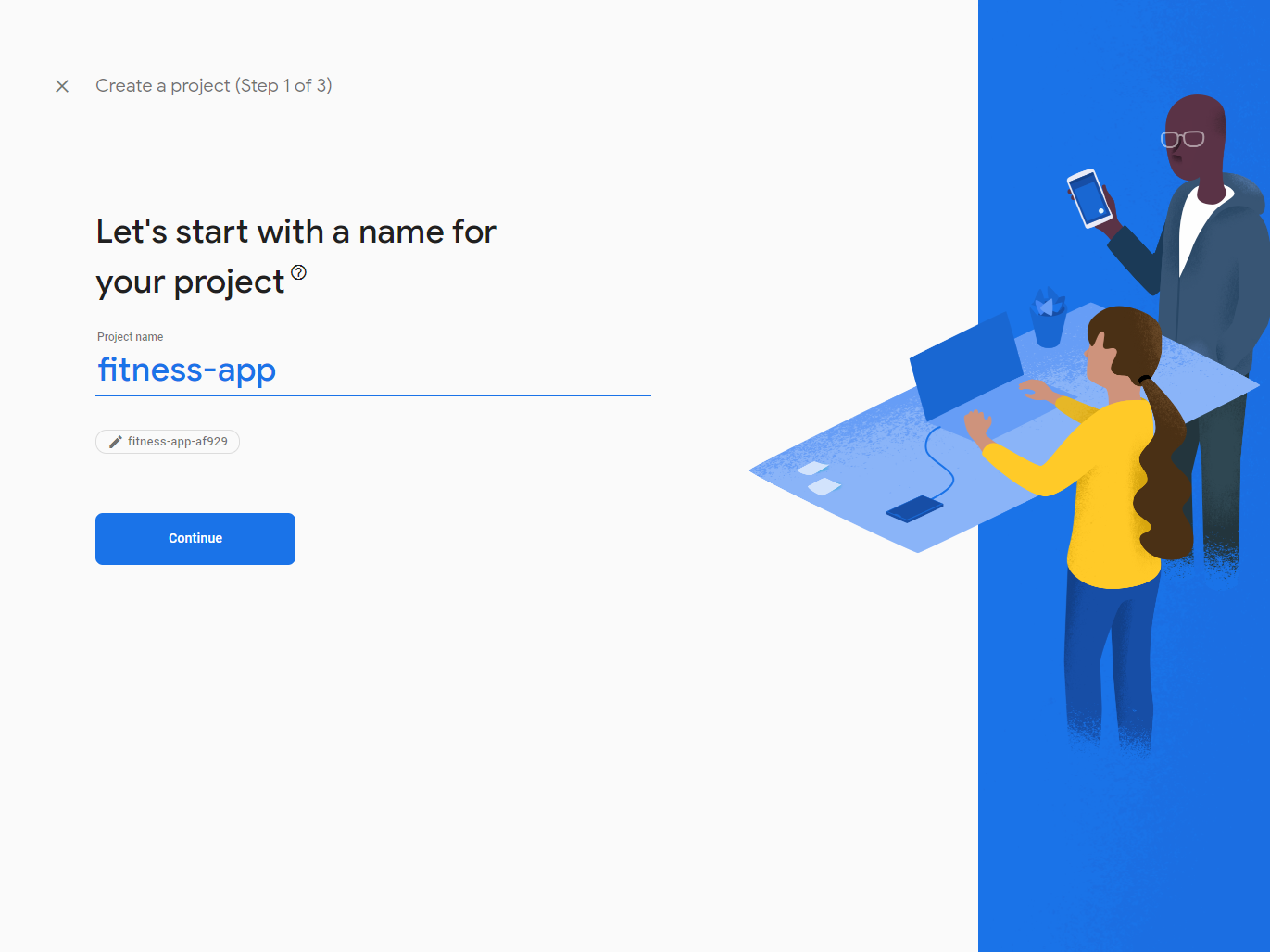


Рисунок 15 – создание БД

Затем нужно отключить аналитику т.к. в данном проекте она не потребуется и нажать кнопку «Create project» (рисунок 16).

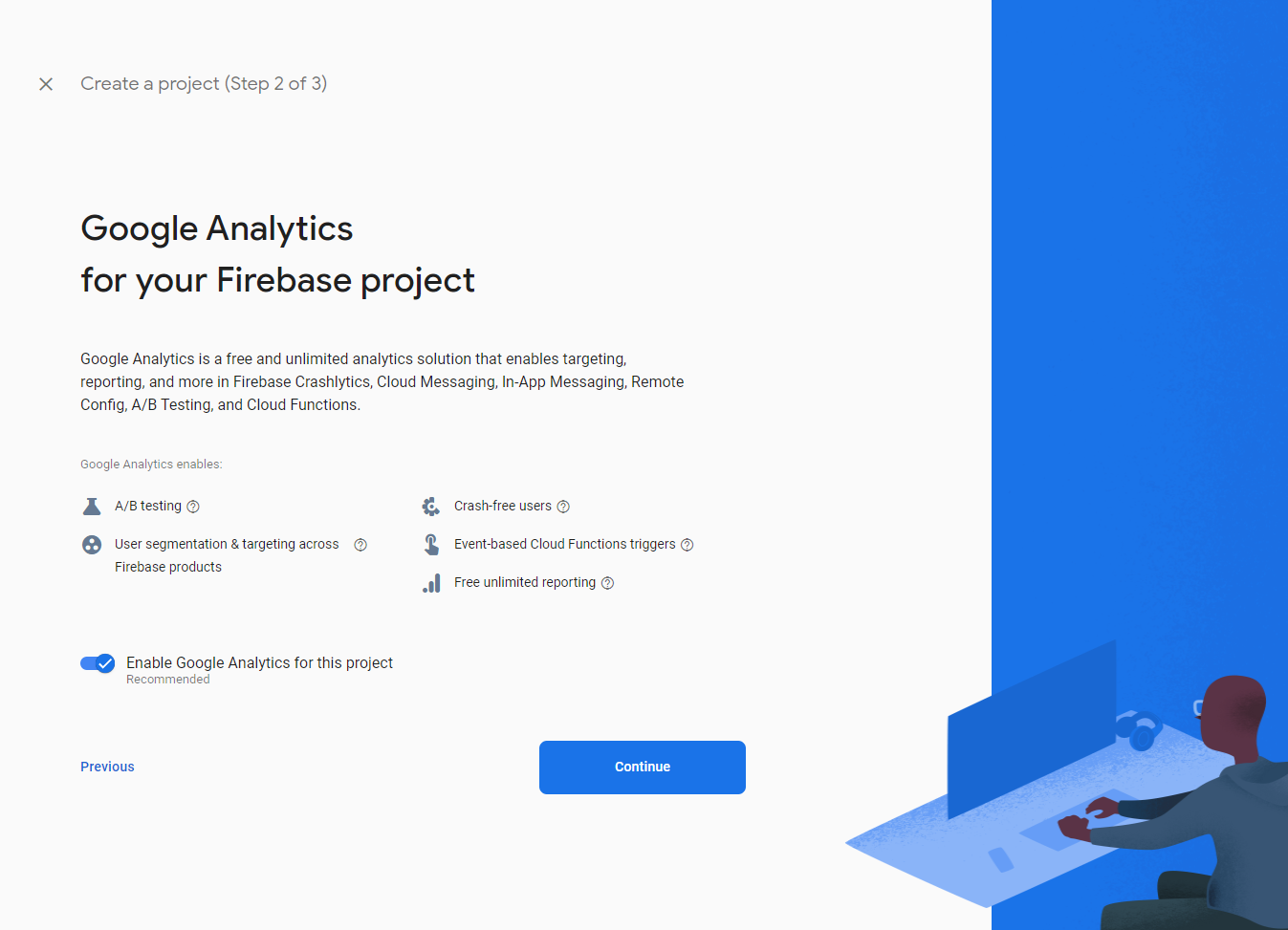


Рисунок 16 – отключение аналитики

Далее пойдет сборка базы проекта (рисунок 17).

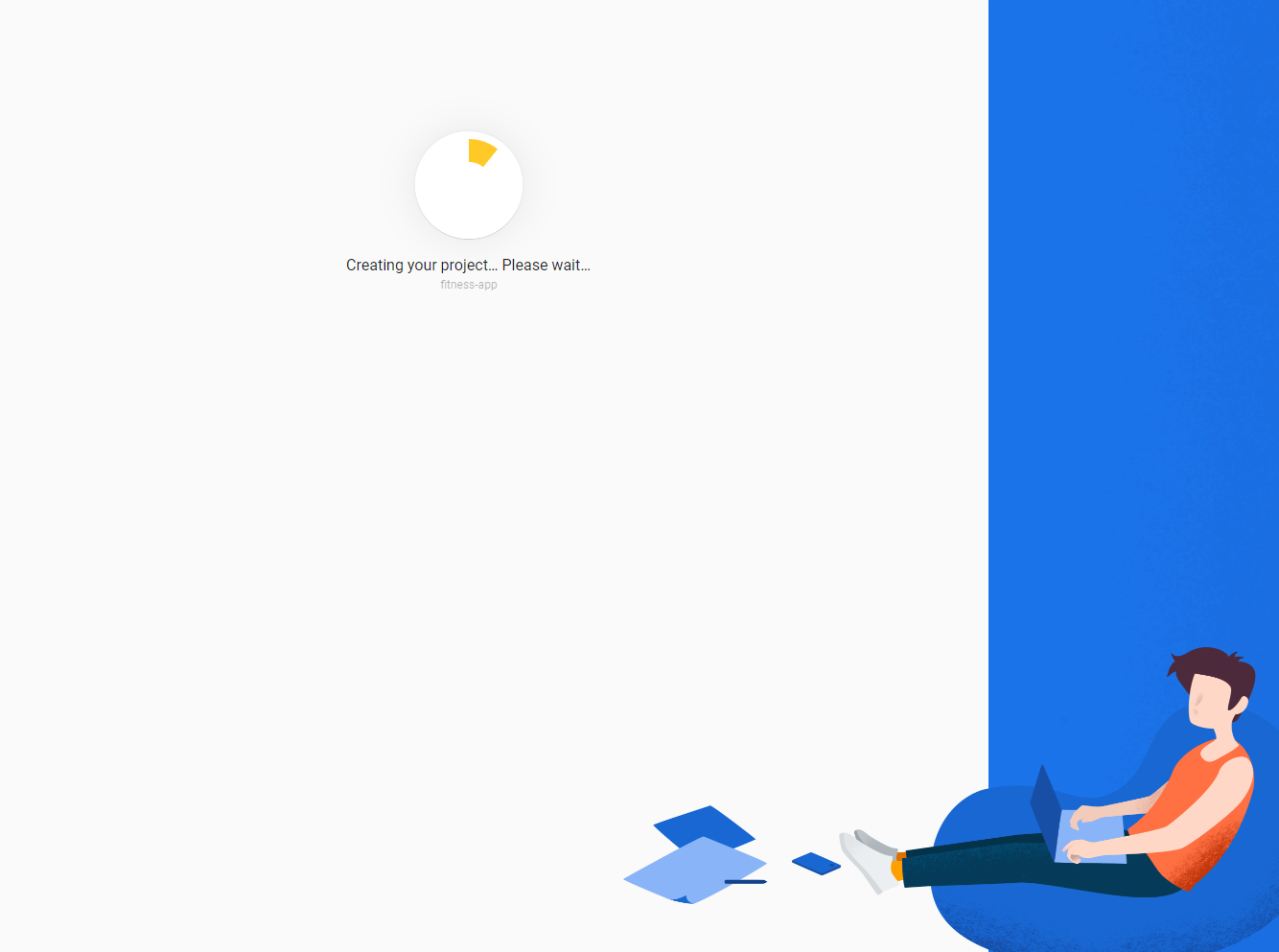


Рисунок 17 – сборка бд

После того как сборка пройдет, появится сообщение «Ваш новый проект готов» (рисунок 18).

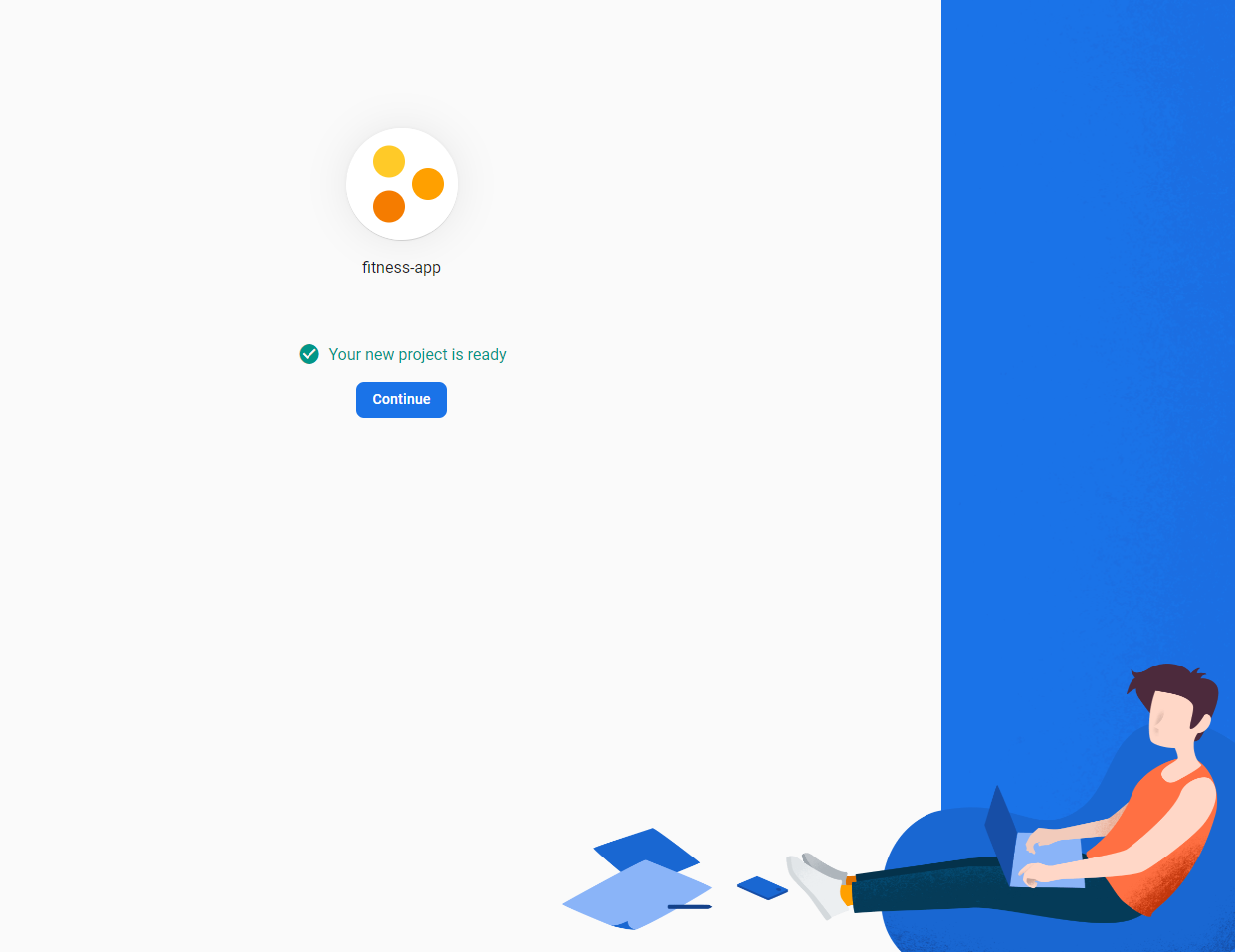


Рисунок 18 – проект готов

Далее на следующей странице в меню нужно выбрать раздел «Authentication» (рисунок 19)

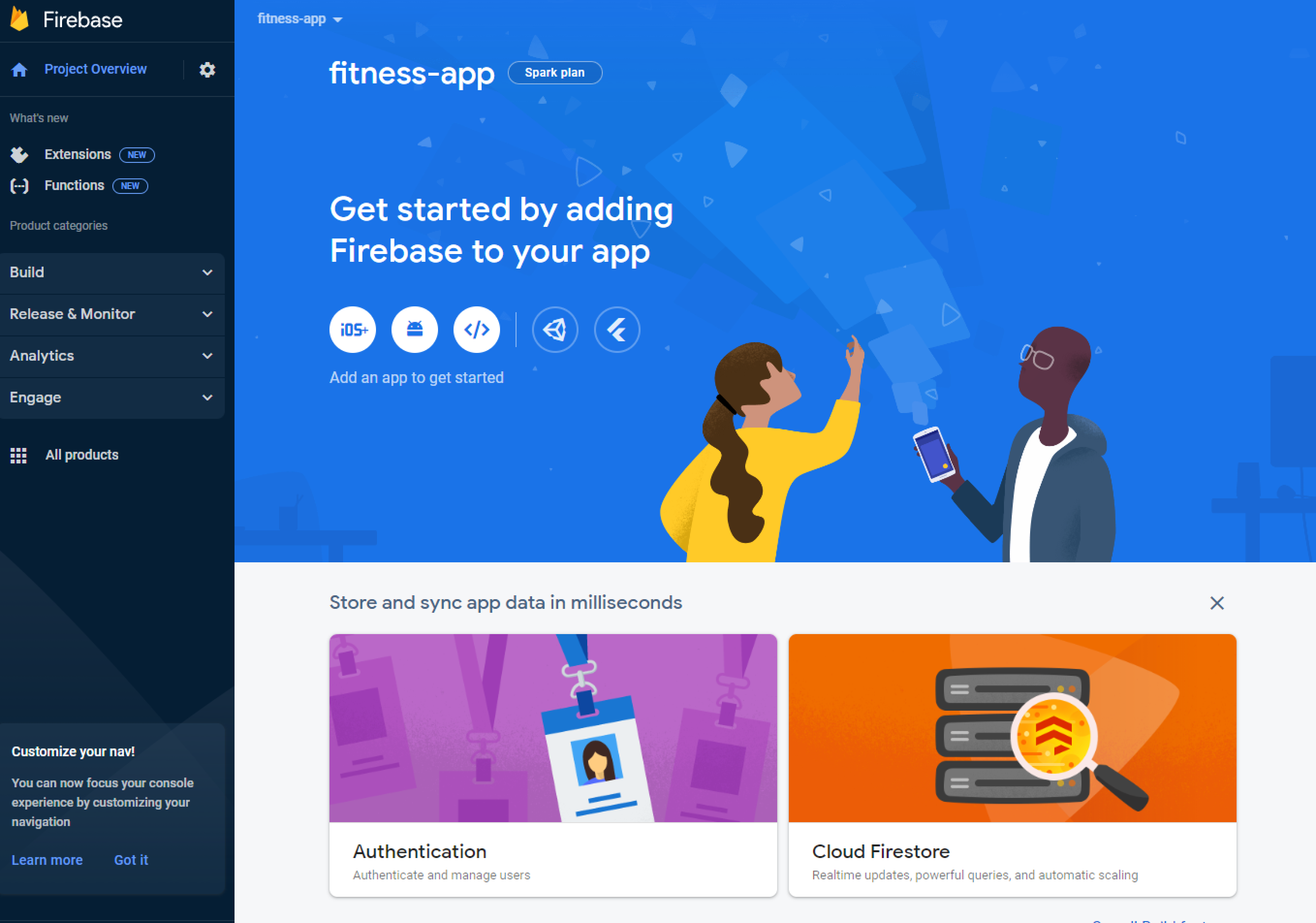


Рисунок 19 – авторизация

И нажать «Get started» (рисунок 20)

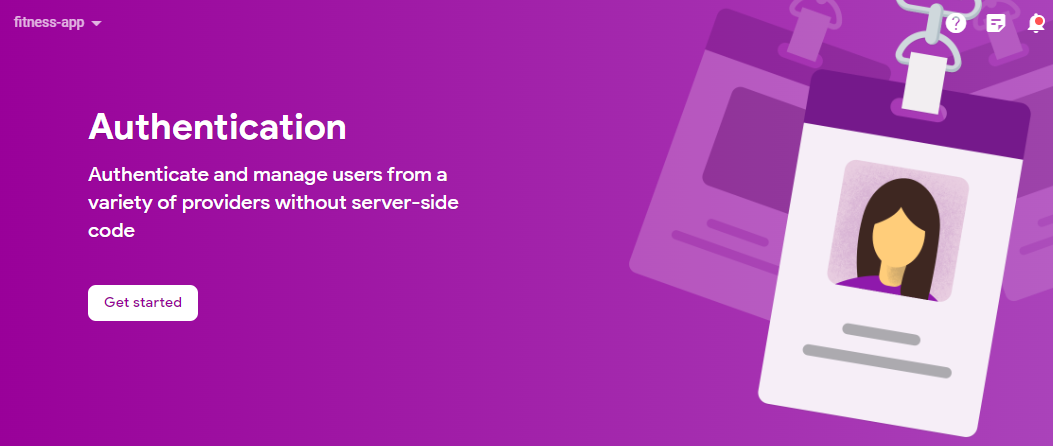


Рисунок 20 – Get started

Далее нужно выбрать метод авторизации в разделе “Sign-in method” на данном сайте будет использоваться только один метод авторизации: “Email/Password” (рисунок 21).

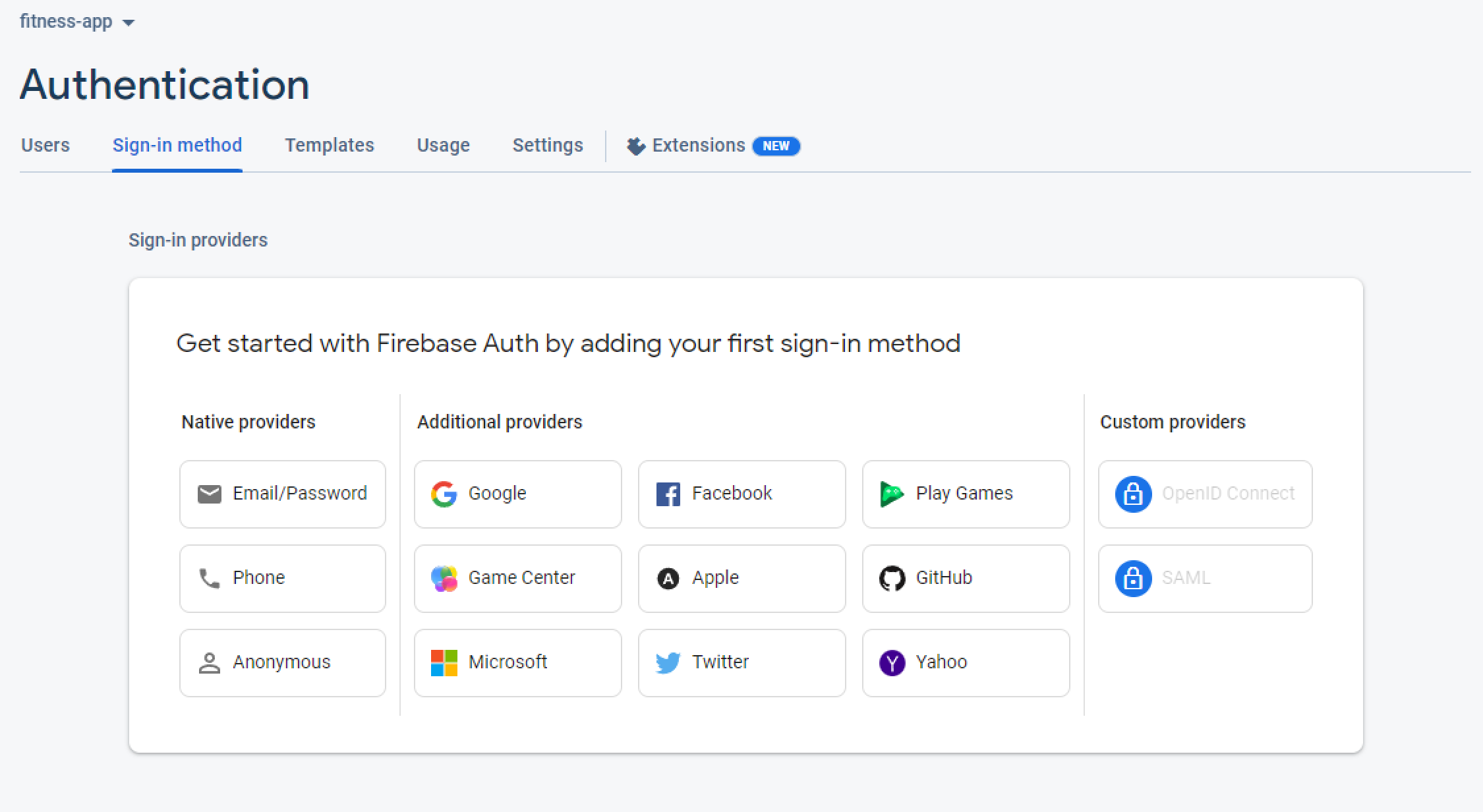


Рисунок 21– авторизация

Для активации данного метода авторизации нужно перейти в данный метод авторизации, активировать регистрацию пользователей по Email и нажать кнопку «Save» (рисунок 22).

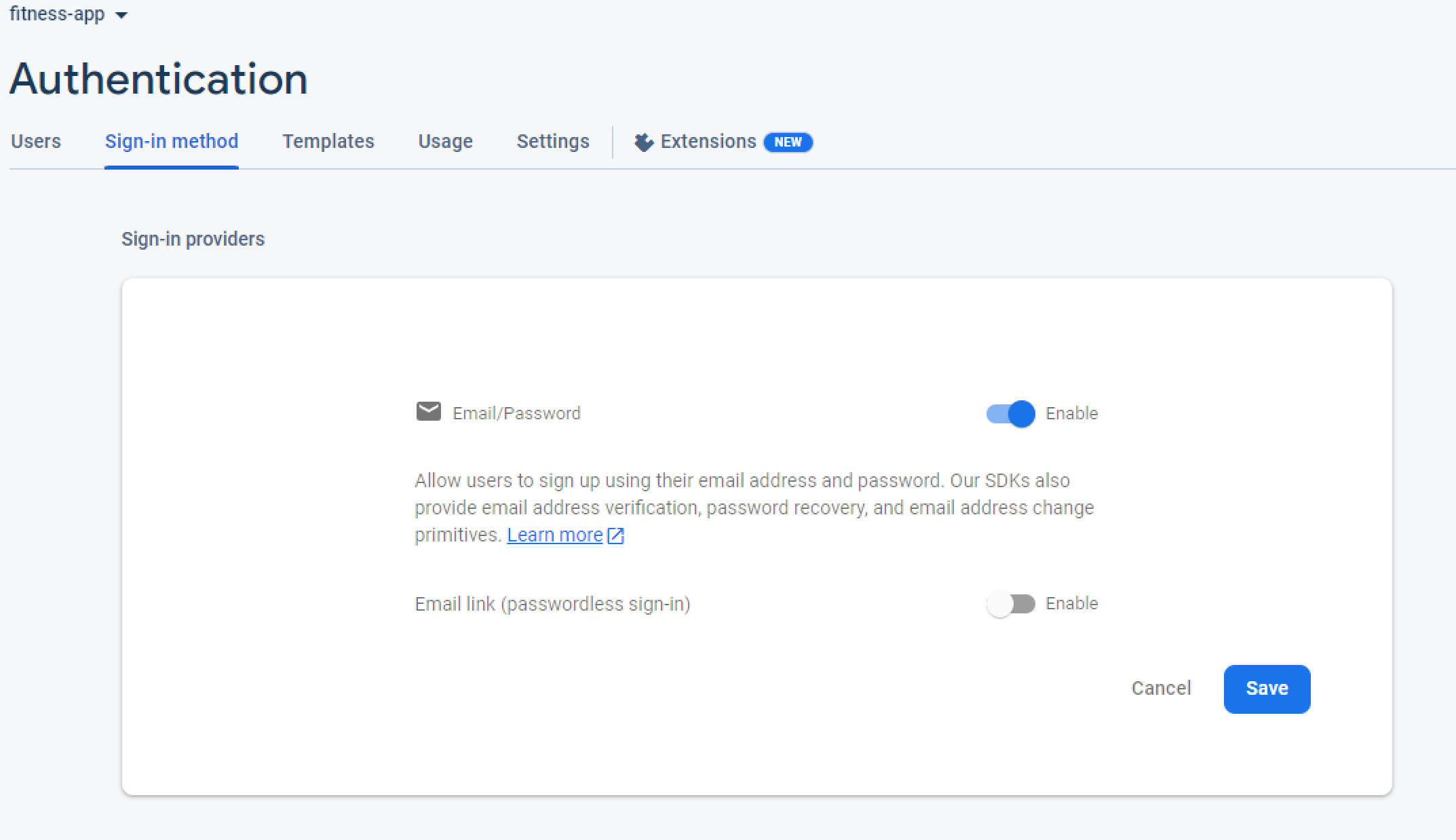


Рисунок 22 – активация авторизации по почте

Затем в меню следует перейти в раздел Realtime Database и создать базу для хранения данных (рисунок 23)

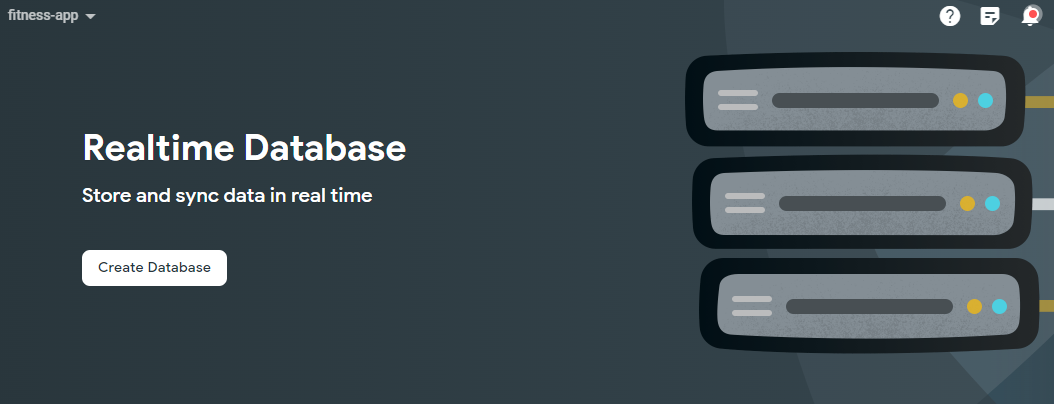


Рисунок 23 – создание БД

После активации появится окно, в котором нужно выбрать «Start in test mode» для режима разработки

Затем нужно выбрать сервер, на котором будут храниться данные для России оптимально выбирать европейские сервера.

Рассмотрим файл server.js (рисунок 24), в котором расположена серверная часть приложения. Полный код представлен в приложении А.

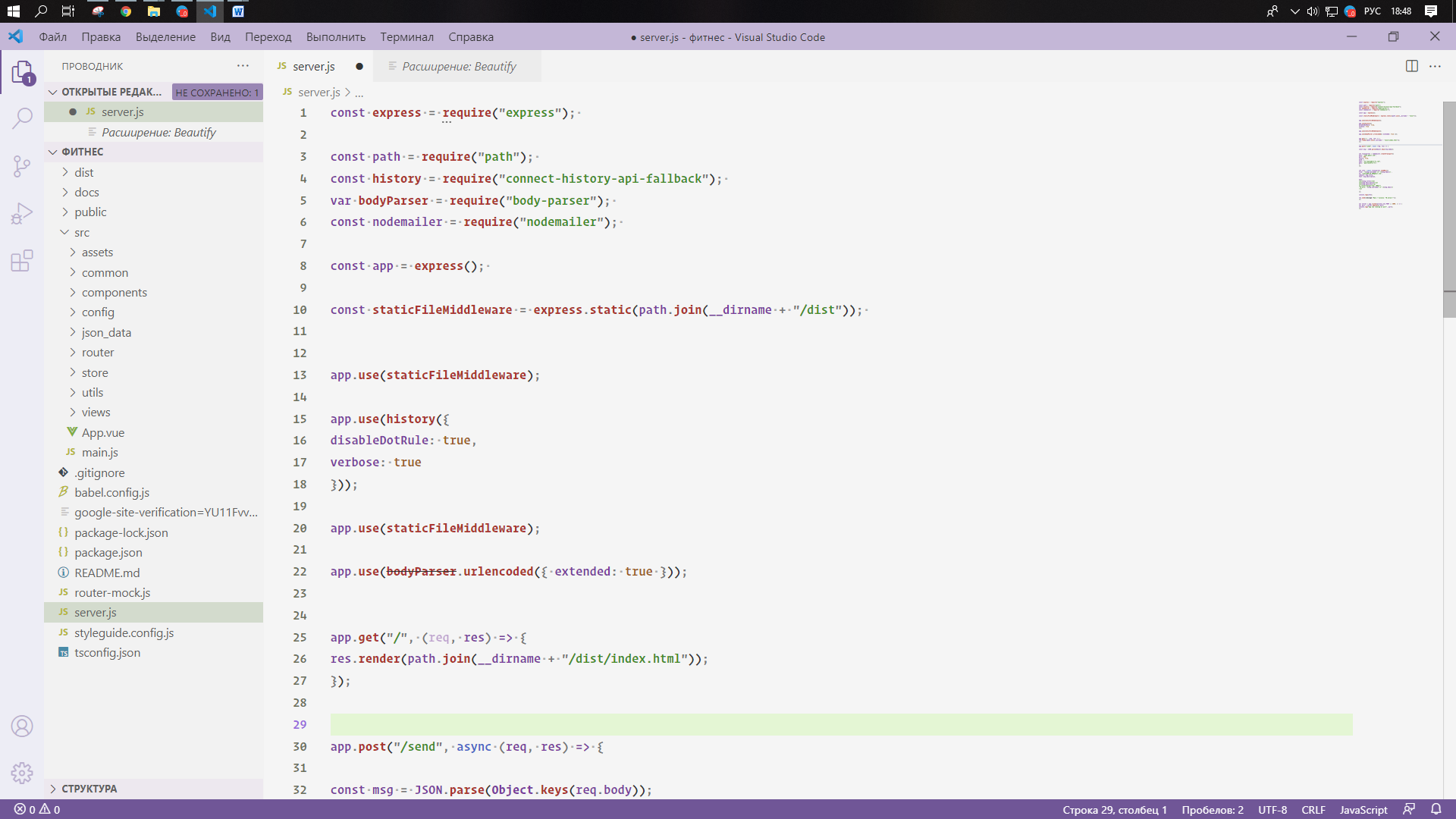


Рисунок 24 – Внешний вид файла server.js

const express = require("express"); - подключение фреймворка express

Express.js, или просто Express, фреймворк web-приложений для Node.js, реализованный как свободное и открытое программное обеспечение под лицензией MIT. Он спроектирован для создания веб-приложений и API. Де-факто является стандартным каркасом для Node.js.

js — Модуль Path. От автора: модуль Node js path используется для обработки и преобразования путей к файлам. Этот модуль можно импортировать, используя следующий синтаксис.

const path = require("path");

Это крошечное промежуточное ПО решает некоторые проблемы. В частности, он изменит запрошенное местоположение на указанный вами индекс (по умолчанию /index.html)

const history = require("connect-history-api-fallback");

var bodyParser = require("body-parser"); - библиотека для чтения тела запросов

const nodemailer = require("nodemailer"); - SMTP менеджер, для отправки сообщений

const app = express(); - создание приложения на бэкенда части

const staticFileMiddleware = express.static(path.join(\_\_dirname + "/dist")); - сохранение пути к статическим файлам, затем данный путь необходимо передать в express

app.use(staticFileMiddleware);

После того как путь к файлам указан, необходимо передать так же history с параметрами vebose.

По умолчанию это промежуточное ПО не регистрирует никакой информации. Если вы хотите активировать ведение журнала, вы можете сделать это с помощью verbose опции или указав функцию регистратора.

disableDotRule - отключает прямой доступ к файлам на сервере

app.use(history({

disableDotRule: true,

verbose: true

}));

Затем данный путь необходимо передать в express

app.use(staticFileMiddleware);

Запуск декодера запросов

app.use(bodyParser.urlencoded({ extended: true }));

Настройка get запроса - когда пользователь обращается к сайту, он всегда будет ссылаться на index.html в директории dist

app.get("/", (req, res) => {

res.render(path.join(\_\_dirname + "/dist/index.html"));

});

app.post("/send", async (req, res) => { - чтение пост запросов по адресу '/send'

const msg = JSON.parse(Object.keys(req.body)); - сохранение сообщения в формате JSON для удобной работы с ним

конфигурация SMTP:

let transporter = nodemailer.createTransport({

host: "smtp.gmail.com", - хост

port: 465, - порт

secure: true,

auth: { - авторизация

user: "no-replay@fitness-app.com",

pass: "qwerty123fitness-app ",

},

});

let info = await transporter.sendMail({ - отправка сообщения

from: `"${msg.username} 👻" <${msg.email}>`, - адрес отправителя

to: "diana07.12.00@gmail.com", - список получателей

subject: msg.title, - заголовок

text: msg.description, - тело сообщения

html: ` - сообщение в html формате

<h1>${msg.title}</h1>

<p>${msg.description}</p>

<p>${msg.question}</p>

<p style="margin-top: 100px">

Отправил: "${msg.username} 👻" <${msg.email}>

</p>

`,

});

console.log(info); - логирование отправок

res.send({message:"Ваше сообщение отправлено!"}); - ответ от сервера

});

Запуск сервера происходит с помощью callback функции listen из фреймворка express, в данную функцию необходимо передать порт, по которому будет обращаться сервер. После успешного запроса listen возвращает callback функция, в которой уже просто выводится в консоль сервера сообщение о том, на каком порту было запущен сайт

var server = app.listen(process.env.PORT || 3000, () => {

var port = server.address().port;

console.log("App now running on port", port);

});

Далее рассмотрим файл main.js (рисунок 25), в котором расположено глобальное подключение компонентов необходимых для работы приложения. Полный код представлен в приложении Б.

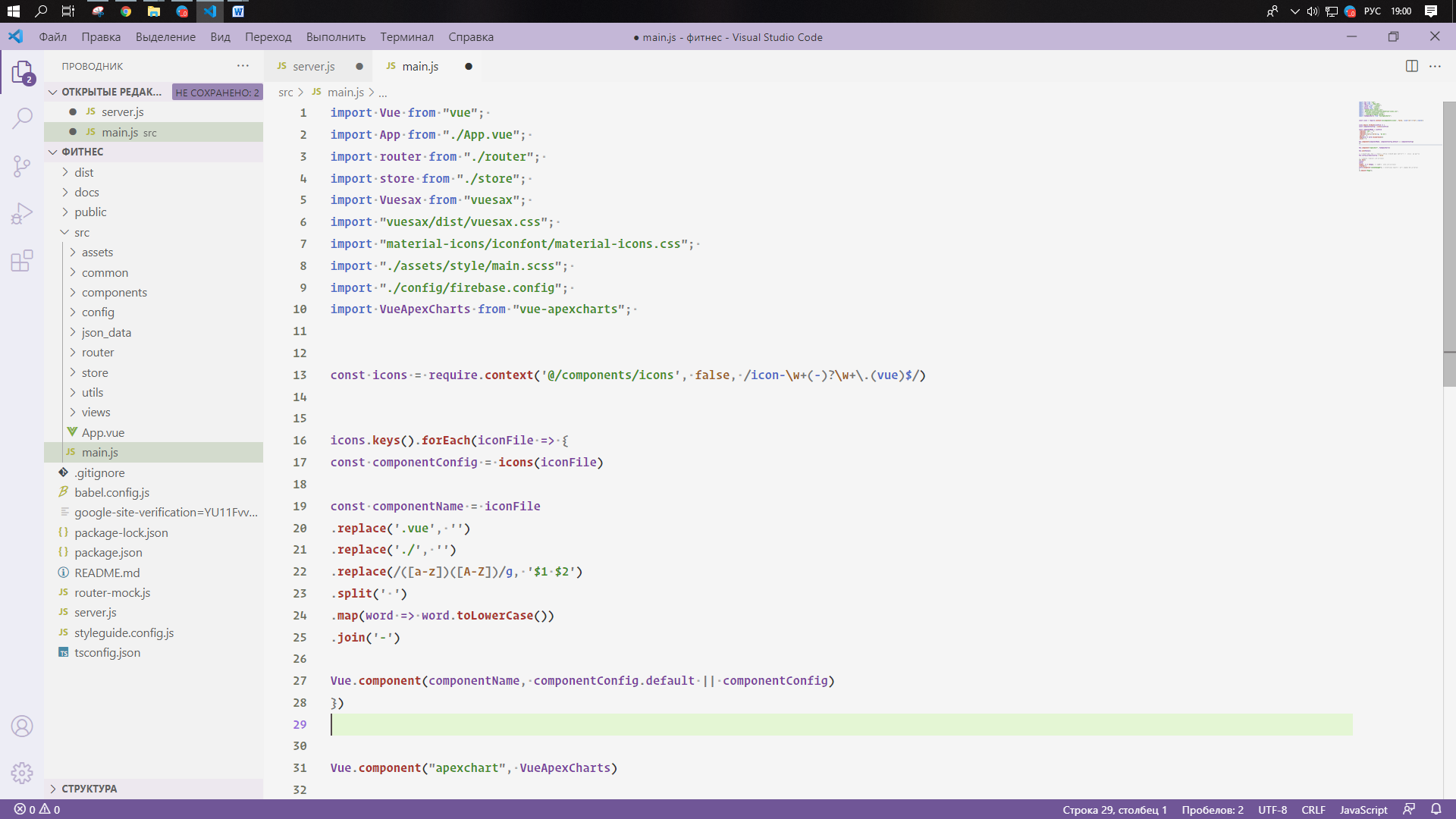


Рисунок 25 – Внешний вид файла main.js

import Vue from "vue"; - подключение фреймворка Vue

import App from "./App.vue"; - подключение стартового файла приложения

import router from "./router"; - подключение роутера приложения для навигации

import store from "./store"; - подключение хранилища

import Vuesax from "vuesax"; - подключение библиотеки компонентов

import "vuesax/dist/vuesax.css"; - подключение стилей библиотеки компонентов

import "material-icons/iconfont/material-icons.css"; - подключение стилей для иконок

import "./assets/style/main.scss"; - подключение своих стилей

import "./config/firebase.config"; - подключение файла конфигурации firebase

import VueApexCharts from "vue-apexcharts"; - подключение библиотеки диаграм

Получаем все компоненты иконок через webpack

const icons = require.context('@/components/icons', false, /icon-\w+(-)?\w+\.(vue)$/)

Регистрируем глобально все иконки

icons.keys().forEach(iconFile => {

const componentConfig = icons(iconFile)

const componentName = iconFile

.replace('.vue', '')

.replace('./', '')

.replace(/([a-z])([A-Z])/g, '$1 $2')

.split(' ')

.map(word => word.toLowerCase())

.join('-')

Vue.component(componentName, componentConfig.default || componentConfig)

})

Глобальная регистрация компонента диаграмм:

Vue.component("apexchart", VueApexCharts)

Vue.use(Vuesax); - добавление экземпляра хранилища в приложение Vue

конфигурация Vue - в данном случае конфигурация находится в режиме разработки

Vue.config.productionTip = false

new Vue({ - создание экземпляра приложения

router,

store,

render: h => h(App), - запуск рендера приложения

created(){

store.dispatch("stateChanged"); - получение данных об авторизации пользователя

}

}).$mount("#app");

Далее рассмотрим файл app.vue (рисунок 26), в котором расположены базовые компоненты приложения. Полный код представлен в приложении В.

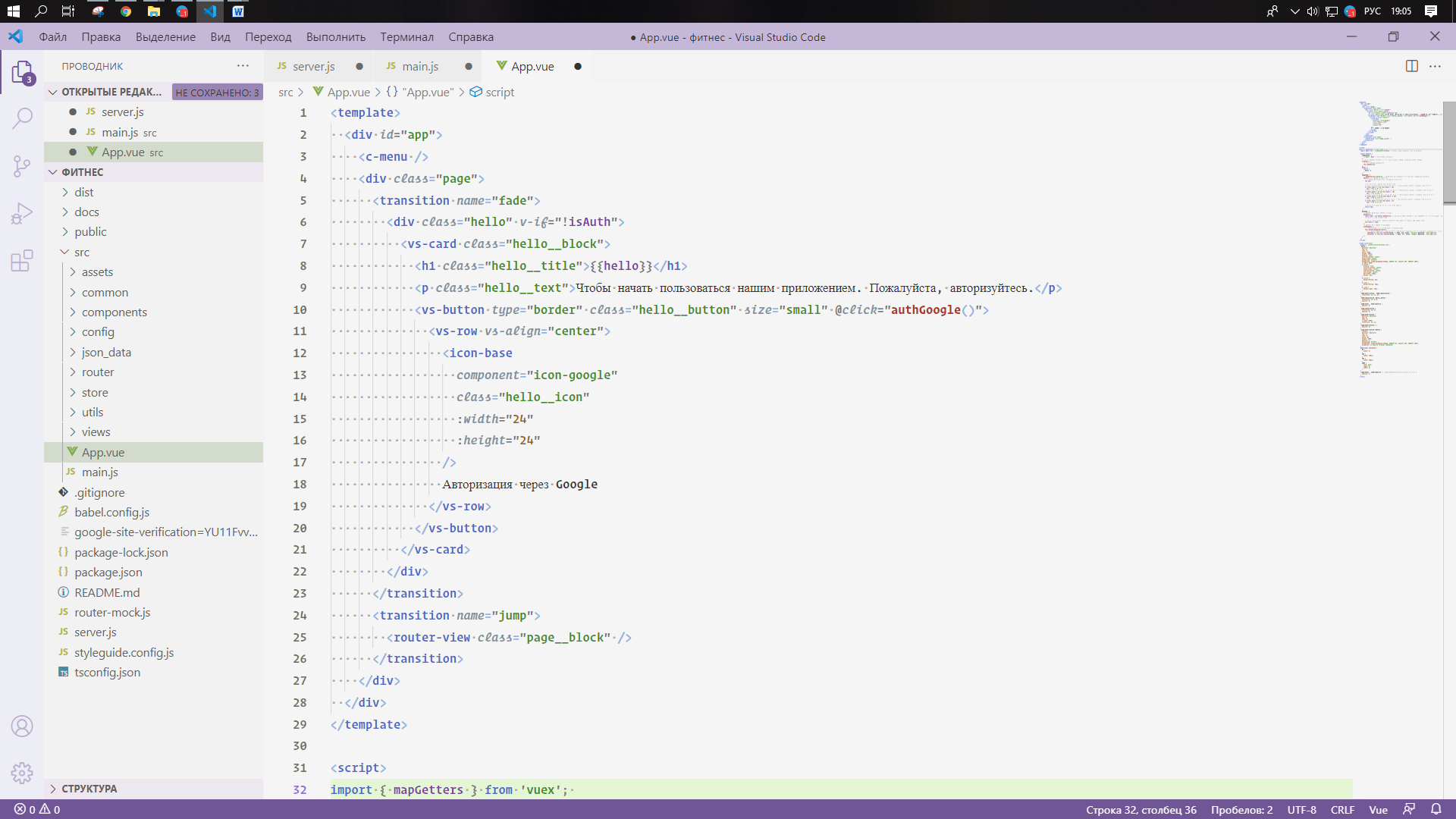


Рисунок 26 – Внешний вид файла app.vue

<script>

import { mapGetters } from 'vuex'; - подключение mapGetters из хранилища vuex, mapGetters необходим для удобной навигации по запрашиваемым данным из vuex

import menu from './components/ui/menu' - подключение компоненты меню приложения

export default {

components: {

'c-menu': menu - именование компоненты

},

Метод created вызывается в тот момент, когда текущая компонента была создана

created() {

this.getHours(); - вызов функции getHours()

},

data() {

return {

hours: 0

}

},

computed: {

...mapGetters(["isAuth"]), - получение из хранилища состояния авторизации пользователя

hello() { // функция приветствия

let say; - создание переменной для сохранения приветствия

Проверки для разных видов приветствий. Если время больше 7 и меньше 12 часов, то пользователь увидит сообщение "Доброе утро!"

if (this.hours > 7 && this.hours < 12)

say = "Доброе утро!";

Если время больше 12 и меньше 18 часов, то пользователь увидит сообщение "Добрый день!"

if (this.hours > 12 && this.hours < 18)

say = "Добрый день!";

Если время больше 18 и меньше 23 часов, то пользователь увидит сообщение "Добрый вечер!"

if (this.hours >= 18 && this.hours <= 23)

say = "Добрый вечер!";

Если время больше 0 и меньше 6 часов, то пользователь увидит сообщение "Доброй ночи!"

if (this.hours >= 0 && this.hours < 6)

say = "Доброй ночи!";

return say; - после прохождения проверок, ответ возвращается

}

},

methods: {

getHours() { - функция получение текущего времени

const time = new Date().getHours(); - из класса Date вызывается метод getHours, который возвращает текущее время, данный ответ сохраняется в переменную time, затем в переменную текущей компоненты this.hours сохраняется перменная time

this.hours = time;

},

authGoogle() { - функция авторизации через Google

this.$store.dispatch("auth") - отправка запроса к методу auth в хранилище Vuex

В случае успеха или ошибки, появится всплывающее сообщение, оно отобразит сообщение, которое вернулось из хранилища

.then(msg => this.$vs.notify({title: "",text: msg, color: "success",position: "top-right"}))

.catch(err => this.$vs.notify({title: "",text: err, color: "danger",position: "top-right"}));

},

}

};

</script>

Далее рассмотрим файл api.service.js (рисунок 27), в котором расположено взаимодействие с API сервисом. Полный код представлен в приложении Г.

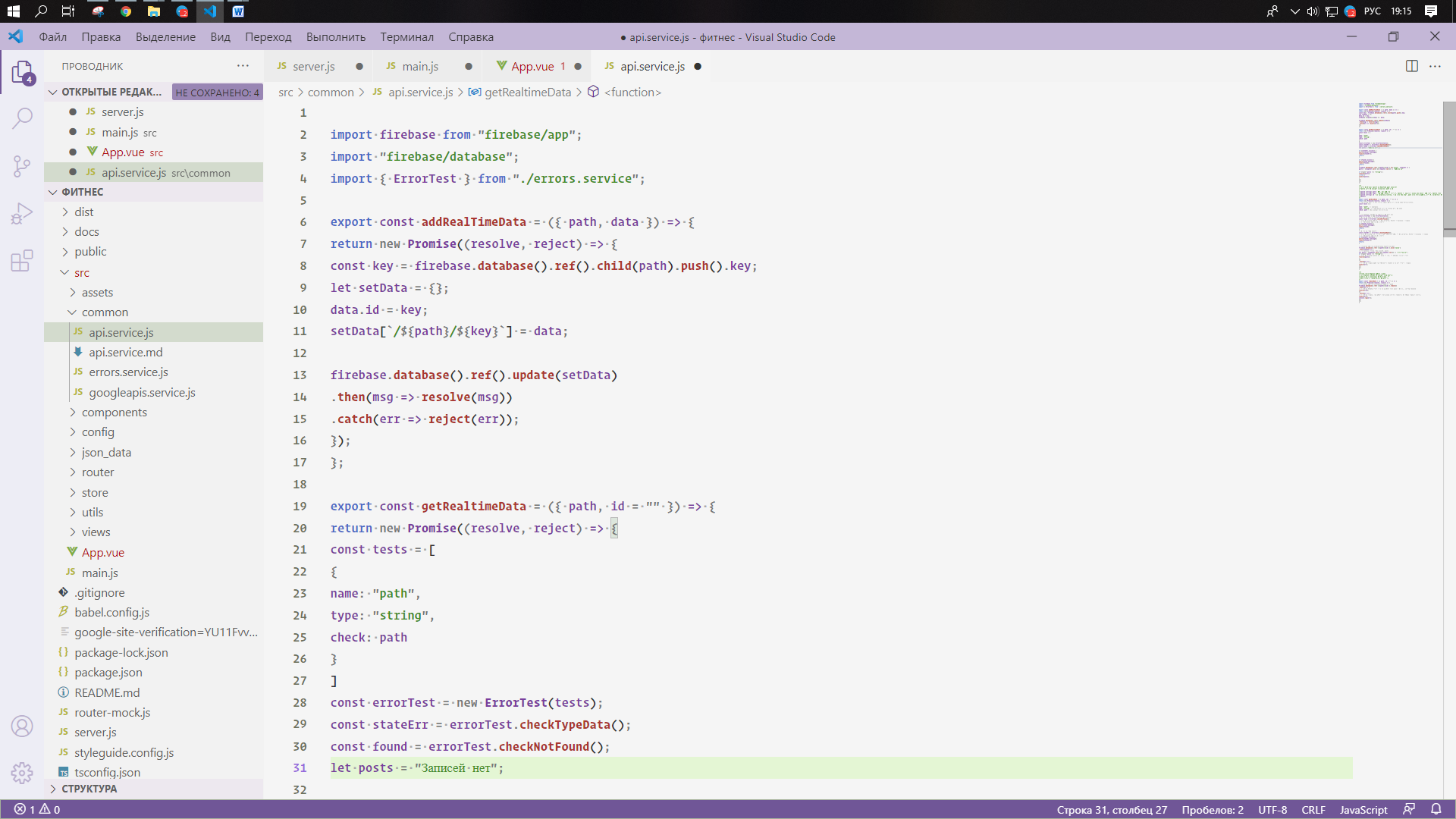


Рисунок 27 – Внешний вид файла api.service.js

Так как Firebase - нестабильная платформа (часто меняется политика, цены и т.д.), необходимо структурировать приложение так, чтобы в случае изменения каких-то требований, которые не устроят, можно было быстро сменить данную платформу, на какую-то другую.

Например: MongoDB. Для начала импортируем нужные пакеты: пакеты firebase и наш сервис ошибок

import firebase from "firebase/app";

import "firebase/database";

import { ErrorTest } from "./errors.service";

В коде ниже представлен метод записи в RealTime базу. RealTime база - это база, работающая в реальном времени, то есть все, что в нее будет записано, другой пользователь может видеть в реальном времени

Данный метод при записи добавляет к записи поле id, которое запрашивается из базы с помощью пустого запроса.

Принимается 3 параметра:

@param {string} path - путь для записи в базу. Например: news/lastNews

@param {object} data – данные, которые необходимо записать.

@returns {array} - возвращается promise с ответом от сервера (успех или ошибка)

export const addRealTimeData = ({ path, data }) => {

return new Promise((resolve, reject) => {

const key = firebase.database().ref().child(path).push().key;

let setData = {};

data.id = key;

setData[`/${path}/${key}`] = data;

firebase.database().ref().update(setData)

.then(msg => resolve(msg))

.catch(err => reject(err));

});

};

Метод чтения данных из RealTime базы в реальном времени. Данный метод принимает 2 значения path и id:

@param {string} path - путь для запроса

@param {string} id - не обязательное поле, сюда можно передать айди конкретной записи, которую необходимо получить

export const getRealtimeData = ({ path, id = "" }) => {

return new Promise((resolve, reject) => {

const tests = [

Метод получения данных из RealTime базы единожды. Данный метод принимает 2 значения path и id:

@param {string} path - путь для запроса

@param {string} type - тип запроса(on - смотреть данные в реаьном времени или once - запросить данные один раз)

@param {string} id - не обязательное поле, сюда можно передать айди конкретной записи, которую необходимо получить

export const getOnceData = ({ path, id = "" }) => {

return new Promise((resolve, reject) => {

const tests = [ - массив параметров для тестирования пути, который передает пользователь

{

name: "path", - название

type: "string", - тип данных, который должен быть передан

check: path - переменная для проверки

}

]

const errorTest = new ErrorTest(tests); - сохранение экземпляра класса с параметрами

const found = errorTest.checkNotFound(); - проверка наличия необходимых переменных

Если переменная пуста, то пользователь увидит сообщение о ошибке и выполнение запроса прервется

if (found.success) {

alert(found.message);

reject(found);

return;

}

const stateErr = errorTest.checkTypeData(); - проверка типа данных

Если переменная не соответствует нужному типу, то пользователь увидит сообщение об ошибке и выполнение запроса прервется

if (stateErr.success) {

alert(stateErr.message);

reject(stateErr);

return;

}

firebase.database().ref(`/${path}/${id}`).once('value') - отправка запроса на получение данных из базы

.then(snapshot => {

var posts = (snapshot.val() && snapshot.val()) || 'Новостей нет'; - сохранение ответа в переменную posts

if (typeof posts !== "string") {

Если переменная posts не равна строке, то функция вернет ответ

resolve(posts);

}

})

.catch(err => {

В случае ошибки запроса выполнится reject и вернет ответ об ошибке

reject(err);

});

});

}

Асинхронное удаление записи с базы. Для удаления функция принимает 2 параметра: path и id со значением «по умолчанию». path является обязательным параметром

export const removeData = ({ path, id = "" }) => {

return new Promise((resolve, reject) => {

firebase.database().ref(`/${path}/${id}`).remove() - отправка запроса на удаление записи в базу

.then(res => {

В случае успеха ответ с сервера будет возвращен с помощью метода resolve

resolve(res);

})

.catch(err => {

В случае ошибки, она будет возвращена методом reject и продублирована в консоль

reject(err);

console.log(err);

});

});

}

Далее рассмотрим файл googleapis.service.js (рисунок 28), в котором расположено взаимодействие с Google API. Полный код представлен в приложении Д.

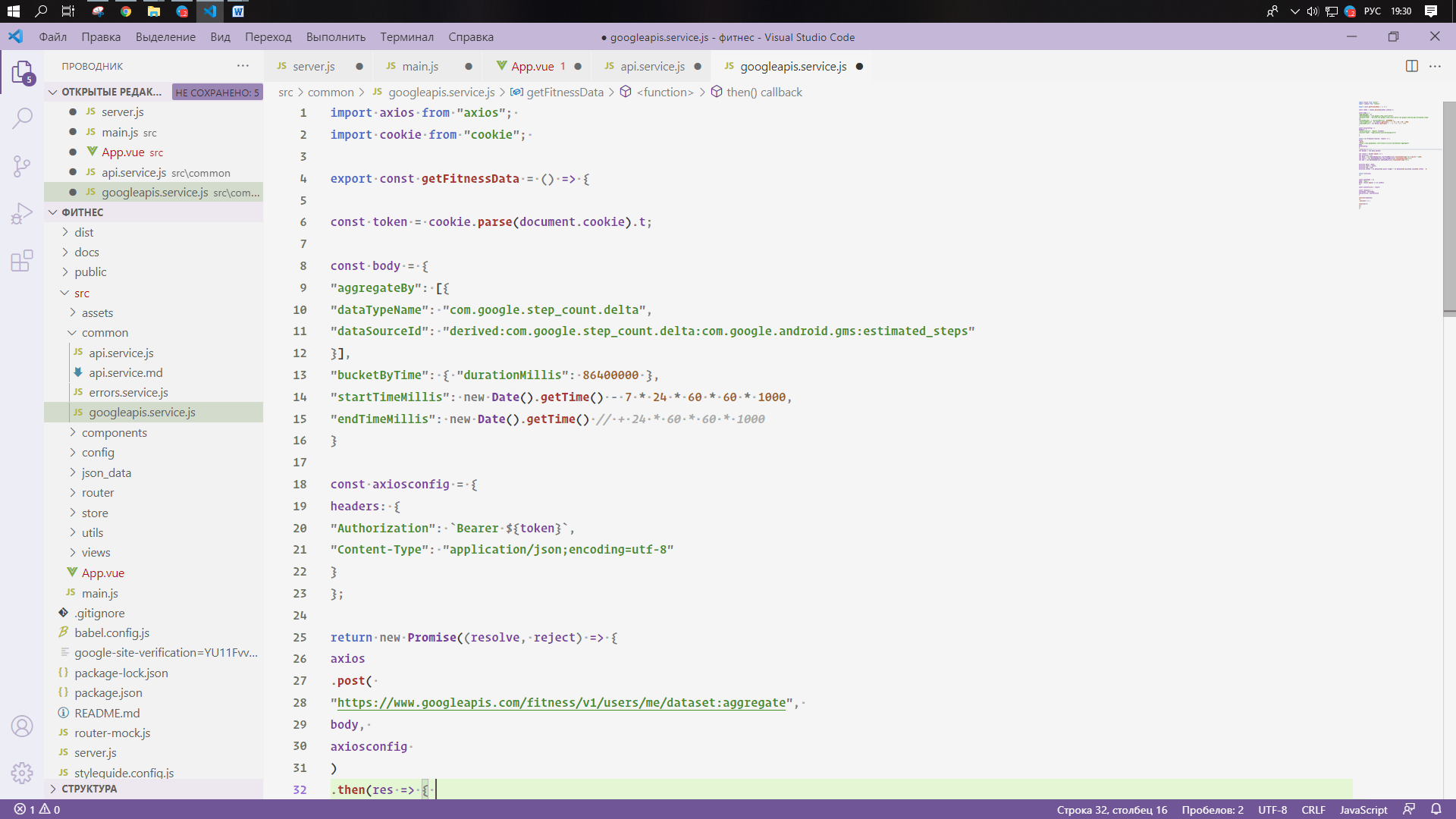


Рисунок 28 – Внешний вид файла googleapis.service.js

import axios from "axios"; - подключение axios для взаимодействия с google API

import cookie from "cookie"; - подключение библиотеки cookie для удобного взаимодействия с куки

Функция запроса статистики по шакам пользователя

export const getFitnessData = () => {

Получение токена из куки, который необходимо отправить в google API т.к. для взаимодействия с апи используется авторизованное подключение

const token = cookie.parse(document.cookie).t;

Создание тела запроса на основе документации Google Fit. В данном теле запроса указываются такие параметры как данные, которые необходимо получить и промежуток времени за который нужны эти данные

Данные, которые приходят с Google Fit при отправке данного запроса: количество пройденных шагов за последние 7 дней

const body = {

"aggregateBy": [{

"dataTypeName": "com.google.step\_count.delta",

"dataSourceId": "derived:com.google.step\_count.delta:com.google.android.gms:estimated\_steps"

}],

"bucketByTime": { "durationMillis": 86400000 },

"startTimeMillis": new Date().getTime() - 7 \* 24 \* 60 \* 60 \* 1000,

"endTimeMillis": new Date().getTime() // + 24 \* 60 \* 60 \* 1000

}

Конфигурация Axios. В шапке запроса отправляется токен авторизации и так же тип контента - это JSON в кодировке utf-8

const axiosconfig = {

headers: {

"Authorization": `Bearer ${token}`,

"Content-Type": "application/json;encoding=utf-8"

}

};

Так как данная функция является асинхронной, она возвращает Promise (в переводе: Обещание)

В случае успешного запроса вернется resolve, а в случае ошибки reject

return new Promise((resolve, reject) => {

axios

.post( - отправка пост запроса с помощью axios

"https://www.googleapis.com/fitness/v1/users/me/dataset:aggregate", - адрес отправки запроса

body, - тело запроса

axiosconfig - конфигурация axios

)

.then(res => { - результат успешного запроса

let bucket = res.data.bucket; - сохранение полученного ответа в переменную

С помощью map, полученный ответ переводится в удобный для обработки формат. Метод map создает новый массив, поэтому результат его работы сохраняется в переменную result

let result = bucket.map(el => {

let activity = {}; - создание пустого объекта

let date = new Date(Number(el.startTimeMillis)).toLocaleString("ru").split(",")[0]; - получение даты активности из строки формата: 01.01.1970, 00:00 - в date сохранится 01.01.1970

let start = new Date(Number(el.startTimeMillis)).toLocaleString("ru"); - получение даты с временем начала активности

let end = new Date(Number(el.endTimeMillis)).toLocaleString("ru"); - получение даты с временем конца активности

activity.date = date; - полученные данные записываются в объект

activity.start = start;

activity.end = end;

activity.intVal = el.dataset[0].point.length ? el.dataset[0].point[0].value[0].intVal : 0;

После того как объект был собран происходит запись в перменную result

return activity;

});

Затем полученный результат сохраняется в формат данных для диаграммы на главной странице приложения в userSteps

const userSteps = [{

name: "Шаги",

data: result.map(el => el.intVal)

}];

const userActivity = result; - полученный результат сохраняется в формат данных для таблицы на главной странице в userActivity, после этого создается переменная ответа

const response = {

userSteps: userSteps,

userActivity: userActivity

}

resolve(response); - результат выполнения функции возвращается с помощью resolve

})

.catch(err => {

reject(err); - в случае ошибки срабатывает reject, который вернет ошибку

});

});

};

Далее рассмотрим файл firebase.config.js (рисунок 29), в котором расположены конфигурации firebase. Полный код представлен в приложении Е.



Рисунок 29 – Внешний вид файла firebase.config.js

import firebase from "firebase/app"; - подключение библиотеки firebase

import "firebase/firestore"; - подключение хранилища firebase

Создание переменной конфигурации firebase. Данная конфигурация генерируется автоматически в базе данных

var firebaseConfig = {

apiKey: "AIzaSyBgcryG2NZhxBGPpdepY22y6a0ibkvaaeM",

authDomain: " fitness-app.firebaseapp.com",

databaseURL: "https://fitness-app-default-rtdb.europe-west1.firebasedatabase.app",

projectId: "fitness-app ",

storageBucket: "fitness-app.appspot.com",

messagingSenderId: "109229991916",

appId: "1:109229991916:web:d6ab9a361b95ac2a5842c9",

measurementId: "G-Y7KKVP7JEK",

clientId: "109229991916-ffvanr8j879nv8tmt9e32hvshenv27sk.apps.googleusercontent.com"

};

firebase.initializeApp(firebaseConfig); - Инициализация базы Firebase

const db = firebase.firestore(); - создание переменной для работы с хранилищем базы

export default db; - экспорт переменной

Далее рассмотрим файл auth.module.js (рисунок 30), в котором расположены модуль авторизации в хранилище приложения. Полный код представлен в приложении Е.

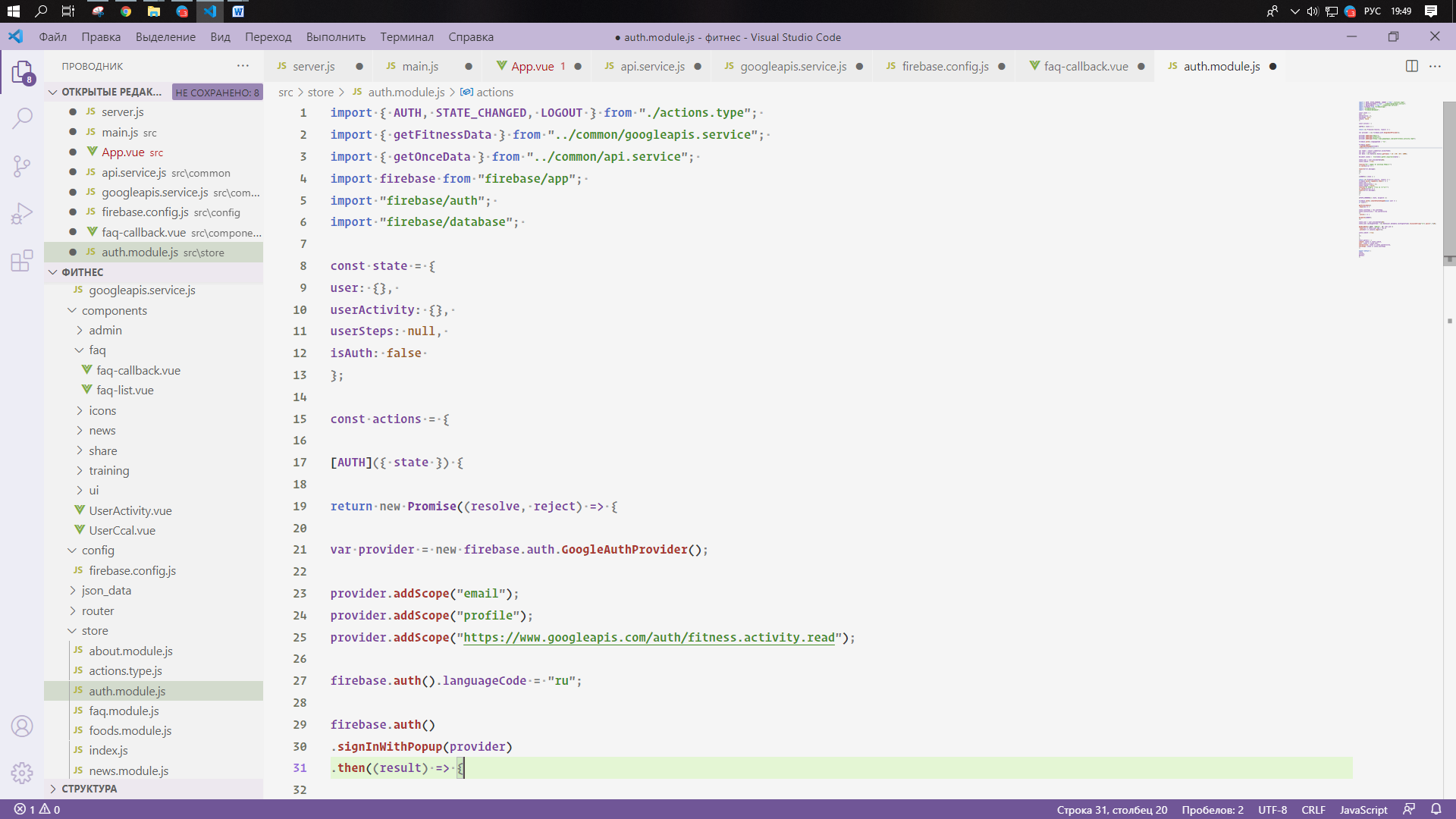


Рисунок 30 – Внешний вид файла auth.module.js

import { AUTH, STATE\_CHANGED, LOGOUT } from "./actions.type"; - подключение наименований асинхронных функций хранилища

import { getFitnessData } from "../common/googleapis.service"; - подключение функции запроса отчета по активности пользователя из google.service

import { getOnceData } from "../common/api.service"; - подключение метода запроса данных из базы

import firebase from "firebase/app"; - подключение firebase

import "firebase/auth"; - подключение библиотеки авторизации

import "firebase/database"; - подключение библиотеки базы данных

Переменная state необходима для хранения данных, которые будут доступны из любой точки приложения

const state = {

user: {}, - пользовательские данные

userActivity: {}, - активность пользователя

userSteps: null, - пройденные шаги пользователя

isAuth: false - состояние авторизации по умолчанию false (не авторизован)

};

const actions = {

[AUTH]({ state }) { - функция авторизации

Авторизация происходит асинхронным методом, поэтому данная функция возвращает Promise

return new Promise((resolve, reject) => {

А перемнную provider записывается результат выполнения метода GoogleAuthProvider, который выполняет подготовку запроса на авторизацию

var provider = new firebase.auth.GoogleAuthProvider();

Методом addScope добавляются данные, которые необходимо получить из аккаунта пользователя. К таким данным относятся: Email, вся информация профиля, и данные активности пользователя(только в режиме чтения)

provider.addScope("email");

provider.addScope("profile");

provider.addScope("https://www.googleapis.com/auth/fitness.activity.read");

languageCode задает язык, в котором будут возвращены данные

firebase.auth().languageCode = "ru";

Вызов всплывающего окна авторизации Google, методом signInWithPopup в который нужно передать запрос который был создан выше

firebase.auth()

.signInWithPopup(provider)

.then((result) => {

После успешного запроса токен пользователя сохраняется в переменную token. Данные пользователя сохраняются в переменную user.

В переменную date записывается метка времени с шагом на 24 часа вперед от текущего времени, данная переменная необходима для куки

var token = result.credential.accessToken;

var user = result.user;

var date = new Date(new Date().getTime() + 24 \* 60 \* 60 \* 1000);[14,654]

Затем токен необходимо записать в куки на устройстве пользователя, на 24 часа, по истечению 24 часов, токен будет удален

document.cookie = `t=${token};path=/;expires=${date}`;

state.user = user.providerData[0]; - запись данных пользователя в state

state.isAuth = true;

resolve("Авторизация произведена успешно!") - ответ об успешной авторизации

}).catch(error => {

reject(error.message); - ответ об ошибке

});

});

},

[LOGOUT]({ state }) { - функция выхода из приложения

Данная функция отправляет запрос в firebase о том, что пользователь вышел и сбрасывает все данные пользователя в хранилище приложения

return new Promise((resolve, reject) => {

firebase.auth().signOut().then(() => {

state.user = {};

state.userActivity = {};

state.isAuth = false;

resolve("Вы успешно вышли из системы!")

}).catch((error) => {

reject(error.message);

});

})

},

[STATE\_CHANGED]({ state, dispatch }){ - функция проверки состояния авторизации пользователя

firebase.auth().onAuthStateChanged(async user => { - отправка запроса на получение данных пользователя из базы

if (user) {

Если пользователь был получен, то происходит вызов метода запроса данных активности пользователя

getFitnessData()

.then(res => {

После успешного запроса все данные будут записаны в хранилище

state.userSteps = res.userSteps,

state.userActivity = res.userActivity

})

.catch(() => {

В случае ошибки пользователя выведет из системы

dispatch(LOGOUT);

});

Затем, когда данные активности получены, данные пользователя будут записаны в хранилище

state.user = user.providerData[0];

state.user.lastSignInTime = new Date(user.metadata.lastSignInTime).toLocaleString("ru").split(",")[0]; - форматирование времени последней авторизации в читаемый формат.

getOnceData({ path: "admins", id: user.uid }) - проверка: является ли пользователь администратором по его id

.then(res => state.user.adm = res.s) - в случае успеха пользователю присваивается статус администратора

.catch(err => console.log(err)); - в случае ошибки в консоль будет отображена ошибка

state.isAuth = true; - после получения всей информации о пользователе, состояние авторизации переключается в true(авторизован)

}

});

}

};

const getters = {

isAuth: state => state.isAuth, - запрос состояния авторизации

user: state => state.user, - данные пользователя

userActivity: state => state.userActivity, - данные активности пользователя

userSteps: state => state.userSteps - данные пройденных шагов

};

export default {

state,

actions,

getters

};

# 3 Руководство пользователю

При загрузке приложения отображается страница, на которой расположена кнопка авторизации в приложение (рисунок 31). Авторизация в прилождение происходит через Google аккаунт. При нажатии на кнопку «Авторизация через Google», откроется форма, в которой необходимо ввести электронную почту или номер телефона, а далее пароль от аккаунта (рисунок 32).

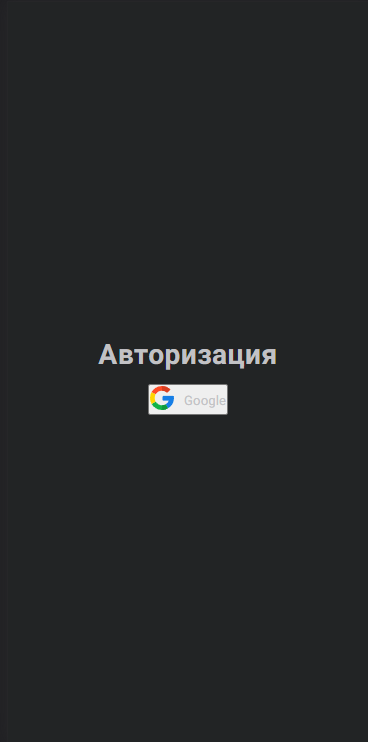


Рисунок 31 – Вход в приложение

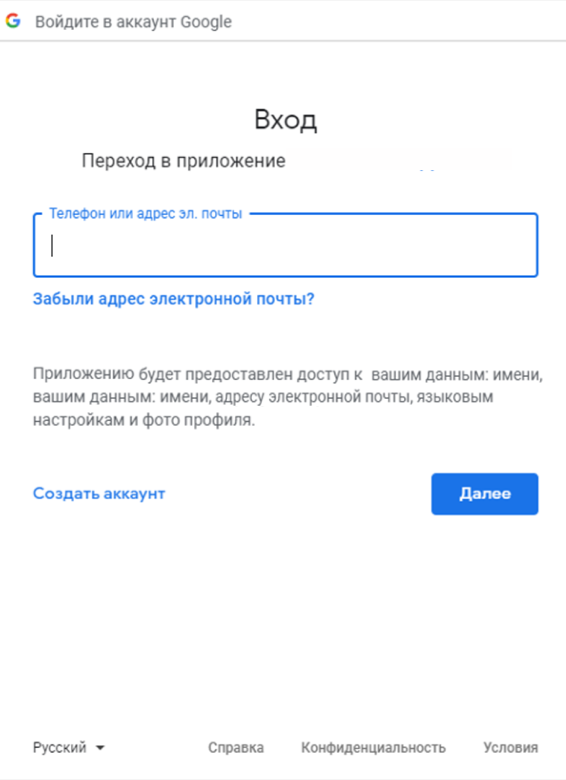


Рисунок 32 – Вход в приложение

После успешного входа перед пользователем отображается баннер, статистика потребления калорий за день, активность и тренировки (рисунки 33 - 35).

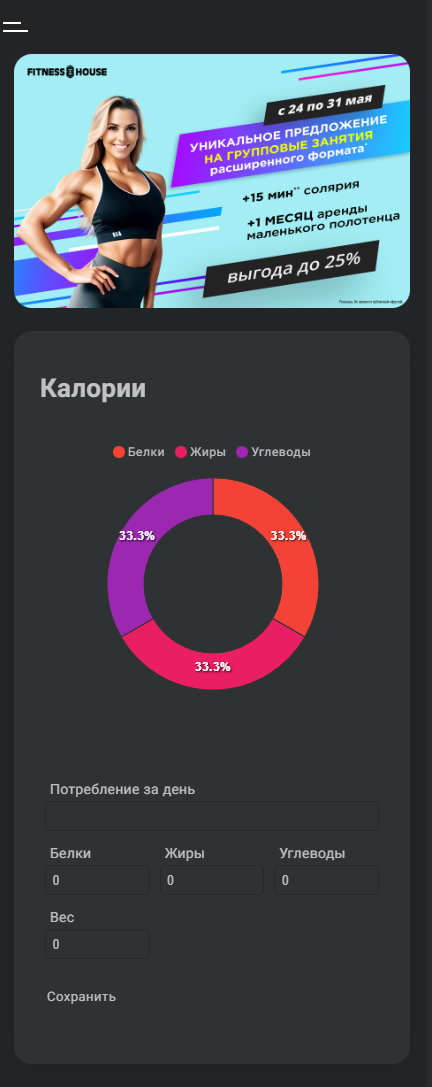


Рисунок 33 – Главное окно приложения

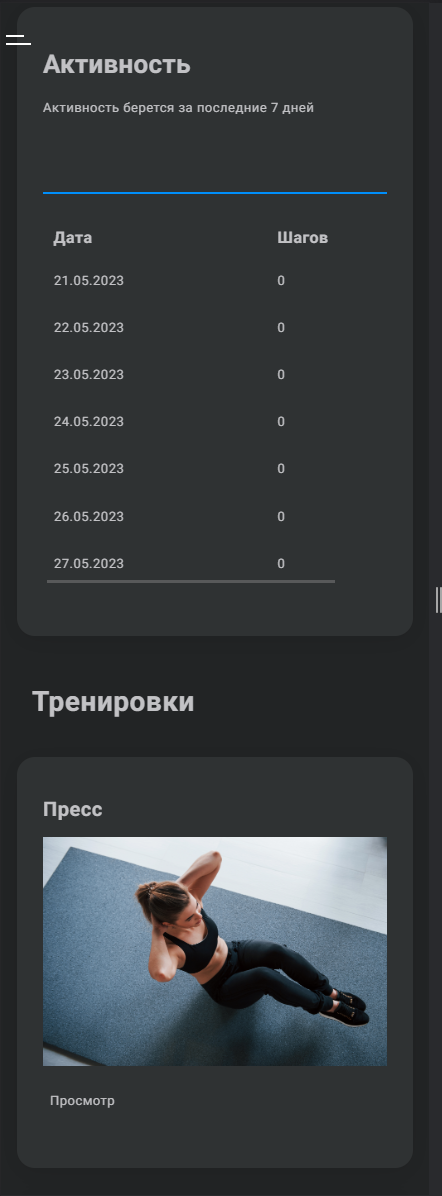


Рисунок 34 – Главное окно приложения

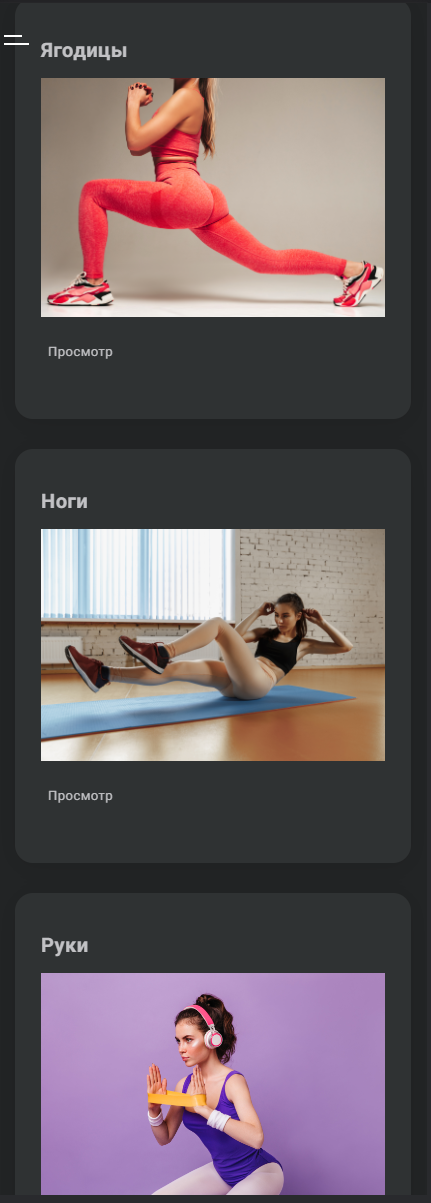


Рисунок 35 – Главное окно приложения

Для того чтобы добавить потребляемый продукт, необходимо ввести его название в поле «Потребление за день», если продукт есть в базе, то ниже отобразятся варианты (рисунок 36). При выборе продукта, автоматически заполнится информация о его пищевой ценности (рисунок 37). Для добавления продукта в диаграмму, необходимо нажать на кнопку «Сохранить». Пункты автоматически очистятся, а в диаграмме поменяются показатели (рисунок 38).

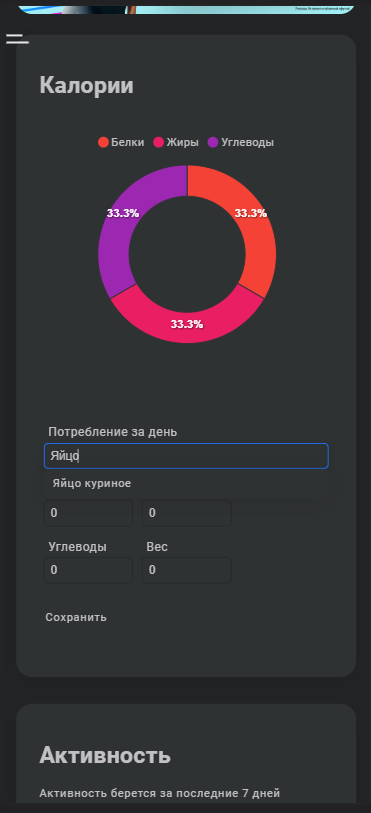


Рисунок 36 – Внешний вид примера заполнения продукта

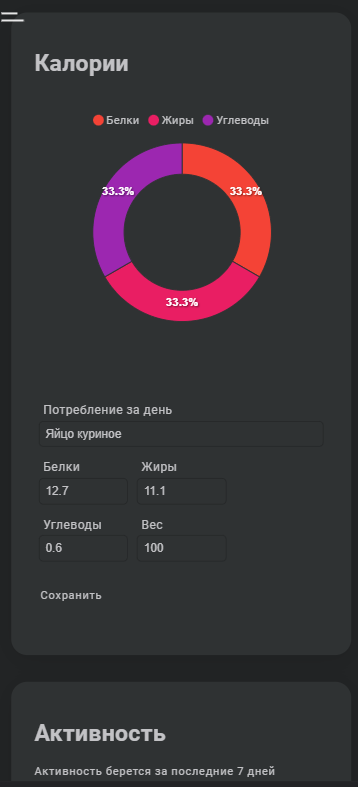


Рисунок 37 – Внешний вид автоматического заполнения энергетической ценности продукта

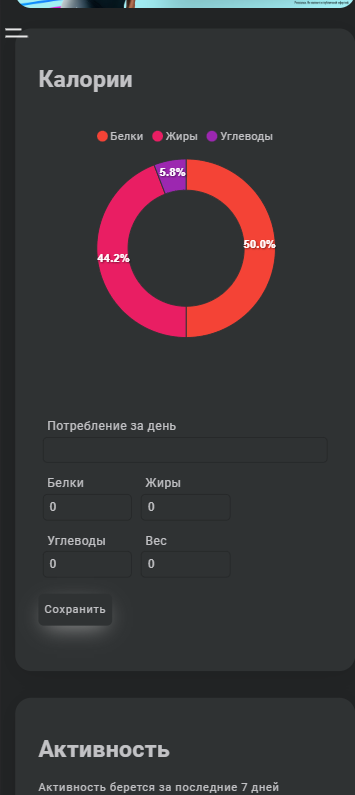


Рисунок 38 – Добавление продукта в диаграмму

При переходе на страницу приложения расписание, перед пользователем отображается расписание тренировок в фитнес клубе (рисунок 39).



Рисунок 39– Внешний вид страницы расписание

При открытии тренировок, перед пользователем отображаются тренировки на различные части тела, чтобы просмотреть комплект тренировок, необходимо нажать на кнопку «Просмотр» (рисунок 40). После нажатия на кнопку перед пользователем отобразятся упражнения, после выполнения тренировок, необходимо поставить галочку рядом с каждым упражнением, это будет означать, что приложение успешно выполнено (рисунок 41)

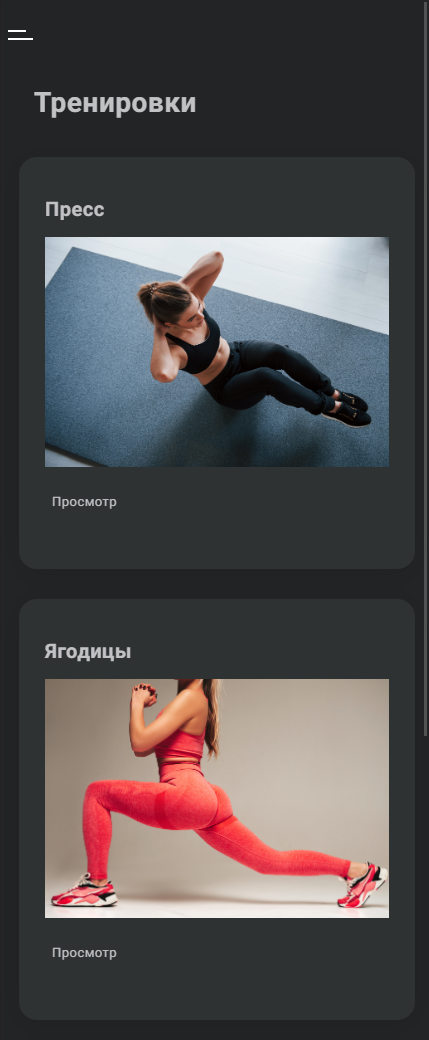


Рисунок 40 – Внешний вид страницы «Тренировки»

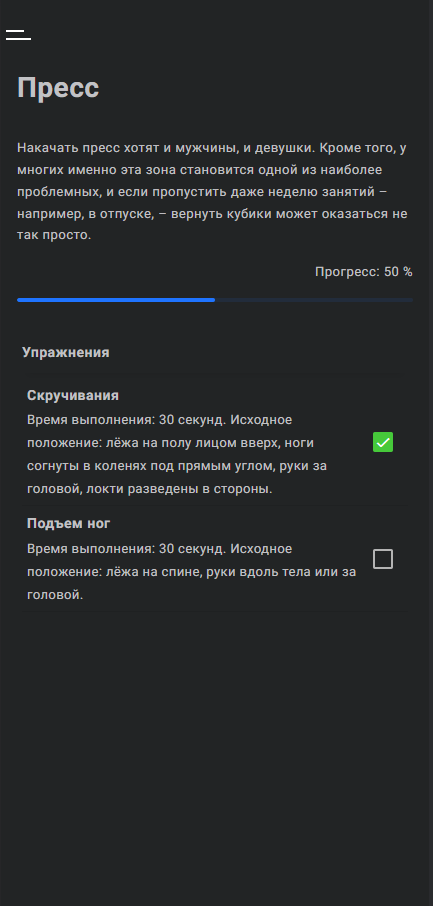


Рисунок 41 – Внешний вид страницы тренировки на пресс

В разделе акции перед пользователем отображаются акции проходимые в фитнес клубе (рисунок 42).

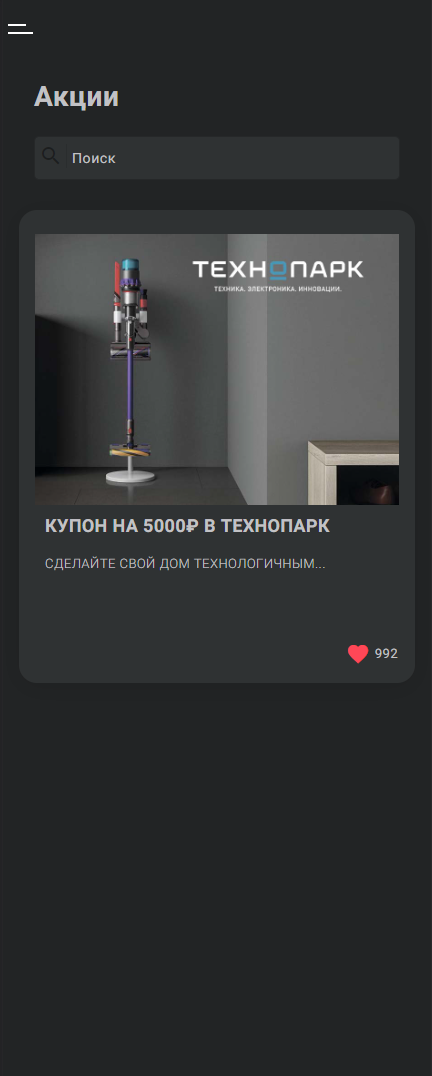


Рисунок 42 – Внешний вид страницы акции

В настройках (рисунок 43) пользователь заполняет свои параметры и следит за прогрессом (рисунок 44).

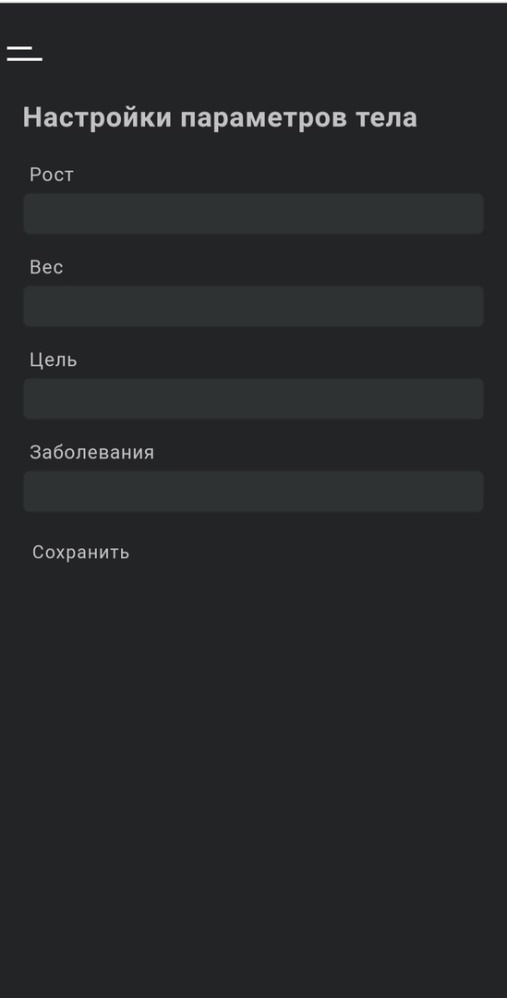


Рисунок 43 – Внешний вид страницы «Настройки»

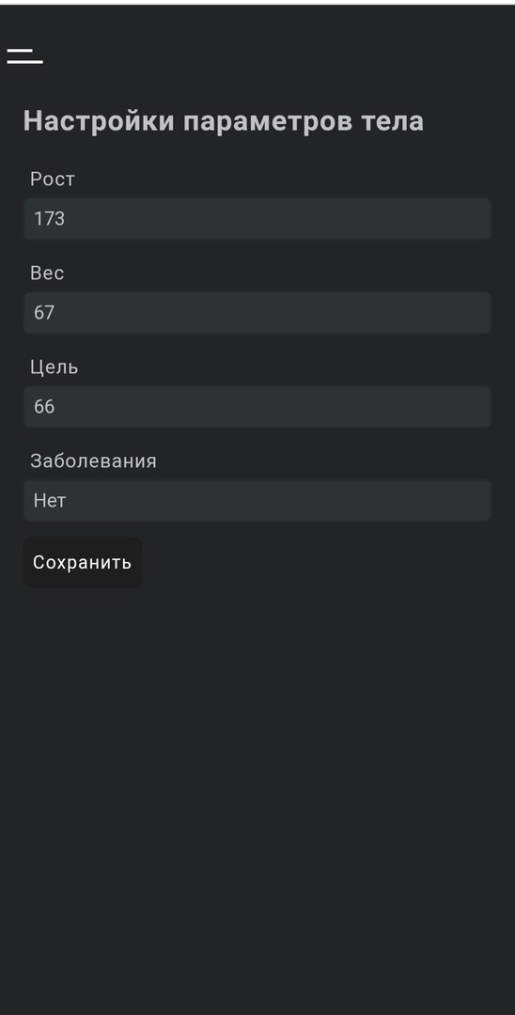


Рисунок 44 – Заполнение параметров тела

# 4 Экономическая часть

## 4.1 Расчет трудоемкости разработки программного продукта

В рамках этого этапа проводится расчет затрат времени на выполнение работ по созданию программного продукта. Все работы, требующиеся для разработки программы, можно разделить на 2 группы:

Составление необходимого перечня работ и кодирование

В разработку программного продукта входит следующий перечень работ:

1. Постановка проблемы;
2. Описание предметной области;
3. Программирование и реализация Web-страниц, БД и т.п.;
4. Отладка приложения;
5. Документирование.

Установка оценок времени продолжительности работ

Для расчета ожидаемого времени выполнения работ возьмем значения времени выполнения при благоприятных условиях и при неблагоприятных условиях и произведем вычисления, взяв среднее арифметическое от оптимистичной и пессимистичной оценок.

Таблица 2 – Расчет трудоемкости внедрения программного продукта

|  |  |  |  |
| --- | --- | --- | --- |
| Наименование работ | Минимальное кол-во дней  Тmin | Максимальное кол-во дней  Тmax | Ожидаемое  кол-во дней  Тож |
| Постановка проблемы | 1 | 2 | 1,5 |
| Описание предметной области | 2 | 4 | 3 |
| Проектирование и дизайн программного продукта | 18 | 19 | 18,5 |
| Отладка Web-приложения, БД и т.п. | 3 | 5 | 4 |
| Оформление документов | 3 | 5 | 4 |
| Внедрение Web-приложения | 1 | 2 | 1,5 |
| Общая продолжительность работы составляет: | 28 | 37 | 32,5 |

Тож = (4.1)

2

Тmin+ Tmax

где Тож – ожидаемое время продолжительности работ

Тmin – оценка при наиболее благоприятных условиях

Tmax – оценка при наиболее неблагоприятных условиях

2

28+37

Тож = = 32,5

Ожидаемое время выполнения работ по разработке программного продукта = 32,5 дней (или 1 месяц и 2,5 дня).

## 4.2 Расчет себестоимости и цены разработки программного продукта

Себестоимость разработки информационной системы - это, как правило, совокупность затрат на разработку программного продукта. Затраты на разработку программы подразделяются на следующие статьи расходов:

1. Материальные затраты;
2. Основная заработная плата;
3. Начисления на оплату труда;
4. Накладные расходы.

## 4.3 Методика расчета статей калькуляции

### 4.3.1 Расчет материальных затрат

В статье «Материальные затраты» предусмотрены примерные затраты на материалы, применяемые при использовании данного программного продукта на предприятии.

Расчет стоимости материальных затрат произведен в таблице 3.

Таблица 3 – Расчет стоимости материальных затрат

|  |  |  |
| --- | --- | --- |
| Наименование материала | Количество  комплектов,  шт. | Цена  руб. |
| USB-флеш-накопитель | 1 | 800,0 |

Продолжение таблицы 3

|  |  |  |
| --- | --- | --- |
| Наименование материала | Количество  комплектов,  шт. | Цена  руб. |
| Бумага | 1 | 400,0 |
| Картридж для принтера | 1 | 3500,0 |
| Ручка | 1 | 50,0 |
| Всего |  | 4750,0 |

Расчет заработной платы программиста

Расчет ЗП программиста производится в соответствии с трудоемкостью разработки программного продукта.

Данные для расчета:

1. Оклад – 33000,0 руб.;
2. Плановый фонд рабочего времени за месяц – 176 часов (22 дня);
3. Стимулирующая надбавка – 50%;
4. Тарифная ставка.

Часовая тарифная ставка (Сч) определяется:

Фрв

Оклад

Сч = (4.3)

где Фрв – плановый фонд рабочего времени за месяц, из расчета 22 рабочих дня по 8 часов.

Сч = = 187,5 руб. в час (4.4)

176

33000,0

Основная заработная плата программиста за разработку программы составит:

ЗПосн = Сч ∙ Тож час (4.5)

ЗПосн = 187,5∙ (32,5 ∙ 8) = 48750 руб.; (4.6)

ЗПдоп = ЗПосн ∙ 0,5 (4.7)

ЗПдоп = 48750 ∙ 0,5 = 24375 руб. (4.8)

Затраты на оплату труда:

ЗПобщ = ЗПосн  + ЗПдоп (4.9)

ЗПобщ = 48750+24375 = 73125 руб. (4.10)

Страховые взносы с заработной платы – 30%:

100

ЗПобщ∙30,0

Стр.взн. = , (4.11)

73125. 30,0

Стр. взн. = 100 = 21937,5 руб. (4.12)

Расчет накладных расходов

В статью включены затраты, связанные с обслуживанием и организацией производства. В данном случае – это расходы на электроэнергию, потребляемую компьютером за время разработки программы и амортизационные отчисления.

Таблица 4 – Затраты на электроэнергию

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Вид оборудования | Мощность, кВт/час | Стоимость, кВт/час | Время работы оборудования Тож (час) | Сумма затрат, руб. |
| ПК | 0,4 | 4,63 | 240 | 444,50 |
| Итого | - | - | - | 444,50 |

В таблице 4 выполнены расчеты по затрате ресурсов на электроэнергию по формуле:

Сумма затрат = (М ∙ С) ∙ Т (4.13)

где М - Мощность, кВт

С - Стоимость 1 кВт/час

Т - Время работы оборудования, Тож час

В таблице 5 рассчитана сумма амортизационных отчислений за период разработки программного продукта.

Таблица 5 – Амортизационные отчисления

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Вид оборудования | Первоначальная стоимость, руб. | Количество рабочих месяцев | Норма амортизации % | Сумма амортизационных  отчислений, руб. |
| ПК | 80000 | 1 | 12,5 | 833,30 |
| Итого | - | - | - | 833,30 |

Сумма амортизационных отчислений за период разработки, определяются по следующим формулам:

100

Сп∙ На

Аг = (4.14)

Тмес

∙

12

Аг

Амес = (4.15)

где Сп – первоначальная стоимость оборудования, рублей;

На – годовая норма амортизации;

Тмес – трудоемкость разработки программы, месяцы;

12 – количество календарных месяцев в году;

Сумма накладных расходов = затраты на электроэнергию + амортизационные отчисления.

Сумма накладных расходов = 444,50 + 833,30 = 1277,80 руб.

Расчет калькуляции затрат на разработку программного продукта

Расчет калькуляции затрат наглядно представлен в таблице 6 «Калькуляция затрат на разработку программного продукта» и на рисунке 45.

Таблица 6 - Калькуляция затрат на разработку программного продукта

|  |  |
| --- | --- |
| Статья затрат | Сумма затрат, руб. |
| Материальные затраты | 4750,0 |
| Общая сумма з/п | 73125,0 |
| Отчисления на страховые взносы | 21937,50 |
| Накладные расходы | 1277,80 |
| Итого: | 101090,30 |

Рисунок 45 - Затраты на разработку программного продукта

### 4.3.2 Определение цены программного продукта

Процент рентабельности (условно) – 18%

Прибыль рассчитывается по формуле:

100

Сполн ∙%Р

П = (4.16)

где Сполн – себестоимость

Р - процент рентабельности

П = = 18196,25 руб.

100

101090,30∙18

Цена программного продукта равна сумме полной себестоимости и прибыли:

Ц = С*полн* + П (4.17)

Ц =101090,30 + 18196,25 = 119286,55 руб.

Цена программного продукта с НДС (ставка 20%):

НДС = (4.18)

100

Ц ∙20

НДС = = 23857,31 (4.19)

100

(119286,55∙20)

Цена с НДС = 119286,55 + 23857,31= 143143,86 руб. (4.20)

Предполагаемая стоимость программного продукта составляет 143143,86 рубля.

Вывод

Для создания мобильного приложения для фитнес-клуба необходимы такие специалисты, как: дизайнеры, разработчики, менеджеры, которым необходимо выплачивать заработную плату.

Я считаю, что данный продукт конкурентоспособен на рынке приложений и сможет быстро окупиться.

# ЗАКЛЮЧЕНИЕ

В заключение дипломной работы по разработке мобильного приложения для фитнес-клуба можно подчеркнуть следующие аспекты:

Результаты и достижения: в ходе работы было разработано и успешно реализовано мобильное приложение для фитнес-клуба. Приложение предоставляет удобный и интуитивно понятный интерфейс, который позволяет пользователям получить максимальную пользу и удовольствие от использования приложения.

Основные функциональные возможности: В приложении реализованы ключевые функциональности, такие как расписание тренировок, планировщик питания, возможность отслеживать свой прогресс и вводить параметры тела, тренировки на различные группы мышц. Эти возможности позволяют пользователям максимально эффективно использовать услуги фитнес-клуба и улучшать свою физическую форму.

Преимущества приложения: Мобильное приложение для фитнес-клуба предоставляет ряд значительных преимуществ, таких как удобство использования, доступность 24/7, персонализация тренировок и программ, удобная система для отслеживания потребляемых продуктов и многое другое. Это позволяет фитнес-клубу привлекать больше клиентов, повышать уровень обслуживания и улучшать их фитнес-опыт.

В целом, разработка и реализация мобильного приложения для фитнес-клуба является важным шагом в современной цифровой трансформации отрасли фитнеса. Приложение позволяет фитнес-клубу предоставлять удобные и инновационные услуги своим клиентам, улучшать их фитнес-опыт и укреплять свою позицию на рынке.

В заключение можно утверждать, что поставленная цель выпускного квалификационного проекта достигнута, а основные задачи выполнены.

В дальнейшем приложение будет дорабатываться, в приложение планируется добавить онлайн сопровождение тренера и консультации.

# СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Бенкен Е.С. PHP, MySQL, XML: программирование для Интернета. – С. Пб.:BHV, 2015. –570 с.
2. Веллинг, Л. Разработка веб-приложений с помощью PHP и MySQL: учебник. - Мск.: Вильямс, 2014. – 848 с.
3. Гаевский, А.Ю. 100% самоучитель. Создание Web-страниц и Web-сайтов. HTML и JavaScript / А.Ю. Гаевский, В.А. Романовский. - М.: Триумф, 2015. - 464 c.
4. Гудман, Д. JavaScript и DHTML. Сборник рецептов. Для профессионалов / Д. Гудман. - М.: Питер, 2015. - 523 c.
5. Дакетт, Джон Основы веб-программирования с использованием HTML, XHTML и CSS / Джон Дакетт. - М.: Эксмо, 2017. - 768 c.
6. Избачков, Ю.С. Информационные системы / Ю.С. Избачков, В.И. Петров, – СПб.: Питер, 2018 – 471с.
7. Кузнецов, М. Самоучитель PHP: учебное пособие. – СПб.: БХВ-Петербург, 2016.-672 с.;
8. Купер А. Рейман Р., Кронин Д. Алан Купер об интерфейсе. Основы проектирования взаимодействия. - Пер. с англ. - СПб.: Символ-Плюс, 2018. – 688 с.
9. Макконнелл С. Совершенный код. Мастер – класс / Пер. с англ. — СПб.: БХВ, 2018. — 896 с.;
10. Макфарланд Д. Новая большая книга CSS. — СПб.: Питер, 2019. — 720 с.;
11. Прохоренок, Н. HTML, JavaScript, PHP и MySQL. Джентельменский набор Web-мастера: учебник. - СПб.: БХВ, 2017. – 912 с.
12. Фаулер М. Рефакторинг кода на JavaScript: улучшение проекта существующего кода. — СПб.: ООО “Диалектика”, 2019. — 464 с.
13. Фримен Э., Робсон Э. Изучаем программирование на JavaScript. – СПб: Питер, 2017. – 640 с.;
14. Фреймворк React.js [Электронный ресурс] /-.Режим доступа: https://habr.com/ru/company/ruvds/blog/428077/, свободный
15. Фреймворк Vue.js. [Электронный ресурс] /-.Режим доступа: https://jetruby.com/ru/blog/vue-js-preimuschestva-i-nedostatki/, свободный

# ПРИЛОЖЕНИЕ А

Server.js – файл, в котором расположена серверная часть приложения

const express = require("express");

const path = require("path");

const history = require("connect-history-api-fallback");

var bodyParser = require("body-parser");

const nodemailer = require("nodemailer");

const app = express();

const staticFileMiddleware = express.static(path.join(\_\_dirname + "/dist"));

app.use(staticFileMiddleware);

app.use(history({

disableDotRule: true,

verbose: true

}));

app.use(staticFileMiddleware);

app.use(bodyParser.urlencoded({ extended: true }));

app.get("/", (req, res) => {

res.render(path.join(\_\_dirname + "/dist/index.html"));

});

app.post("/send", async (req, res) => {

const msg = JSON.parse(Object.keys(req.body));

let transporter = nodemailer.createTransport({

host: "smtp.gmail.com",

port: 465,

secure: true,

auth: {

user: "no-replay@fitness-app.com",

pass: "qwerty123fitness-app ",

},

});

let info = await transporter.sendMail({

from: `"${msg.username} 👻" <${msg.email}>`,

to: "diana07.12.00@gmail.com",

subject: msg.title,

text: msg.description,

html: `

<h1>${msg.title}</h1>

<p>${msg.description}</p>

<p>${msg.question}</p>

<p style="margin-top: 100px">

Отправил: "${msg.username} 👻" <${msg.email}>

</p>

`,

});

console.log(info);

res.send({message:"Ваше сообщение отправлено!"});

});

var server = app.listen(process.env.PORT || 3000, () => {

var port = server.address().port;

console.log("App now running on port", port);

});

# ПРИЛОЖЕНИЕ Б

Main.js – файл, в котором расположено глобальное подключение компонентов необходимых для работы приложения

import Vue from "vue";

import App from "./App.vue";

import router from "./router";

import store from "./store";

import Vuesax from "vuesax";

import "vuesax/dist/vuesax.css";

import "material-icons/iconfont/material-icons.css";

import "./assets/style/main.scss";

import "./config/firebase.config";

import VueApexCharts from "vue-apexcharts";

const icons = require.context('@/components/icons', false, /icon-\w+(-)?\w+\.(vue)$/)

icons.keys().forEach(iconFile => {

const componentConfig = icons(iconFile)

const componentName = iconFile

.replace('.vue', '')

.replace('./', '')

.replace(/([a-z])([A-Z])/g, '$1 $2')

.split(' ')

.map(word => word.toLowerCase())

.join('-')

Vue.component(componentName, componentConfig.default || componentConfig)

})

Vue.component("apexchart", VueApexCharts)

Vue.use(Vuesax);

Vue.config.productionTip = false

new Vue({

router,

store,

render: h => h(App),

created(){

store.dispatch("stateChanged");

}

}).$mount("#app");

# ПРИЛОЖЕНИЕ В

App.vue – код, в котором расположены базовые компоненты приложения

<template>

<div id="app">

<c-menu />

<div class="page">

<transition name="fade">

<div class="hello" v-if="!isAuth">

<vs-card class="hello\_\_block">

<h1 class="hello\_\_title">{{hello}}</h1>

<p class="hello\_\_text">Чтобы начать пользоваться нашим приложением. Пожалуйста, авторизуйтесь.</p>

<vs-button type="border" class="hello\_\_button" size="small" @click="authGoogle()">

<vs-row vs-align="center">

<icon-base

component="icon-google"

class="hello\_\_icon"

:width="24"

:height="24"

/>

Авторизация через Google

</vs-row>

</vs-button>

</vs-card>

</div>

</transition>

<transition name="jump">

<router-view class="page\_\_block" />

</transition>

</div>

</div>

</template>

<script>

import { mapGetters } from 'vuex';

import menu from './components/ui/menu'

export default {

components: {

'c-menu': menu /

},

created() {

this.getHours();

},

data() {

return {

hours: 0

}

},

computed: {

...mapGetters(["isAuth"]),

hello() {

let say;

if (this.hours > 7 && this.hours < 12)

say = "Доброе утро!";

if (this.hours > 12 && this.hours < 18)

say = "Добрый день!";

if (this.hours >= 18 && this.hours <= 23)

say = "Добрый вечер!";

if (this.hours >= 0 && this.hours < 6)

say = "Доброй ночи!";

return say;

}

},

methods: {

getHours() {

const time = new Date().getHours();

this.hours = time;

},

authGoogle() {

this.$store.dispatch("auth")

.then(msg => this.$vs.notify({title: "",text: msg, color: "success",position: "top-right"}))

.catch(err => this.$vs.notify({title: "",text: err, color: "danger",position: "top-right"}));

},

}

};

</script>

<style lang="scss">

@import './assets/style/varibles.scss';

.hello {

position: absolute;

top: 0;

left: 0;

width: 100%;

height: 100vh;

display: flex;

justify-content: center;

align-items: center;

background: #1f74ff;

background: linear-gradient(135deg, #d083f7 0%, #1a2eff 50%, #00d4ff 100%);

z-index: 9999;

&\_\_block {

display: flex;

justify-content: center;

align-items: center;

flex-direction: column;

text-align: center;

max-width: 420px;

margin: 5px;

}

&\_\_title {

margin-bottom: 5px;

}

&\_\_text {

margin-bottom: 20px;

}

&\_\_icon {

margin-right: 10px;

}

}

.fade-enter-active, .fade-leave-active {

transition: all 1s .6s;

}

.fade-leave-active .hello\_\_block {

transition: all 0.3s;

opacity: 0;

}

.fade-enter, .fade-leave-to {

opacity: 0;

}

.jump-leave-active {

transition: all 1s;

opacity: 0;

}

.jump-enter-active {

position: absolute;

top: 0;

left: 0;

z-index: 9999;

transition: all 1s;

}

.jump-enter-active \* {

opacity: 0;

}

.jump-enter-active::before {

content: '';

position: absolute;

top: 0;

left: 0;

width: 0%;

height: 100%;

opacity: 1;

background: #1f74ff;

background: linear-gradient(135deg, #d083f7 0%, #1a2eff 50%, #00d4ff 100%);

animation: 1s ease-in forwards leaveAnim;

}

@keyframes leaveAnim {

0% {

width: 0;

}

25% {

width: 100vw;

}

75% {

width: 100vw;

}

100% {

left: auto;

right: 0;

width: 0;

}

}

.jump-enter, .jump-leave-to /\* .jump-leave-active below version 2.1.8 \*/ {

opacity: 1;

}

</style>

# ПРИЛОЖЕНИЕ Г

Api.service.js – файл, в котором расположено взаимодействие с API

import firebase from "firebase/app";

import "firebase/database";

import { ErrorTest } from "./errors.service";

export const addRealTimeData = ({ path, data }) => {

return new Promise((resolve, reject) => {

const key = firebase.database().ref().child(path).push().key;

let setData = {};

data.id = key;

setData[`/${path}/${key}`] = data;

firebase.database().ref().update(setData)

.then(msg => resolve(msg))

.catch(err => reject(err));

});

};

export const getRealtimeData = ({ path, id = "" }) => {

return new Promise((resolve, reject) => {

const tests = [

{

name: "path",

type: "string",

check: path

}

]

const errorTest = new ErrorTest(tests);

const stateErr = errorTest.checkTypeData();

const found = errorTest.checkNotFound();

let posts = "Записей нет";

if (stateErr.success) {

alert(stateErr.message);

reject(stateErr);

return;

}

if (found.success) {

alert(found.message);

reject(found);

return;

}

firebase.database().ref(`/${path}/${id}`).on('value', snapshot => {

posts = (snapshot.val() && snapshot.val()) || 'Записей нет'

if (typeof posts !== "string") {

resolve(posts);

} else {

reject(posts);

}

});

});

};

export const getOnceData = ({ path, id = "" }) => {

return new Promise((resolve, reject) => {

const tests = [

{

name: "path",

type: "string",

check: path

}

]

const errorTest = new ErrorTest(tests);

const found = errorTest.checkNotFound();

if (found.success) {

alert(found.message);

reject(found);

return;

}

const stateErr = errorTest.checkTypeData();

if (stateErr.success) {

alert(stateErr.message);

reject(stateErr);

return;

}

firebase.database().ref(`/${path}/${id}`).once('value')

.then(snapshot => {

var posts = (snapshot.val() && snapshot.val()) || 'Новостей нет';

if (typeof posts !== "string") {

resolve(posts);

}

})

.catch(err => {

reject(err);

});

});

}

export const removeData = ({ path, id = "" }) => {

return new Promise((resolve, reject) => {

firebase.database().ref(`/${path}/${id}`).remove()

.then(res => {

resolve(res);

})

.catch(err => {

reject(err);

console.log(err);

});

});

}

# ПРИЛОЖЕНИЕ Д

Googleapis.service.js – файл, в котором расположено взаимодействие с Google API

import axios from "axios";

import cookie from "cookie";

export const getFitnessData = () => {

const token = cookie.parse(document.cookie).t;

const body = {

"aggregateBy": [{

"dataTypeName": "com.google.step\_count.delta",

"dataSourceId": "derived:com.google.step\_count.delta:com.google.android.gms:estimated\_steps"

}],

"bucketByTime": { "durationMillis": 86400000 },

"startTimeMillis": new Date().getTime() - 7 \* 24 \* 60 \* 60 \* 1000,

"endTimeMillis": new Date().getTime() // + 24 \* 60 \* 60 \* 1000

}

const axiosconfig = {

headers: {

"Authorization": `Bearer ${token}`,

"Content-Type": "application/json;encoding=utf-8"

}

};

return new Promise((resolve, reject) => {

axios

.post(

"https://www.googleapis.com/fitness/v1/users/me/dataset:aggregate",

body,

axiosconfig

)

.then(res => {

let bucket = res.data.bucket;

let result = bucket.map(el => {

let activity = {};

let date = new Date(Number(el.startTimeMillis)).toLocaleString("ru").split(",")[0];

let start = new Date(Number(el.startTimeMillis)).toLocaleString("ru");

let end = new Date(Number(el.endTimeMillis)).toLocaleString("ru");

activity.date = date;

activity.start = start;

activity.end = end;

activity.intVal = el.dataset[0].point.length ? el.dataset[0].point[0].value[0].intVal : 0;

return activity;

});

const userSteps = [{

name: "Шаги",

data: result.map(el => el.intVal)

}];

const userActivity = result;

const response = {

userSteps: userSteps,

userActivity: userActivity

}

resolve(response);

})

.catch(err => {

reject(err);

});

});

};

# ПРИЛОЖЕНИЕ Е

Firebase.config.js – файл, в котором расположены конфигурации firebase

import firebase from "firebase/app";

import "firebase/firestore";

var firebaseConfig = {

apiKey: "AIzaSyBgcryG2NZhxBGPpdepY22y6a0ibkvaaeM",

authDomain: "fitness-app.firebaseapp.com",

databaseURL: "https://fitness-app-default-rtdb.europe-west1.firebasedatabase.app",

projectId: "fitness-app",

storageBucket: "fitness-app.appspot.com",

messagingSenderId: "109229991916",

appId: "1:109229991916:web:d6ab9a361b95ac2a5842c9",

measurementId: "G-Y7KKVP7JEK",

clientId: "109229991916-ffvanr8j879nv8tmt9e32hvshenv27sk.apps.googleusercontent.com"

};

firebase.initializeApp(firebaseConfig);

const db = firebase.firestore();

export default db;

# ПРИЛОЖЕНИЕ Ж

Auth.module.js – файл, в котором расположены модуль авторизации в хранилище приложения

import { AUTH, STATE\_CHANGED, LOGOUT } from "./actions.type";

import { getFitnessData } from "../common/googleapis.service";

import { getOnceData } from "../common/api.service";

import firebase from "firebase/app";

import "firebase/auth";

import "firebase/database";

const state = {

user: {},

userActivity: {},

userSteps: null,

isAuth: false

};

const actions = {

[AUTH]({ state }) {

return new Promise((resolve, reject) => {

var provider = new firebase.auth.GoogleAuthProvider();

provider.addScope("email");

provider.addScope("profile");

provider.addScope("https://www.googleapis.com/auth/fitness.activity.read");

firebase.auth().languageCode = "ru";

firebase.auth()

.signInWithPopup(provider)

.then((result) => {

var token = result.credential.accessToken;

var user = result.user;

var date = new Date(new Date().getTime() + 24 \* 60 \* 60 \* 1000);

document.cookie = `t=${token};path=/;expires=${date}`;

state.user = user.providerData[0];

state.isAuth = true;

resolve("Авторизация произведена успешно!")

}).catch(error => {

reject(error.message);

});

});

},

[LOGOUT]({ state }) {

return new Promise((resolve, reject) => {

firebase.auth().signOut().then(() => {

state.user = {};

state.userActivity = {};

state.isAuth = false;

resolve("Вы успешно вышли из системы!")

}).catch((error) => {

reject(error.message);

});

})

},

[STATE\_CHANGED]({ state, dispatch }){

firebase.auth().onAuthStateChanged(async user => {

if (user) {

getFitnessData()

.then(res => {

state.userSteps = res.userSteps,

state.userActivity = res.userActivity

})

.catch(() => {

dispatch(LOGOUT);

});

state.user = user.providerData[0];

state.user.lastSignInTime = new Date(user.metadata.lastSignInTime).toLocaleString("ru").split(",")[0];

getOnceData({ path: "admins", id: user.uid })

.then(res => state.user.adm = res.s)

.catch(err => console.log(err));

state.isAuth = true;

}

});

}

};

const getters = {

isAuth: state => state.isAuth,

user: state => state.user,

userActivity: state => state.userActivity,

userSteps: state => state.userSteps

};

export default {

state,

actions,

getters

};