Министерство образования Республики Беларусь

Учреждение образования

«Белорусский государственный университет

информатики и радиоэлектроники»

|  |  |
| --- | --- |
|  | Номер зачетной книжки 15350133 |
|  | Практика зачтена с оценкой  *\_\_\_\_\_* (*\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_*)  (цифрой) (прописью) |
|  | *\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_*  (подпись руководителя практики от БГУИР)  *\_\_\_\_\_\_\_\_\_\_\_\_\_*.2025 |

**ОТЧЕТ**

**по практике**

Место прохождения практики: ООО «УайтСнейк», г.Минск

Сроки прохождения практики: с 10.02.2025 по 23.03.2025

|  |  |  |
| --- | --- | --- |
| Руководитель практики от предприятия:  *\_\_\_\_\_\_\_\_\_\_\_\_\_*А.А.Зубрицкий  (подпись руководителя)  М.П. |  | Студент группы 153504  *\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_* И.А.Тиханёнок  (подпись студента)  Руководитель практики от БГУИР  Давыдчик А.В. – ассистент кафедры информатики |

Минск 2025

**СОДЕРЖАНИЕ**

[Введение 3](#_Toc194576975)

[1 Анализ аналогов, литературных источников и формирование требований к проектируемому программному средству 5](#_Toc194576976)

[1.1 Аналитический обзор литературных источников 5](#_Toc194576977)

[1.2 Обзор и анализ существующих аналогов 11](#_Toc194576978)

[1.3 Требования к проектируемому программному средству 13](#_Toc194576979)

[2 Анализ требований к программному средству и разработка функциональных требований 19](#_Toc194576980)

[2.1 Функциональная модель программного средства 19](#_Toc194576981)

[2.2 Разработка спецификаций функциональных требований 22](#_Toc194576982)

[3 Технико-экономическое обоснование разработки и реализации на рынке игрового веб-приложения для моделирования стартапов 25](#_Toc194576983)

[3.2 Расчет инвестиций в разработку программного средства для его реализации на рынке 26](#_Toc194576984)

[3.3 Расчет экономического эффекта от реализации программного средства на рынке 28](#_Toc194576985)

[3.4 Расчет показателей экономической эффективности разработки и реализации программного средства на рынке 29](#_Toc194576986)

[Заключение 31](#_Toc194576987)

[Список использованных источников 32](#_Toc194576988)

ВВЕДЕНИЕ

Безусловно, можно с уверенностью утверждать, что современные веб-технологии могут значительно улучшить процесс моделирования и управления бизнес-процессами. В наше время веб-приложения становятся неотъемлемой частью жизни, и использование их для моделирования бизнес-процессов — это естественный шаг в эволюции инструментов для стартапов и предпринимателей. В частности, разработки, ориентированные на моделирование бизнес-процессов, могут значительно упростить процессы планирования и масштабирования для начинающих предпринимателей, обеспечивая им необходимые инструменты для построения и управления бизнесом в реальном времени.

В качестве области применения выбран инструмент для моделирования бизнес-процессов, ориентированный на стартапы, с акцентом на две ключевые области: физический бизнес, такой как открытие заводов и фабрик, и инвестиционный бизнес, включающий криптовалюты, ставки и другие инвестиционные активы. Этот инструмент не просто отображает графики и диаграммы, но позволяет пользователю взаимодействовать с данными в реальном времени, делая моделирование динамичным и интуитивно понятным. Система будет предоставлять возможность пользователю вводить данные о бизнесе, выбирать стратегии развития, управлять сотрудниками и отслеживать изменения в своей бизнес-модели.

Ключевой особенностью данного проекта является то, что пользователь сможет наблюдать визуальные эффекты, отображающие изменения в бизнесе, которые будут происходить под капотом системы. При этом для пользователя будет два типа бизнеса — физический и инвестиционный. В физическом бизнесе, например, при открытии фабрики, пользователь начнёт с одного маленького отдела, который со временем будет расширяться, добавляя новые отделы и увеличивая масштабы бизнеса. В инвестиционном же бизнесе пользователь будет работать с виртуальными деньгами, вкладывая их в криптовалюты, инвестиции и ставки, не рискуя реальными средствами. Это позволяет начинающим инвесторам и предпринимателям безопасно тестировать различные стратегии без финансовых рисков.

Программное средство будет работать на веб-платформе, обеспечивая пользователям удобный и гибкий доступ через браузер с возможностью использования как на мобильных, так и на десктопных устройствах. Важно отметить, что система будет ориентирована на работу с большим объёмом данных и обеспечит высокую производительность, что необходимо для стартапов, которые часто сталкиваются с быстро меняющимися условиями и требованиями.

Для реализации этого проекта было решено создать веб-приложение с возможностями динамичного взаимодействия в реальном времени. Оно будет обеспечивать максимальную скорость отклика, надежность работы, а также привлекать внимание пользователя через различные визуальные и интерактивные элементы. Все это позволит стартапам эффективно моделировать свои бизнес-процессы, тестировать стратегии и адаптироваться к изменениям рынка.

Система будет включать возможность ввода и обработки данных о бизнесе, а также предложит пользователю широкий выбор инструментов для анализа и визуализации бизнес-процессов, что позволит максимально эффективно управлять и оптимизировать деятельность компании. В будущем проект будет исследовать возможности использования современных платформ для создания и масштабирования таких приложений, а также обеспечения их безопасности и защиты данных.

Таким образом, цель проекта — разработать веб-приложение, которое предоставляет пользователю все необходимые функции для моделирования и оптимизации бизнес-процессов, предоставляя стартапам и предпринимателям мощный инструмент для развития бизнеса, от старта до масштабирования.

1. АНАЛИЗ АНАЛОГОВ, ЛИТЕРАТУРНЫХ ИСТОЧНИКОВ И ФОРМИРОВАНИЕ ТРЕБОВАНИЙ К ПРОЕКТИРУЕМОМУ ПРОГРАММНОМУ СРЕДСТВУ

Разработка программного обеспечения начинается с детального анализа предметной области, изучения существующих решений и формирования требований к будущей системе. Этот этап критически важен, поскольку именно здесь определяется основная проблема, которую должна решать система, и закладывается фундамент для её успешной реализации. В контексте нашего проекта — создания программного средства для моделирования бизнес-процессов в веб-среде для стартапов — цель состоит в разработке интуитивно понятного и интерактивного инструмента, который поможет начинающим предпринимателям анализировать и оптимизировать свои бизнес-процессы, включая как физический, так и инвестиционный бизнес.

Первоочередная задача — чётко сформулировать проблему: стартапы часто сталкиваются с недостатком доступных инструментов, которые одновременно просты в использовании, поддерживают динамическое моделирование и учитывают специфику их потребностей. Отсутствие ясности на этом этапе может привести к ошибкам в архитектуре и функциональности системы на более поздних стадиях разработки. Поэтому данная глава посвящена систематизации всех аспектов проекта: от анализа аналогов до формирования требований, что позволит заранее выявить слабые места и определить пути их устранения.

* 1. Аналитический обзор литературных источников

В данном разделе рассматриваются ключевые аспекты, которые оказывают влияние на определение требований, выбор технологических решений и разработку структуры программного средства для моделирования бизнес-процессов в веб-среде, ориентированного на стартапы. Основное внимание уделяется анализу платформ и подходов к разработке, которые обеспечивают удобство использования, производительность и гибкость создаваемого инструмента.

* + 1. Выбор платформы и подходы к разработке

Выбор платформы — один из определяющих этапов проектирования программного обеспечения. От этого решения зависят такие характеристики, как удобство использования, доступность приложения, его производительность, масштабируемость и перспективы дальнейшего развития. В условиях современного рынка существует множество платформ и сред разработки, каждая из которых обладает своими сильными и слабыми сторонами. Для стартапов, которым требуется быстрый доступ к инструментам управления бизнес-процессами с минимальными затратами, веб-приложения выделяются как универсальное решение благодаря своей кроссплатформенности, доступности через браузер и отсутствию необходимости в установке дополнительного программного обеспечения.

Кроссплатформенность веб-приложений обеспечивается использованием стандартов веб-разработки (HTML, CSS, JavaScript) и поддержкой браузеров на различных операционных системах. Это позволяет пользователям работать с приложением с любого устройства — будь то персональный компьютер, ноутбук, планшет или смартфон. Такой подход идеально подходит для стартапов, где важны мобильность и оперативность.

Кроссплатформенность веб-приложений обеспечивается использованием стандартов веб-разработки (HTML, CSS, JavaScript) и поддержкой браузеров на разных операционных системах. Это позволяет пользователям работать с приложением с любого устройства: компьютера, ноутбука, планшета или смартфона. Такой подход особенно подходит для стартапов, ценящих мобильность и скорость. Однако, веб-приложения имеют ограничения: зависимость от интернета, ограниченный доступ к аппаратным ресурсам устройства и сложности адаптации интерфейса под разные разрешения экранов.

При разработке веб-приложений для моделирования бизнес-процессов используются различные архитектурные подходы. Классическая модель клиент-серверного взаимодействия проста в реализации и стабильна, но создает высокую нагрузку на сервер и ограниченную интерактивность интерфейса. Архитектура одностраничных приложений (SPA) обеспечивает высокую производительность и плавность интерфейса, но сложнее в разработке и требует управления состоянием приложения на стороне клиента. Прогрессивные веб-приложения (PWA) сочетают преимущества веб- и мобильных приложений, предлагая офлайн-режим, push-уведомления и быструю загрузку, но требуют дополнительных усилий для реализации и адаптации.

Веб-технологии обладают преимуществами: доступностью с любого устройства через браузер, отсутствием необходимости установки ПО и простотой обновлений. Однако, существуют ограничения: зависимость от интернет-соединения, ограниченная работа с аппаратными ресурсами и необходимость адаптации под разные устройства.

Для стартапов веб-приложения являются оптимальным выбором, обеспечивая баланс между доступностью, гибкостью и скоростью развертывания. Успешная реализация требует комплексного подхода к выбору архитектуры и инструментов разработки.

* + 1. Технологические особенности целевых платформ

Наш продукт — программное обеспечение для моделирования бизнес-процессов, разработанное специально для стартапов, — создан на веб-платформе. Это решение обусловлено необходимостью кроссплатформенности, обеспечивающей доступ через браузер на любых устройствах (ПК, планшетах, смартфонах). Веб-технологии (HTML, CSS, JavaScript) гарантируют универсальность, позволяя стартапам использовать инструмент без установки дополнительного ПО, что идеально соответствует их динамичным задачам и ограниченным ресурсам.

Преимущества веб-платформы для стартапов включают универсальность и доступность (доступ через браузер из любой точки с интернетом), простоту обновлений и распространения (мгновенное обновление на сервере), и экономию ресурсов (разработка одной веб-версии вместо отдельных приложений для разных ОС).

Несмотря на преимущества, веб-приложения имеют ограничения: зависимость от интернета (редко проблематично для стартапов, работающих в офисах или коворкингах), ограниченный доступ к аппаратным ресурсам (не критично для моделирования бизнес-процессов), и потенциально меньшую производительность по сравнению с нативными приложениями (компенсируется кешированием и оптимизацией кода).

Для улучшения пользовательского опыта используются Progressive Web Apps (PWA) для офлайн-режима, быстрой загрузки и уведомлений, а также адаптивный дизайн для удобства на любых устройствах.

Для реализации продукта выбрана архитектура одностраничного приложения (SPA). Загрузка интерфейса происходит один раз, а обновления данных — асинхронно через взаимодействие с сервером. Это обеспечивает высокую интерактивность и мгновенную реакцию на действия пользователя.

Технологический стек включает Frontend (React), Backend (Python с FastAPI/Flask/Django), и базу данных (PostgreSQL или MongoDB).

Приложение проектируется с учетом разнообразия устройств: мобильные устройства (оптимизированный интерфейс) и ПК (сложные визуализации). Для стабильной работы применяется адаптивный дизайн, оптимизация ресурсов и кеширование.

Безопасность и кроссбраузерность являются приоритетами. Защита данных обеспечивается HTTPS, шифрованием и надежной аутентификацией. Приложение протестировано в популярных браузерах (Chrome, Firefox, Safari) на разных версиях.

Веб-платформа — идеальный выбор для программного обеспечения моделирования бизнес-процессов, ориентированного на стартапы. Универсальность, простота доступа, экономичность, использование SPA и PWA, оптимизация под разные устройства, безопасность и совместимость делают продукт удобным и надежным инструментом для управления бизнес-процессами.

* + 1. Принципы построения архитектуры программных систем

При разработке архитектуры веб-приложения для моделирования бизнес-процессов, ориентированного на стартапы, ключевыми являются принципы организации кода, модульность и интеграция компонентов. Эти аспекты определяют гибкость, масштабируемость и устойчивость к изменениям. Приложение должно быть простым в использовании, производительным, масштабируемым и гибким.

Рассмотрим несколько архитектурных подходов:

Монолитная архитектура: вся функциональность в едином кодовом базисе. Проста в реализации и управлении на ранних этапах, обладает высокой производительностью. Однако, ограничена в масштабируемости и сложна в обновлении. Подходит для быстрого старта, но недостаточно гибкая для долгосрочного развития.

Микросервисная архитектура: система разделена на независимые сервисы, каждый отвечает за отдельную функцию. Обладает высокой масштабируемостью, гибкостью и отказоустойчивостью. Однако, сложна в управлении и требует больших начальных затрат. Может быть избыточна для стартапов на начальном этапе, но перспективна для роста.

Многоуровневая (слоистая) архитектура: приложение разделено на слои: презентационный (интерфейс), бизнес-логики и доступа к данным. Обеспечивает четкое разделение задач, упрощает поддержку и тестирование. Возможны дублирование кода и накладные расходы на взаимодействие слоев. Идеально подходит для нашего приложения, разделяя интерфейс, логику и данные.

Сервисно-ориентированная архитектура (SOA): система как набор сервисов с четкими интерфейсами, удобна для интеграции с внешними системами. Лёгка в интеграции, гибка в выборе технологий. Однако, сложна в стандартизации интерфейсов и избыточна для начальной версии. Может пригодиться в будущем для подключения внешних данных.

Гибридный подход: сочетание элементов разных архитектур. Адаптивен и экономичен, но требует продуманного планирования. Отличный вариант для стартапов: быстрый запуск с возможностью масштабирования.

Рекомендация: начать с многоуровневой архитектуры с перспективой перехода на гибридный подход. Это обеспечивает простоту, масштабируемость и гибкость, балансируя скорость запуска и потенциал роста.

Выбор архитектуры критичен для успеха веб-приложения. Многоуровневая архитектура с возможностью эволюции в гибридную модель позволит создать удобный, производительный и масштабируемый продукт, развивающийся вместе с потребностями стартапов.

* + 1. Проектирование структур и баз данных

Процесс проектирования баз данных для нашего программного средства, ориентированного на стартапы и моделирование их бизнес-процессов, требует учета специфики задач. Нам нужно обеспечить быстрый доступ к данным, масштабируемость и целостность информации. Программное средство должно поддерживать точное и оперативное моделирование бизнес-процессов, предоставлять многопользовательский доступ для команд стартапов, гарантировать надежность данных для принятия бизнес-решений, быть готовым к росту числа пользователей и объема информации, а также защищать конфиденциальную информацию. Для этого мы создаем оптимальные структуры данных, выбираем подходящие технологии и внедряем механизмы оптимизации.

Нормализация данных играет важную роль в проектировании нашей системы. Она помогает минимизировать избыточность и устранить аномалии обновления, что особенно важно для стартапов, где данные о клиентах, транзакциях или ресурсах должны оставаться согласованными. Например, информацию о клиентах, такую как имя, контакты и история взаимодействий, мы разбиваем на таблицы в третьей нормальной форме, чтобы избежать дублирования и обеспечить целостность. Однако в ситуациях, где скорость доступа становится приоритетом, например для отчетов по бизнес-моделям, мы можем прибегнуть к денормализации, объединяя данные о заказах и клиентах в одну таблицу для ускорения запросов, даже если это увеличивает избыточность.

Быстродействие системы критично для стартапов, которым нужно оперативно анализировать свои бизнес-процессы. Для этого мы используем индексацию и оптимизацию запросов. Индексы создаются для часто используемых полей, таких как идентификаторы клиентов или даты транзакций, чтобы ускорить поиск. Если запросы фильтруют данные по нескольким параметрам, например по дате и статусу заказа, мы применяем композитные индексы. При этом избыточное количество индексов может замедлить операции вставки и обновления, поэтому мы тщательно анализируем типичные запросы пользователей и ограничиваемся только необходимыми индексами.

Чтобы еще больше повысить скорость работы и снизить нагрузку на базу данных, мы внедряем кэширование. Это особенно полезно для обработки повторяющихся запросов. На уровне приложения мы кэшируем результаты частых операций, таких как отображение текущих бизнес-моделей или аналитика по клиентам. Например, используя Redis, мы сохраняем временные данные, такие как статистика продаж за месяц, что позволяет сократить время отклика с секунд до миллисекунд.

Наше программное средство рассчитано на многопользовательский режим, где члены команды стартапа работают с данными одновременно. Для управления конкурентным доступом мы используем механизм транзакций, обеспечивая атомарное выполнение операций. Например, при обновлении бизнес-модели все изменения — ресурсы, этапы, бюджет — сохраняются одновременно или откатываются при ошибке. Мы также настраиваем минимально необходимые блокировки, чтобы избежать конфликтов, но не замедлять систему, предоставляя другим пользователям доступ к модели в режиме просмотра во время редактирования.

Выбор технологий хранения зависит от структуры данных и требований к производительности. В качестве основного решения для структурированных данных, таких как таблицы клиентов, заказов и процессов, мы используем PostgreSQL. Эта система обеспечивает высокую целостность, поддержку сложных запросов и транзакций, что идеально подходит для моделирования бизнес-процессов. Если потребуется гибкость для неструктурированных данных, например JSON-описаний бизнес-моделей, мы можем интегрировать MongoDB для горизонтального масштабирования.

Для защиты данных стартапов от потерь мы внедряем надежные механизмы резервного копирования. Ежедневно создаются резервные копии базы данных, а данные реплицируются на запасной сервер для обеспечения доступности системы даже при сбоях. Например, в PostgreSQL мы используем потоковую репликацию, чтобы синхронизировать данные в реальном времени.

Конфиденциальность данных — еще один ключевой аспект. Мы защищаем базу данных на всех уровнях. Доступ ограничивается ролями, такими как администратор команды или аналитик, а данные шифруются как при хранении, так и при передаче с использованием SSL/TLS. Чтобы предотвратить SQL-инъекции, мы применяем параметризованные запросы и регулярно обновляем систему для устранения уязвимостей.

Таким образом, проектирование баз данных для нашего программного средства — это комплексный процесс. Он включает моделирование данных с учетом нормализации и денормализации, оптимизацию через индексацию и кэширование, управление конкурентным доступом, выбор PostgreSQL как основной технологии с возможной интеграцией NoSQL, а также обеспечение безопасности и отказоустойчивости через шифрование и репликацию. Такой подход позволяет нашей системе эффективно поддерживать стартапы на всех этапах — от моделирования бизнес-процессов до масштабирования при росте компании.

* + 1. Современные тенденции и основы искусственного интеллекта

Современные исследования в области моделирования бизнес-процессов демонстрируют рост интереса к интерактивным и визуально ориентированным подходам. Традиционные методы бизнес-анализа, основанные на статических отчетах и графиках, постепенно уступают место динамическим моделям, которые позволяют пользователям не только анализировать данные, но и взаимодействовать с ними в реальном времени.

Одним из ключевых направлений является развитие цифровых симуляторов и инструментов прогнозирования, основанных на методах математического моделирования и интеллектуального анализа данных. Такие системы позволяют предпринимателям и инвесторам оценивать возможные сценарии развития бизнеса, тестировать стратегии управления и адаптироваться к изменениям рыночных условий.

Современные технологии визуализации, включая WebGL и интерактивные графические библиотеки (Three.js, D3.js), открывают новые возможности для создания реалистичных и интуитивно понятных бизнес-моделей. Использование облачных вычислений и распределенных систем обработки данных способствует повышению гибкости и производительности подобных решений.

Перспективным направлением также остается интеграция элементов геймификации и симуляционного обучения, что делает процесс освоения бизнес-стратегий более доступным и понятным для пользователей с разным уровнем подготовки. Таким образом, развитие подобных инструментов способствует упрощению процесса принятия решений и минимизации рисков при запуске и масштабировании бизнеса.

* 1. Обзор и анализ существующих аналогов

Разработка программного средства для моделирования бизнес-процессов требует анализа существующих решений, позволяющего выявить их сильные и слабые стороны, определить наиболее эффективные подходы и обоснованно сформулировать требования к разрабатываемой системе. Ниже приведен обзор наиболее популярных платформ и сервисов, связанных с бизнес-симуляциями и инструментами для моделирования.

* + 1. Sim Companies

Sim Companies – это онлайн-симулятор бизнеса, в котором пользователи могут управлять собственной виртуальной компанией, разрабатывать стратегии производства, торговли и инвестирования. В игре присутствуют различные экономические механики, такие как ценообразование, конкурентный анализ и логистика. Главная страница данной платформы представлена на рисунке 1.1.



Рисунок 1.1 – Главная страница сайта SimCompanies.com

Одним из ключевых преимуществ платформы является реалистичная экономическая модель, позволяющая игрокам анализировать спрос и предложение, управлять запасами и формировать прибыльные стратегии. Интерфейс приложения достаточно интуитивен, что делает процесс игры доступным для пользователей разного уровня подготовки.

Однако у Sim Companies есть и недостатки. Основное внимание уделяется экономическим расчетам и табличному анализу, а визуализация ограничивается простыми графиками и диаграммами. Это делает процесс взаимодействия с системой менее наглядным для пользователей, которым важны визуальные эффекты и наглядное представление роста их бизнеса.

* + 1. Virtonomics

В процессе проектирования модуля управления шаговым двигателем для применения в медицинском оборудовании необходимо тщательно изучить существующие схемотехнические и программные решения, реализованные в различных системах управления, от простейших лабораторных проектов до промышленных применений. Virtonomics – это экономическая стратегия, в которой пользователи могут строить и развивать свои компании в разных отраслях: от производства до финансовых услуг. Платформа предлагает гибкую систему управления бизнесом, включая маркетинг, наем персонала, закупки и логистику.

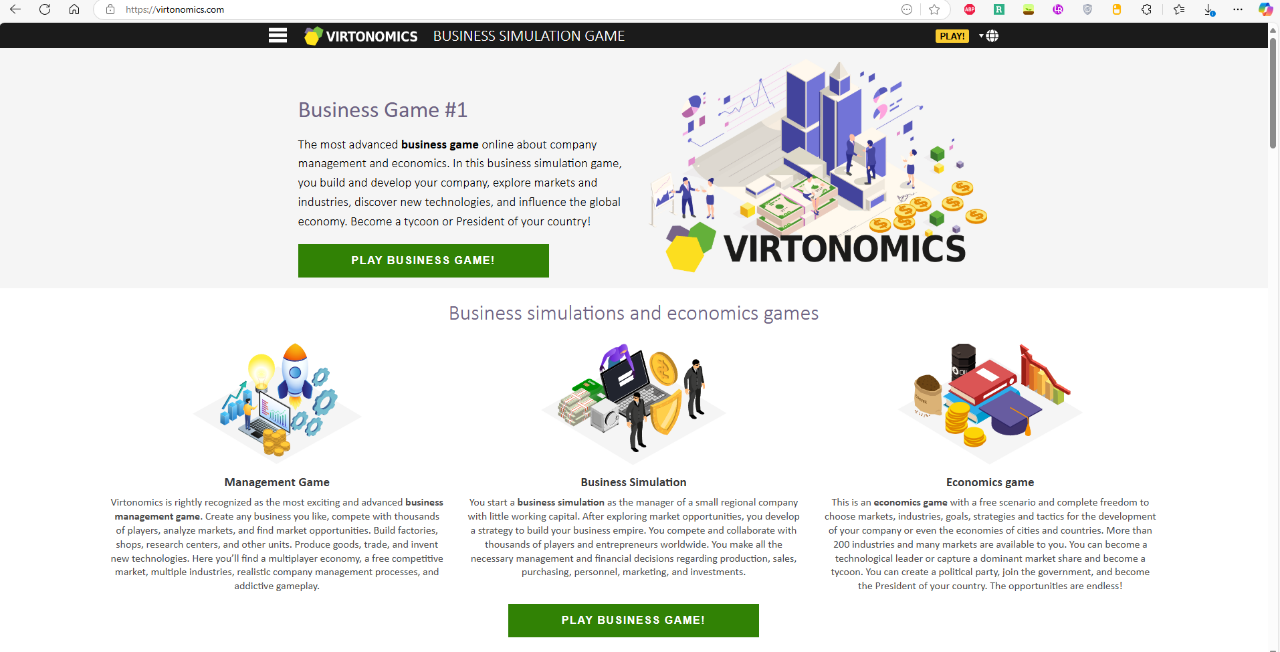


Рисунок 1.2 – Главная страница сайта Virtonomics.com

Основным преимуществом Virtonomics является глубина проработки бизнес-моделей и наличие реальной конкурентной среды, поскольку симуляция включает множество игроков, взаимодействующих друг с другом. Пользователи могут анализировать экономические тенденции, строить долгосрочные стратегии и адаптироваться к изменяющимся условиям рынка.

Тем не менее, Virtonomics имеет достаточно сложный интерфейс и требует значительного времени на освоение. Новичкам сложно разобраться в механиках игры, что может стать барьером для вовлечения пользователей. Кроме того, основное внимание уделяется экономическому моделированию, а визуальные элементы остаются второстепенными.

* + 1. Capitalism Lab

Capitalism Lab – это детализированный бизнес-симулятор, который позволяет игрокам управлять различными аспектами бизнеса, от производства до маркетинга и финансового планирования. Отличается высокой степенью реалистичности и гибкости настроек.



Рисунок 1.3 – Главная страница сайта CapitalismLab.com

Главное достоинство Capitalism Lab – это сложная и точная экономическая модель, основанная на реальных рыночных механиках. Пользователи могут экспериментировать с разными стратегиями ведения бизнеса, анализировать финансовые показатели и управлять целыми корпорациями.

Однако этот продукт ориентирован в первую очередь на профессионалов и энтузиастов экономических симуляций. Высокий порог вхождения и сложные финансовые расчеты могут затруднить освоение игры для начинающих пользователей. Кроме того, визуальная составляющая ограничивается базовыми графиками и диаграммами, что делает взаимодействие с системой менее интуитивным.

* 1. Требования к проектируемому программному средству

На основе проведенного анализа литературы и обзора существующих аналогичных решений необходимо сформулировать требования к проектируемому программному средству, учитывая функциональные и нефункциональные аспекты, потребности целевой аудитории, технические ограничения и результаты сравнительного анализа конкурентных решений, что позволит создать эффективный и конкурентоспособный продукт.

* + 1. Назначение проекта

Цель проекта заключается в создании веб-платформы, позволяющей стартапам моделировать и симулировать свои бизнес-процессы в интерактивном режиме. Инструмент поддерживает как моделирование физических бизнесов (например, заводов, магазинов), так и инвестиционных портфелей (например, акций, криптовалют). Платформа обеспечивает визуализацию в реальном времени и обратную связь, помогая стартапам принимать обоснованные решения без реальных рисков.

* + 1. Основные функции

Программное средство должно поддерживать следующие основные функции:

* Регистрация, аутентификация и авторизация пользователей.
* Система ролей (например, владелец стартапа, член команды, инвестор).
* Интерактивное моделирование физических бизнесов, включая управление ресурсами и найм сотрудников.
* Моделирование инвестиционных портфелей с распределением активов и отслеживанием доходности.
* Визуализация роста бизнеса или доходности портфеля в реальном времени через графики и анимации.
* Ввод данных о бизнес-параметрах, таких как бюджет, ресурсы и стратегии.
* Выбор стратегий развития для бизнеса или уровней риска для инвестиций.
* Симуляция бизнес-сценариев с обновлениями в реальном времени.
* Генерация отчетов и аналитики по производительности бизнеса или инвестиций.
* Уведомления о ключевых событиях или достижениях.

Эти функции формируют комплексное решение для моделирования, симуляции и оптимизации бизнес-процессов стартапов.

* + 1. Требования к входным данным

Входные данные вводятся пользователем через интерфейс и включают текстовые поля и опции выбора, такие как параметры бизнеса (начальный капитал, количество сотрудников), детали инвестиций (типы активов, суммы) и выбор стратегий (стратегии роста, уровни риска); при этом система должна проверять все данные на соответствие требуемым форматам и ограничениям, уведомляя пользователя об ошибках и запрашивая исправления при необходимости.

Система должна проверять все данные на соответствие требуемым форматам и ограничениям, уведомляя пользователя об ошибках и запрашивая исправления при необходимости.

* + 1. Требования к выходным данным

Выходные данные должны отображаться через различные элементы пользовательского интерфейса, предоставляя пользователю необходимую информацию в удобной и доступной форме.

* + 1. Требования к техническим и программным средствам

ЭВМ, на которой должна функционировать серверная часть программного средства, должна обладать следующими минимальными характеристиками:

1 Процессор: Intel Core i7 с тактовой частотой 3.0 ГГц или аналогичный многозадачный процессор с высокой производительностью(например, M4 Pro).

2 Оперативная память: 16 ГБ для обеспечения быстрой обработки запросов и работы с большими объемами данных.

3 Жесткий диск: SSD объемом 512 ГБ для быстрого доступа к данным и эффективного хранения информации о пользователях, книгах, обменах и других данных.

4 Сетевая карта: Ethernet 1 Гбит/с для стабильной передачи данных и обеспечения быстрой связи между сервером и пользователями.

Для работы серверной части программного средства должны быть установлены следующие программные компоненты:

1 Операционная система: Windows для обеспечения стабильности и безопасности работы серверной части или MacOS.

2 Веб-сервер: Nginx или Apache для обработки HTTP-запросов и балансировки нагрузки между различными компонентами системы.

3 Фреймворк для веб-разработки: React (JavaScript) для реализации серверной логики и обработки запросов пользователей и FastAPI(Python) для реализации логики.

4 База данных: PostgreSQL для хранения информации о пользователях, книгах, обменах, отзывах и других данных.

5 Контейнеризация: Docker для упаковки и развертывания приложения в контейнерах.

6 Система управления версиями: Git для контроля версий программного кода, а также GitHub для хранения репозиториев и обеспечения совместной работы над проектом.

Эти технические и программные средства обеспечат стабильную, быструю и безопасную работу платформы.

* + 1. Требования к информационной и программной совместимости

Программное обеспечение должно быть совместимо с различными информационными системами и программными средами, а также обеспечивать эффективную работу с внешними сервисами и компонентами.

1 Совместимость с операционными системами: Программное обеспечение должно поддерживать работу на операционных системах Windows, Linux и macOS для серверной и клиентской части, обеспечивая гибкость при развертывании.

2 Совместимость с веб-браузерами: Веб-клиент должен корректно функционировать в последних версиях популярных браузеров: Google Chrome, Mozilla Firefox, Microsoft Edge и Safari.

3 Совместимость с базами данных: Программное обеспечение должно быть совместимо с реляционными базами данных, такими как PostgreSQL или MySQL для хранения данных о пользователях, их бизнесах и других данных.

4 Совместимость с контейнерами и оркестраторами: Программное обеспечение должно быть совместимо с Docker для упаковки и развертывания приложения в контейнерах, а также с Kubernetes и другими оркестраторами контейнеров для автоматического масштабирования и управления контейнерами.

5 Интеграция с системами управления версиями: Проект должен быть совместим с системами управления версиями Git.

6 Совместимость с системами безопасности: Программное обеспечение должно поддерживать механизмы безопасности, такие как JWT (JSON Web Token) для аутентификации и авторизации пользователей.

7 Поддержка кроссплатформенности: Веб-приложение должно быть совместимо с мобильными устройствами, поддерживать адаптивный дизайн для различных разрешений экранов.

Эти требования обеспечат успешную интеграцию программного обеспечения с другими системами и его стабильную работу в различных технологических средах.

* + 1. Обоснование выбора языка и сред разработки

На основании анализа функциональных требований и специфики разрабатываемого программного средства, предназначенного для моделирования бизнес-процессов и ориентированного на стартапы, был сформирован следующий технологический стек: Python с FastAPI для серверной части, React для клиентской части, PostgreSQL в качестве базы данных, а также дополнительные инструменты, такие как WebSockets для real-time коммуникации, Pytest и Jest для тестирования, Flake8, Black, ESLint и Prettier для контроля качества кода, и Visual Studio Code как основная среда разработки.

Этот выбор обусловлен стремлением создать высокопроизводительный, масштабируемый и удобный в использовании инструмент, который отвечает потребностям стартапов. Ниже представлено подробное обоснование каждого компонента.

Для разработки серверной части был выбран язык программирования Python в сочетании с фреймворком FastAPI. Этот выбор обусловлен несколькими ключевыми факторами [1].

Для клиентской части был выбран React — библиотека JavaScript, широко используемая для создания современных пользовательских интерфейсов.

React позволяет разбивать интерфейс на независимые компоненты, что упрощает разработку, тестирование и поддержку кода. Использование виртуального DOM минимизирует прямые манипуляции с реальным DOM, обеспечивая быструю отрисовку интерфейса. React идеально подходит для создания динамичных интерфейсов, где пользователи могут взаимодействовать с моделью бизнес-процессов. Кроме того, React обладает огромным сообществом разработчиков и множеством готовых библиотек, что ускоряет разработку и позволяет легко добавлять новые функции.

React стал стандартом для построения современных веб-приложений, и его выбор гарантирует удобство использования и гибкость интерфейса нашего инструмента.

В качестве системы управления базами данных была выбрана PostgreSQL.

PostgreSQL обеспечивает строгую согласованность данных и поддержку транзакций. Возможность выполнять сложные SQL-запросы позволяет эффективно обрабатывать данные о бизнес-процессах. Система поддерживает работу с различными типами данных, включая JSON, геоданные и пользовательские типы, что открывает возможности для дальнейшего расширения функциональности платформы. Кроме того, PostgreSQL хорошо справляется с ростом объема данных и нагрузки, что важно для стартапов, планирующих масштабировать свой продукт [2].

Для обновлений и симуляций в реальном времени используются WebSockets, интегрированные с FastAPI.

WebSockets позволяют устанавливать двустороннюю связь между клиентом и сервером, что идеально для передачи данных о состоянии бизнес-моделей без необходимости постоянных запросов от клиента. FastAPI предоставляет встроенные инструменты для работы с WebSockets, что упрощает их внедрение и управление. Благодаря прямой коммуникации WebSockets обеспечивают мгновенную передачу данных, что важно для интерактивного опыта пользователей.

Использование WebSockets делает наше приложение динамичным и отзывчивым.

Для обеспечения качества кода мы выбрали следующие инструменты.

Pytest поддерживает модульные и интеграционные тесты, обладает мощными функциями, такими как параметризация и фикстуры, и легко интегрируется с FastAPI. Jest является стандартом для тестирования React-приложений, поддерживает snapshot-тестирование и проверку компонентов, что гарантирует корректность интерфейса при изменениях.

Комбинация Pytest и Jest позволяет поддерживать высокое качество кода и минимизировать ошибки.

Для поддержания единообразия и чистоты кода используются следующие инструменты.

Flake8 является линтером для Python, выявляющим стилистические и логические ошибки. Black — это автоматический форматтер, который приводит код к единому стилю. ESLint помогает следовать лучшим практикам JavaScript/React, а Prettier автоматически форматирует код для улучшения читаемости.

Эти инструменты автоматизируют процесс контроля качества, позволяя разработчикам сосредоточиться на реализации функциональности.

В качестве основной среды разработки был выбран PyCharm.

PyCharm предоставляет мощные инструменты для работы с Python, включая интеллектуальную навигацию по коду, рефакторинг, поддержку отладки и тестирования. Он обладает встроенной интеграцией с системами контроля версий, такими как Git, а также возможностью работы с базами данных и Docker-контейнерами. PyCharm упрощает процесс разработки за счет множества встроенных функций и гибких настроек.

Выбранный технологический стек — Python с FastAPI для серверной части, React для клиентской части, PostgreSQL для хранения данных, WebSockets для real-time коммуникации, а также инструменты тестирования (Pytest, Jest) и контроля качества кода (Flake8, Black, ESLint, Prettier) в сочетании с PyCharm — обеспечивает разработку высокопроизводительного, масштабируемого и удобного инструмента для моделирования бизнес-процессов.

Этот подход позволяет создать платформу, которая отвечает потребностям стартапов, предоставляя им интерактивные и эффективные средства для управления и оптимизации своих бизнес-процессов.

1. АНАЛИЗ ТРЕБОВАНИЙ К ПРОГРАММНОМУ СРЕДСТВУ И РАЗРАБОТКА ФУНКЦИОНАЛЬНЫХ ТРЕБОВАНИЙ

Проектируемая система предназначена для моделирования бизнес-процессов в веб-среде, ориентирована на стартапы и предоставляет возможность симулировать как физические бизнесы (например, заводы, магазины), так и инвестиционные портфели (например, акции, криптовалюты). В данном разделе проводится анализ требований к разрабатываемому программному средству, формируются функциональные требования, а также разрабатываются спецификации, необходимые для проектирования системы [3].

* 1. Функциональная модель программного средства

Функциональность разрабатываемого программного средства представлена в виде диаграммы вариантов использования (Use Case Diagram), выполненной в соответствии с нотацией UML. Диаграмма позволяет наглядно отразить основные сценарии взаимодействия пользователей с системой, определяет роли пользователей и их функциональные возможности, а также иллюстрирует взаимосвязи между действиями в системе.

С учетом требований, сформулированных в разделе 1.3, разрабатываемое программное средство должно обеспечивать поддержку ролевой модели. Это предполагает разделение пользователей на несколько категорий с различными уровнями доступа и набором прав, включая гостя, зарегистрированного пользователя и администратора.

Функциональные возможности пользователей системы представлены на рисунке 2.1 в виде диаграммы вариантов использования, которая наглядно демонстрирует доступные действия для каждой категории пользователей, а также их взаимосвязи внутри программного средства [4].

Регистрация, аутентификация и авторизация – это функции, которые доступны для роли «Гость» (пользователь, не зарегистрированный в системе). В рамках приложения реализована возможность входа с помощью собственной системы авторизации, а также возможность регистрации с помощью внешних поставщиков данных (например, Google). Также пользователь с ролью «Гость» имеет возможность просмотра каталога бизнес-моделей, представленных на платформе.

После регистрации пользователь получает доступ к различным функциям системы, включая управление профилем, работу с личными бизнес-моделями, поиск и обмен бизнес-идеями, взаимодействие с другими пользователями, а также возможность оставлять отзывы [5]. Рассмотрим каждую из представленных групп функций подробнее.

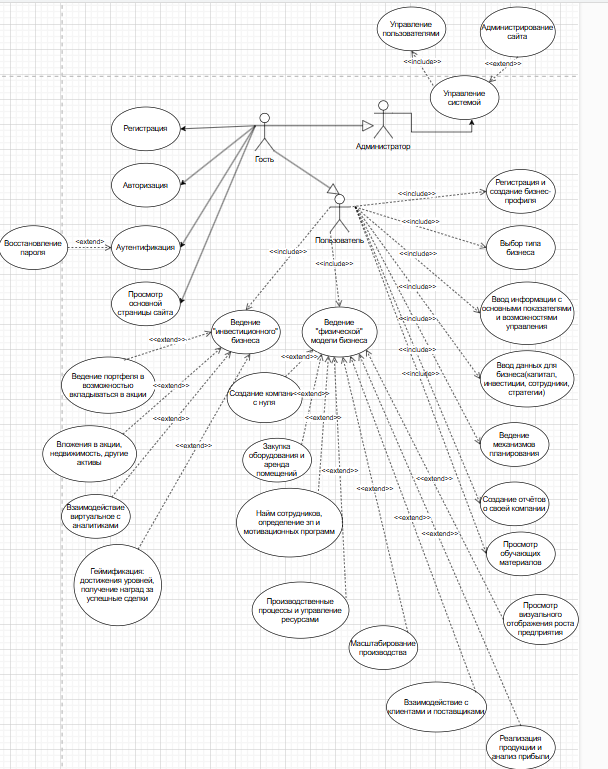


Рисунок 2.1 – Диаграмма вариантов использования ПС(UML)

Система имеет три типа пользователей: гости (ограниченный доступ, просмотр главной страницы, регистрация и авторизация); пользователи (полноценное управление бизнесом после регистрации); и администраторы (управление системой и пользователями).

Функции включают регистрацию и авторизацию пользователей, аутентификацию, восстановление пароля, и просмотр главной страницы. Пользователи управляют бизнесом, выбирая между "инвестиционным" (создание портфеля, вложения в активы, взаимодействие с экспертами, геймификация) и "физическим" (создание компании, закупка оборудования, найм сотрудников, управление производством, взаимодействие с клиентами, реализация продукции).

Дополнительные функции включают регистрацию бизнес-профиля, выбор стратегии, ввод данных, создание отчетов, просмотр аналитики и обучающих материалов, а также визуальное отображение роста предприятия. Администраторы управляют пользователями, системой и сайтом.

Предметная область программного средства включает ключевые сущности и соответствующие им атрибуты, которые отражают структуру и характеристики данных, используемых в системе.

Пользователь:

* Идентификатор пользователя.
* Имя.
* Электронная почта.
* Хешированный пароль.
* Дата регистрации.
* Дата последней авторизации.
* Роль (гость, пользователь, администратор).
* Фото профиля.
* Выбранный тип бизнеса.
* Финансовый баланс.
* История операций.

Бизнес-профиль:

* Идентификатор профиля.
* Идентификатор пользователя.
* Название компании.
* Тип бизнеса (инвестиционный, производственный).
* Дата создания бизнеса.
* Капитал компании.
* Список сотрудников.
* Активы компании (оборудование, недвижимость).

Инвестиционный портфель:

* Идентификатор портфеля.
* Идентификатор пользователя.
* Список активов (акции, облигации, недвижимость).
* Истории инвестиции.
* Дивиденды и доходность.

Производственная модель бизнеса:

* Идентификатор компании.
* Идентификатор пользователя.
* Список сотрудников и их роли.
* Закупленное оборудование.
* Производственные мощности.
* Складские запасы.
* История производства.
* Доходы и расходы.

Финансовые транзакции:

* Идентификатор транзакции.
* Идентификатор пользователя / компании.
* Идентификатор клиента или поставщика.
* Дата заключения контракта.
* Условия контракта.
* Статус выполнения.

Сообщения и уведомления:

* Идентификатор сообщения.
* Идентификатор отправителя.
* Идентификатор получателя.
* Текст сообщения.
* Дата и время отправки.
* Статус прочтения.

Взаимодействие с клиентами и поставщиками:

* Идентификатор контракта.
* Идентификатор компании.
* Идентификатор клиента или поставщика.
* Дата заключения контракта.
* Условия контракта.
* Статус выполнения.

Отчёты и аналитика:

* Идентификатор отчёта.
* Идентификатор пользователя или компании.
* Тип отчёта (финансовый, производственный, аналитический).
* Дата создания.
* Содержимое отчёта.

Эта инфологическая модель базы данных описывает все ключевые сущности и их атрибуты, которые необходимы для реализации системы с поддержкой ролевой модели пользователей, а также их взаимодействия через обмен книгами, отзывы, подписки и сообщения [6].

* 1. Разработка спецификаций функциональных требований

Учитывая требования из раздела 1.3, представим детализированное описание функциональных возможностей платформы для моделирования бизнеса. Этот раздел описывает ключевые функции, их назначение и требования к реализации.

* + 1. Функция регистрации

Создает учетную запись нового пользователя. Доступна только незарегистрированным пользователям (роль «Гость»). Требуется указать электронную почту и пароль, или использовать внешний сервис аутентификации. Электронная почта должна быть уникальной и соответствовать формату. Пароль должен соответствовать требованиям безопасности. При ошибке, пользователь получает сообщение с указанием причины. После успешной регистрации, пользователь получает роль «Пользователь» и доступ к системе. Включает возможность восстановления пароля через электронную почту.

* + 1. Функция аутентификации и авторизации

Проверяет учетные данные пользователя и определяет его права. Пользователь вводит логин и пароль, или использует внешние сервисы. При успешной аутентификации, система фиксирует время последнего входа. При ошибке, предоставляется возможность повторной попытки или восстановления доступа [7]. Система определяет роль пользователя (гость, пользователь, администратор) и открывает соответствующий доступ. Ограничено количество неудачных попыток входа.

* + 1. Функция регистрации и ведения бизнес-профиля

Позволяет создать бизнес-профиль и вести деятельность. Пользователь указывает название компании и тип бизнеса (инвестиционный или производственный). Вводится основная информация о компании, включая капитал, сотрудников и стратегию. Предусмотрена возможность редактирования и удаления профиля. Включает ведение отчетности и генерацию аналитики. Реализована визуализация роста предприятия.

* + 1. Функция ведения инвестиционного бизнеса

Управление инвестиционным портфелем и его ростом. Пользователь может инвестировать в акции, недвижимость и другие активы. Возможность управления портфелем и анализа доходности. Взаимодействие с виртуальными агентами. Реализована геймификация: уровни и награды.

* + 1. Функция ведения производственного бизнеса

Управление процессами компании. Пользователь может создавать компанию с нуля. Включает систему закупки оборудования и аренды помещений, найм сотрудников, управление производством и ресурсами, масштабирование производства и взаимодействие с клиентами и поставщиками.

* + 1. Функция управления финансами

Отслеживание движения финансов. Ведение учета доходов и расходов, подключение системы платежных операций, ведение данных о заработной плате и анализ финансовых показателей.

* + 1. Функция отчетности и аналитики

Инструменты для анализа бизнеса. Автоматическая генерация отчетов, визуализация данных, и просмотр обучающих материалов и рекомендаций.

* + 1. Функция администратора

Расширенные права. Управление пользователями (блокировка, удаление, изменение ролей), администрирование системы (контроль бизнес-профилей, удаление аккаунтов), мониторинг активности пользователей, обработка жалоб и отчетность [8].

Раздел 2.2 подробно описывает функциональные возможности платформы для моделирования бизнеса, охватывая все этапы от регистрации пользователя до администрирования системы. Особое внимание уделено разделению функций на основе типов пользователей (гость, пользователь, администратор), а также на различных моделях бизнеса (инвестиционный и производственный). Детальное описание требований к каждой функции обеспечивает четкое понимание целей разработки и позволяет создать полноценную и масштабируемую платформу, способную удовлетворить потребности широкого круга пользователей. Комплексный подход к описанию функциональности позволяет минимизировать риски и обеспечить эффективное развитие проекта.

1. ПРОЕКТИРОВАНИЕ И РАЗРАБОТКА ПРОГРАММНОГО СРЕДСТВА
   1. Разработка архитектуры программного средства

На основании анализа вариантов проектирования приложения, приведенного в пункте 1.3, а также обзора технологических особенностей платформ, описанных в пункте 1.1.2, было принято решение выбрать в качестве основы для разработки платформу веб-приложений. Такой выбор обусловлен кроссплатформенностью, доступностью через браузеры и отсутствием необходимости установки дополнительного ПО, что идеально соответствует потребностям стартапов, для которых разрабатывается наше программное средство.

Классической архитектурой для веб-приложений является двухзвенная клиент-серверная модель. Схема такой архитектуры представлена на рисунке 3.1. В этой модели сервер отвечает за обработку запросов и возврат статических страниц с информацией, а клиентская часть отображает полученные данные. Однако современные приложения, включая наше, требуют высокого уровня интерактивности, чтобы пользователи могли активно взаимодействовать с системой, например, моделировать бизнес-процессы, изменять параметры и получать результаты в реальном времени. Простая двухзвенная архитектура не способна в полной мере обеспечить такую функциональность.

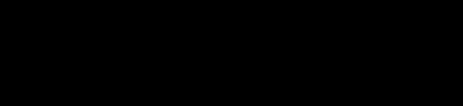


Рисунок 3.1 – Схема двухзвенной архитетуры

Исходя из требований к интерактивности и производительности, было принято решение разработать насыщенное веб-приложение (Rich Internet Application, RIA), основанное на архитектуре одностраничного приложения (Single Page Application, SPA). Схема такой архитектуры представлена на рисунке 3.2. В отличие от традиционных веб-приложений, SPA загружает основной код приложения один раз, а последующие обновления данных происходят асинхронно без перезагрузки страницы. Это обеспечивает высокую скорость работы и плавный пользовательский опыт, что критически важно для моделирования бизнес-процессов в реальном времени.

Преимущества использования насыщенных веб-приложений для нашего проекта включают:

* возможность выполнения вычислений на стороне клиента, что снижает нагрузку на сервер и ускоряет отклик интерфейса;
* асинхронная загрузка данных, позволяющая получать информацию с сервера до того, как пользователь запросит конкретные данные, что улучшает восприятие скорости работы приложения;
* снижение нагрузки на сервер за счет распределения вычислений между клиентом и сервером, что повышает способность системы обрабатывать большее количество одновременных пользовательских сессий.

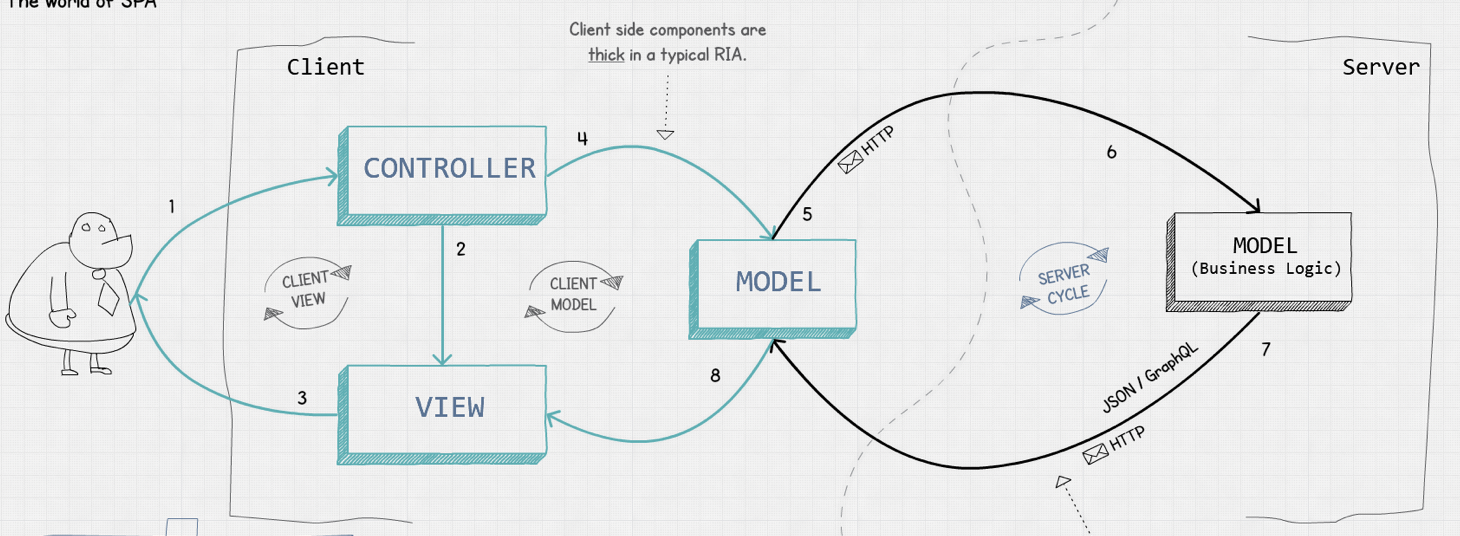


Рисунок 3.2 — Схема RIA архитектуры

Для реализации клиентской части приложения выбран фреймворк React, как указано в пункте 1.3.7. React поддерживает компонентный подход, что упрощает разработку сложных интерфейсов, таких как интерактивные панели для управления бизнес-моделями или визуализации данных. Благодаря использованию виртуального DOM, React обеспечивает быструю отрисовку изменений, что особенно важно для динамичных элементов, таких как графики и анимации, отображающие рост бизнеса или доходность портфеля.

На стороне сервера будет использоваться Python с FastAPI, как указано в пункте 1.3.7. FastAPI поддерживает асинхронные операции, что позволяет эффективно обрабатывать запросы в реальном времени, такие как обновления данных бизнес-моделей или симуляции инвестиционных сценариев. Кроме того, FastAPI обеспечивает автоматическую генерацию документации API, что упрощает интеграцию серверной и клиентской частей, а также последующее тестирование.

Для хранения данных выбрана реляционная база данных PostgreSQL, как указано в пункте 1.3.7. PostgreSQL обеспечивает высокую целостность данных, поддержку сложных запросов и транзаций, что необходимо для многопользовательской системы, где стартапы работают с бизнес-моделями, параметрами и аналитикой. Реляционная структура базы данных оптимальна для структурированных данных, таких как информация о пользователях, бизнес-моделях, транзакциях и отчетах.

Для обеспечения real-time функциональности, например, мгновенного обновления визуализаций или уведомлений о событиях, используются WebSockets, поддерживаемые FastAPI. Это позволяет установить постоянное соединение между клиентом и сервером, минимизируя задержки при передаче данных и обеспечивая интерактивный опыт, который соответствует ожиданиям современных пользователей.

В отличие от подхода Backend as a Service (BaaS), который предполагает использование сторонних облачных решений, таких как Firebase, мы выбрали собственную серверную инфраструктуру с FastAPI и PostgreSQL. Это решение обеспечивает полный контроль над данными, гибкость в реализации специфических функций и возможность масштабирования под будущие потребности стартапов, например, интеграцию с внешними аналитическими сервисами или добавление новых типов бизнес-моделей.

Таким образом, архитектура нашего программного средства базируется на модели SPA с использованием React для клиентской части, FastAPI для серверной части, PostgreSQL для хранения данных и WebSockets для real-time коммуникации. Эта структура обеспечивает высокую производительность, интерактивность и масштабируемость, необходимые для эффективного моделирования бизнес-процессов стартапов.

* 1. Разработка датологической и физической базы данных

Как было указано в пункте 1.3.7, для хранения данных в нашем программном средстве, предназначенном для моделирования бизнес-процессов стартапов, выбрана реляционная база данных PostgreSQL. Этот выбор обусловлен необходимостью работы со структурированными данными, такими как параметры бизнес-моделей, пользовательские профили и аналитические отчеты, а также требованиями к целостности данных и поддержке сложных запросов. В отличие от нереляционных баз данных, PostgreSQL обеспечивает строгую согласованность и транзакционную надежность, что критично для многопользовательской системы, где стартапы одновременно взаимодействуют с бизнес-моделями.

Датологическая модель базы данных была разработана на основе инфологической модели, описанной в пункте 2.1, и включает сущности, такие как пользователи, бизнес-модели, личные коллекции, отзывы, обмены, подписки и сообщения. Каждая сущность представлена в виде таблицы с четко определенными полями и связями. Реляционная структура позволяет эффективно управлять данными, минимизировать избыточность и обеспечить быстрый доступ к информации через SQL-запросы.

Для реализации физической базы данных используется PostgreSQL, которая настраивается через инструменты администрирования, такие как pgAdmin или командная строка (см. рисунок 3.3). База данных создается в рамках проекта и подключается к серверной части приложения, реализованной на Python с FastAPI. PostgreSQL поддерживает высокую производительность и масштабируемость, что делает ее подходящей для обработки запросов от множества пользователей, работающих с динамическими бизнес-моделями.

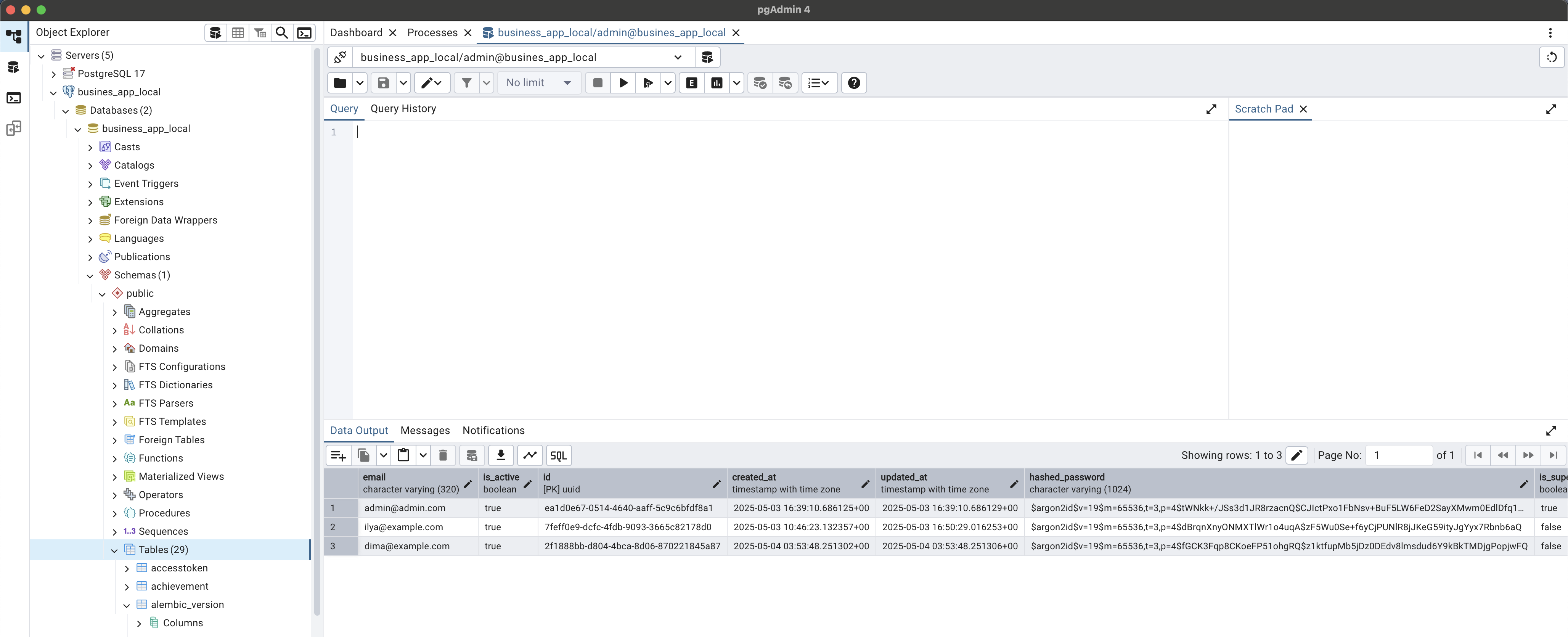


Рисунок 3.3 – Интерфейс pgAdmin для управления PostgreSQL

Для обеспечения быстрого доступа к данным и оптимизации производительности применяются индексы. Например, индексы создаются для часто запрашиваемых полей, таких как идентификаторы пользователей (user\_id) или бизнес-моделей (model\_id), чтобы ускорить поиск и фильтрацию. Для сложных запросов, включающих несколько условий (например, выбор бизнес-моделей по типу и дате создания), используются композитные индексы. Однако, чтобы избежать замедления операций вставки и обновления, количество индексов тщательно балансируется на основе анализа типичных сценариев использования.

Кэширование данных реализуется с использованием Redis, интегрированного с серверной частью на FastAPI. Это позволяет хранить результаты часто выполняемых запросов, таких как статистика по бизнес-моделям или аналитика по инвестиционным портфелям. Например, если пользователь запрашивает отчет о доходности портфеля, результат сохраняется в Redis, и последующие запросы обрабатываются быстрее, снижая нагрузку на PostgreSQL. Для управления кэшем используется следующая логика:

from fastapi import FastAPI

from redis.asyncio import Redis

app = FastAPI()

redis = Redis(host='localhost', port=6379, db=0)

async def get\_cached\_report(user\_id: str, report\_type: str):

cache\_key = f"report:{user\_id}:{report\_type}"

cached\_data = await redis.get(cache\_key)

if cached\_data:

return cached\_data.decode('utf-8')

# Если данных нет в кэше, запрашиваем из PostgreSQL

report\_data = await fetch\_report\_from\_db(user\_id, report\_type)

await redis.setex(cache\_key, 3600, report\_data) # Кэшируем на 1 час

return report\_data

Для поддержки многопользовательского доступа и предотвращения конфликтов при одновременной работе с данными применяются транзакции. Например, при обновлении параметров бизнес-модели все изменения (ресурсы, бюджет, этапы) сохраняются атомарно, чтобы избежать несогласованности. FastAPI позволяет легко интегрировать управление транзакциями с PostgreSQL через библиотеку SQLAlchemy:

from sqlalchemy.orm import Session

def update\_business\_model(db: Session, model\_id: int, updates: dict):

db.execute("BEGIN TRANSACTION")

try:

db.query(BusinessModel).filter(BusinessModel.id == model\_id).update(updates)

db.commit()

except Exception as e:

db.rollback()

raise e

Для обеспечения офлайн-доступа к данным, например, при временной потере соединения, мы используем локальное кэширование на стороне клиента с помощью IndexedDB. Это позволяет временно сохранять данные о бизнес-моделях или отчетах в браузере пользователя. При восстановлении соединения данные синхронизируются с сервером.

Операции, связанные с сохранением данных в IndexedDB, выполняются асинхронно, чтобы не блокировать интерфейс. При восстановлении соединения клиентская часть отправляет накопленные изменения на сервер через API FastAPI, что обеспечивает надежность работы системы.

Для повышения отказоустойчивости настроена репликация данных в PostgreSQL. Основной сервер синхронизируется с запасным через потоковую репликацию, что позволяет минимизировать риск потери данных при сбоях. Регулярные резервные копии создаются с помощью утилиты pg\_dump и хранятся в защищенном облачном хранилище.

Безопасность базы данных обеспечивается на нескольких уровнях:

* Шифрование: Данные передаются по HTTPS, а в PostgreSQL используется шифрование на уровне хранения.;
* Аутентификация: Доступ к базе ограничен через роли и пароли, настроенные в PostgreSQL.
* Защита от атак: Параметризованные запросы в SQLAlchemy предотвращают SQL-инъекции.

Таким образом, физическая база данных на основе PostgreSQL, дополненная кэшированием в Redis и локальным хранением в IndexedDB, обеспечивает высокую производительность, надежность и безопасность. Датологическая модель, построенная на реляционной структуре, позволяет эффективно управлять данными, поддерживая сложные сценарии моделирования бизнес-процессов для стартапов.

* 1. Проектирование и разработка алгоритмов программного средства
     1. Оболочка приложения

Концепции прогрессивных веб-приложений подразумевают, как уже было сказано выше, поведение, максимально приближенное к так называемым native-приложениям, что, в свою очередь, включает минимальную скорость отклика и инициализации, однако достичь этого, в рамках работы с интернет-ресурсами, как правило, бывает крайне сложно, ведь контент, к которому пользователь получает доступ, находиться на удалённом ресурсе, на скорость доступа к которому может влиять масса факторов, часть из которых никак не зависит и неподконтрольна конечному пользователю, вследствие чего поведение приложения может быть крайне непредсказуемым, а время так называемого first meaningful paint (периода от отправки первичного запроса до отрисовки приложения на экране клиента) может растягиваться до очень больших значений, что, в свою очередь, негативно сказывается на UX (опыте пользователя) и даже понизить порог вовлеченности, ведь, согласно исследованиям [9], при длительной первичной отрисовке (более 2 секунд) до 80% может закрыть вкладку с медленным сайтом так и не дождавшись окончательной загрузки и выполнения функции, которую они собирались сделать, что особенно негативно сказывается на сайтах и веб-приложениях категории e-commerce (интернет-магазины и т.д.), а значит, и к потере прибыли или рейтингов конверсии посетителей, а это, как известно, уже не голые слова, а реальная прибыль, которую может потерять ресурс просто из-за того, что система вовремя не загрузилась, что может быть вызвано рядом факторов, связанных как с самим устройством клиента, так и с каналом данных и, что немаловажно, с самим веб-сервисом, при создании которого, возможно, не были выполнены необходимые оптимизации контента, к которому получает доступ пользователь, а ведь правильное проектирование приложения и организация контента, а также использование современных технологий, функций веб-обозревателей, таких как предзагрузка (prefetch/preload), и серверных протоколов (HTTP\2) позволяют значительно снизить (можно сказать, свести к нулю) время загрузки приложения и его инициализации в полностью (или частично) функциональное состояние, что позволит пользователю получить first meaningful paint с основными элементами интерфейса, которые принято называть app shell (см. рисунок 3.4), ввиду того, что они представляют собой лишь каркас приложения (пусть даже без основного функционала), однако даже такой трюк заставит пользователя сохранить фокус на выполняемой функции и снизит вероятность того, что загружаемая вкладка будет закрыта из-за долгой загрузки.

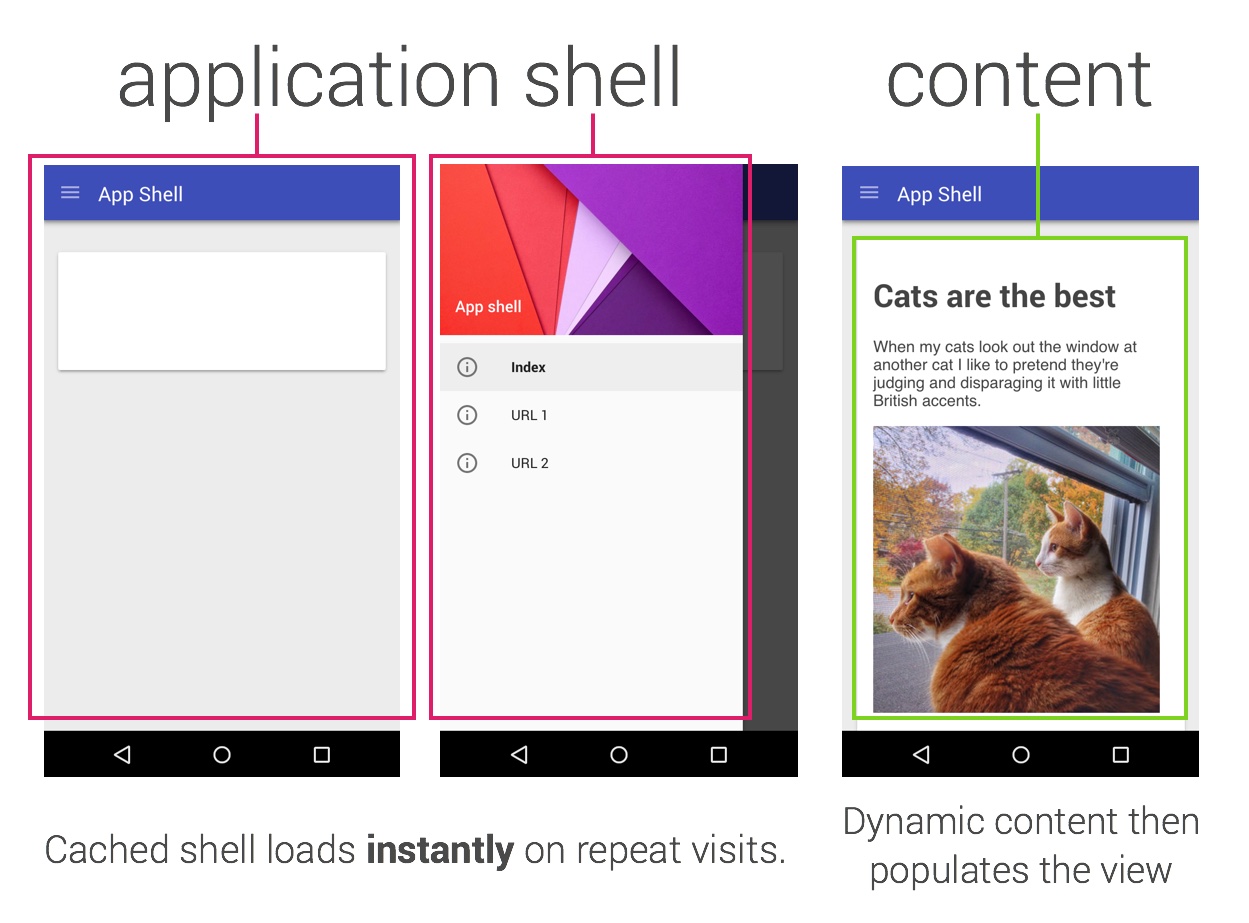


Рисунок 3.4 – Концепция Application Shell

Оболочка, как правило, освобождается от скриптовой составляющей, картинок и динамически загружаемого контента. Изначально, согласно спецификации, все скрипты приложения рекомендовалось загружать в тэге <head> страницы, однако эта часть загружается перед контентом, что влечёт за собой задержки загрузки, для предотвращения которых большинство скриптов помещается в нижнюю часть страницы, которая будет загружена и запущена после основного контента.

Ещё одним способом получить ускоренный отклик и не потерять пользователя является использование так называемых plachoders (см. рисунок 3.5). В этом случае отрисовывается не только каркас приложения, но и обозначаются основные блоки контента, которые со временем будут заменятся реальными данными, полученными с сервера. В данный момент множество популярных сервисов (в том числе и native-приложений), работающих с удалёнными данными, прибегает к этой уловке: Google Maps, YouTube и так далее.

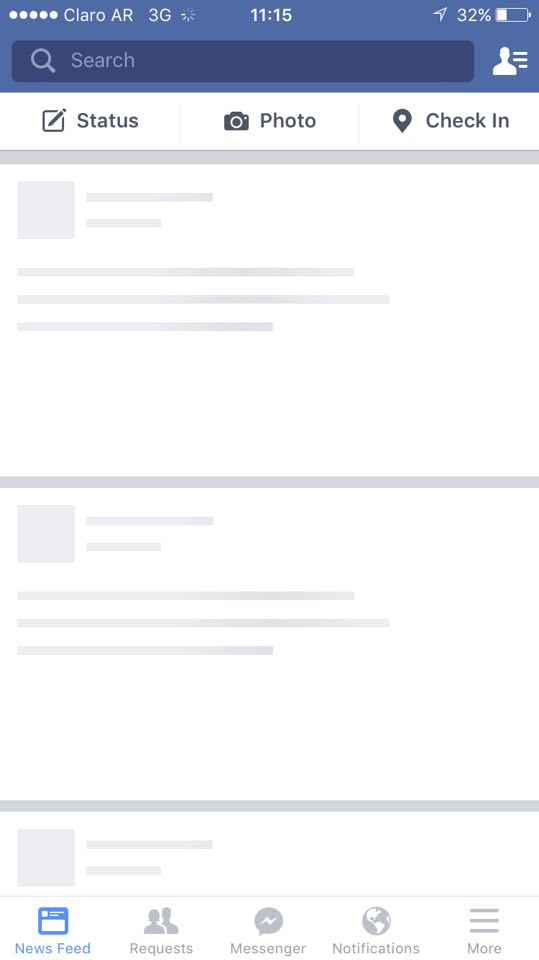


Рисунок 3.5 – Content placeholder

При использовании сторонних фреймворков вопрос отклика стоит ещё острее: кроме загрузки разметки, стилей и ресурсов необходимо выполнить и некоторый код для запуска функциональности этого фреймворка. В случае с React вступает в силу понятие рендериннга компонента. Библиотеке необходимо найти в стандартной разметке страницы все тэги, компоненты которых были определены, заменить (добавить) их содержимое, загрузить необходимые дополнительные ресурсы и запустить логику каждого компонента. Этот процесс, в зависимости от количества и функциональной насыщенности компонентов, может занимать достаточно длительный период времени, который и будет отнимать драгоценное время загрузки у пользователя. Но и игнорировать этот период нельзя: от момента загрузки разметки страницы до момента окончания апгрейда наступает так называемый FOUC (flash of unstyled content), который приводит к скачкам контента на экране и, как результат, негативному опыту использования.

* + 1. Аутентификация

Для идентификации пользователей в системе мы пользуемся сервисом Firebase, а для его интеграции в само приложение существует набор компонентов от разработчиков сервиса под названием FirebaseUIAuth. При помощи этих компонентов можно получать доступ к основным функциям firebase используя лишь декларативный подход.

Параметры авторизации выдаются системой при регистрации нового проекта и однозначно идентифицируют приложение, которое получает доступ к Firebase.

Одним из параметров является провайдер данных. Firebase поддерживает несколько провайдеров авторизации (cм рисунок 3.6):

* Google;
* Twitter;
* Facebook;
* GitHub;
* Email/password;
* Anonymous;
* Custom provider.

Кроме того, существует возможность создания собственного провайдера, Firebase предоставляет обширную документацию с примерами использования.

При работе с клиент-серверной архитектурой, доступ пользователя к различным частям и функциям приложения контролируется посредством формирования шаблона страницы с учётом прав доступа пользователя.

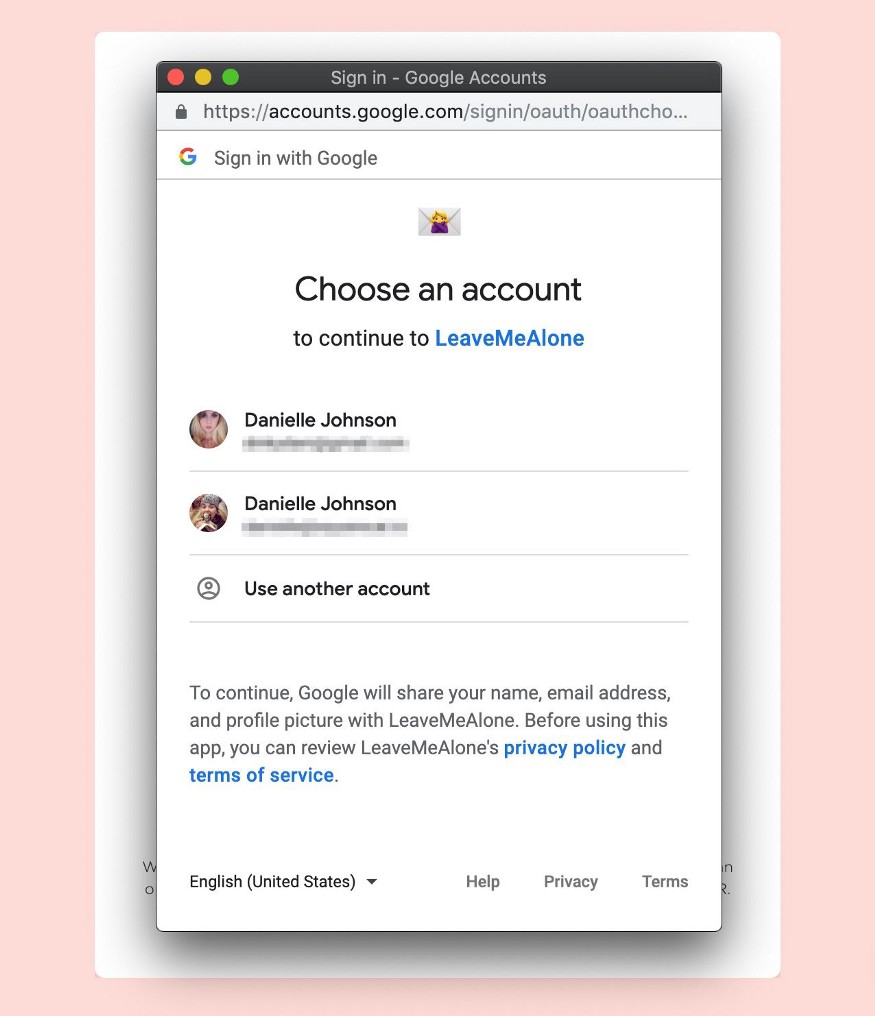


Рисунок 3.6 – Поп-ап авторизации Firebase провайдера Google

Таким образом, из общей структуры убираются те составляющие, к которым у пользователя доступа нет. Для разрабатываемого приложения эта фукнция также полезна, ведь оптимизация разметки позволяет сократить время на обработку и отрисовку страницы.

Алгоритм аутентификации и авторизации представлен на рисунке 3.7.

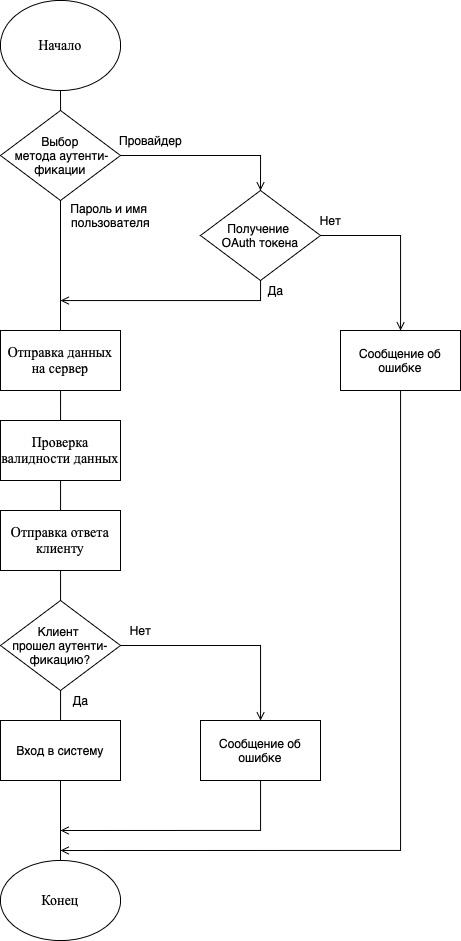


Рисунок 3.7 – Алгоритм аутентификации и авторизации

* + 1. Отложенная загрузка

Современное приложение может состоять из десятков компонентов, на инициализацию каждого из которых могут быть потрачены драгоценные ресурсы компьютера и время. Если приложение использует клиент-серверную архитектуру, то необходимые компоненты могут подгружаться по мере необходимости посредствам технологии AJAX, однако в случае разрабатываемого приложения, мы не имеем такой возможности, так как не контролируем серверную часть.

Однако, с появлением технологии HTML import мы можем динамически подгружать документы в другие документы, а каждый компонент и является HTML документом. В совокупности с возможностью обновления компонентов React, мы можем добиться необходимой функциональности: компоненты будут присутствовать в разметке изначально, однако их апгрейд будет производиться при импорте документов в момент, когда появляется необходимость доступа к этим компонентам. К примеру, таким образом можно организовать отложенную загрузку страниц приложения, ведь при первом запуске нам не нужны все страницы.

Ещё одной ресурсоёмкой стороной приложения могут являться графические ресурсы. С повышением разрешения дисплеев современных устройств, дизайнеры прибегают к изображениям всё большего и большего разрешения, что может негативно отразиться на владельцах более слабых устройств. Уже сейчас в стандарте присутствует возможность указывать различные версии изображения для дисплеев разного разрешения, однако и эта функция решает проблему лишь на половину. Что, если попробовать оптимизировать первичную загрузку приложения, путём оптимизации картинок? А полноценное изображение можно подгрузить в фоне асинхронно, тем самым значительно ускорив первичную инициализацию приложения (см. рисунок 3.8).

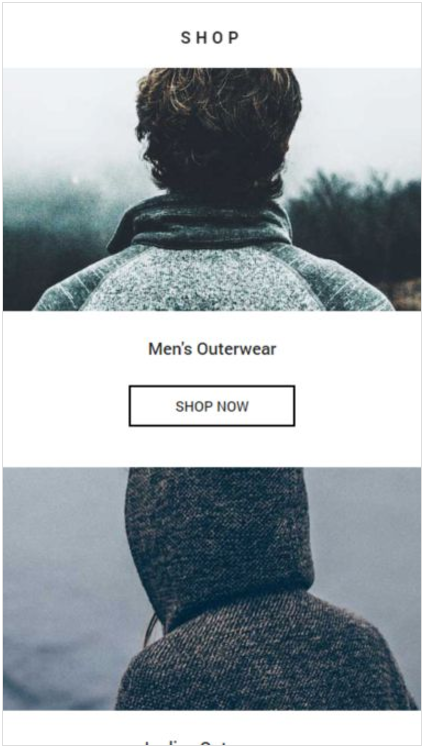
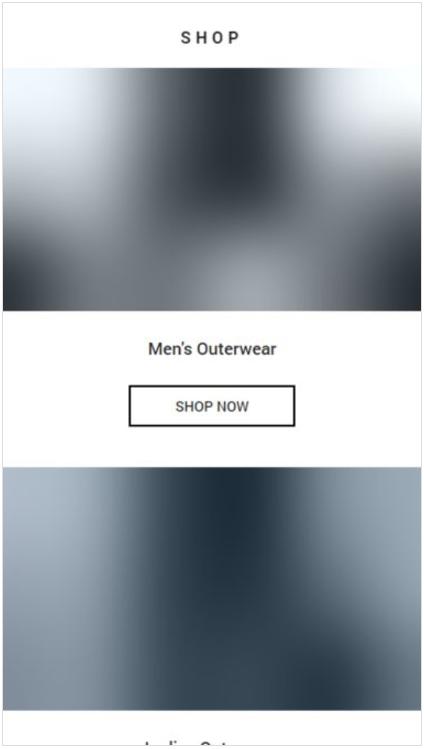


Рисунок 3.8 – Отложенная загрузка изображений

Изначально (в момент первичной отрисовки) мы можем загрузить значительно уменьшенную версию картинки, увеличив её до размеров исходного изображения. Для пущего ускорения мы можем закодировать эту версию при помощи base64 и вставить непосредственно в разметку документа, тем самым сократив время на его обработку. Далее, после инициализации компонента, мы выставляем внутреннему тэгу <img> путь к полноразмерному файлу, тем самым активируя его загрузку. Всё остальное браузер сделает самостоятельно. Нам остаётся лишь сделать красивый переход между оптимизированной и полноразмерной версией изображения.

Последующая загрузка полноразмерной версии будет практически мгновенная, благодаря использованию кэширующего service-worker.

* + 1. Кэширование

Нет необходимости каждый раз посылать запрос на сервер для получения данных, который однажды уже были отправлены. Для решения этой проблемы в брузеры когда-то была добавлена функция кэширования. Она работает полностью автономно и практически не может контролироваться программистом. Для решения этого недоразумения была создана библиотека sw-precache, которая создает service worker (SW) для перехвата запросов пользователя к удалённому серверу, позволяя подменять или возвращать закэшированные данные. Алгоритм установки ServiceWorker изображен на рисунке 3.9.



Рисунок 3.9 – Алгоритм установки ServiceWorker

Библиотека позволяет не просто гибко настраивать своё поведение, но даже позволяет писать собственные стратегии кэширования и доступа к контенту:

function serveFromCacheOrNetwork(request) {

if (request.headers.has('Authorization') &&

!request.headers.has('X-Cache-Only')) {

return global.toolbox.networkOnly(request);

}

if (request.headers.get('X-Cache-Only') === 'true') {

return global.toolbox.cacheOnly(request)

.then(function(response) {

if (response) {

return response;

}

return new Response(null, {

status: 204,

statusText: 'No cached content available.'

});

});

}

}

Здесь мы проверяем, присутствуют ли в ответе от сервера определённые заголовки, которые говорят нам, что контент должен возвращаться не из кэша и, в зависимости от результата, возвращаем ответ. Так как service worker выступает в роли прокси, мы можем не только обрабатывать ответ сервера, а даже полностью подменять тело ответа на необходимые нам данные. К примеру, если приложение находится в оффлайн-режиме и пользователь загружает профиль, то мы можем попытаться получить изображение из кэша, но, если его нет, можем вернуть заглушку.

var DEFAULT\_PROFILE\_IMAGE\_URL =

'images/touch/homescreen96.png';

function profileImageRequest(request) {

return global.toolbox.networkFirst(request).catch(function() {

return global.toolbox.cacheOnly(new Request(

DEFAULT\_PROFILE\_IMAGE\_URL));

});

}

global.toolbox.precache([DEFAULT\_PROFILE\_IMAGE\_URL]);

global.toolbox.router.get('/(.+)/firebasestorage/(.\*)', profileImageRequest);

В некоторых случаях возникает необходимость контролировать кэш и очищать его по требованию. Мы не можем напрямую общаться из кода к методам SW, так как они выполняются в отдельном потоке обозревателя (что и позволяет им выполнять большие объёмы кода, не влияя на UX). Для обмена сообщениями с service worker существует специальная технология MessageChannel.

Канал создаётся автоматически для всех объектов типа SW и позволяет отправлять и обрабатывать сообщения следующим образом:

if (navigator.serviceWorker && navigator.serviceWorker.controller) {

navigator.serviceWorker.controller.postMessage('clear-cached-user-data');

}

global.addEventListener('message', function(event) {

if (event.data === 'clear-cached-user-data') {

global.caches.open(global.toolbox.options.cacheName)

.then(function(cache) {

cache.keys().then(function(requests) {

return requests.filter(function(request) {

return request.url.indexOf(

USER\_DATA\_URL\_SUBSTRING) !== -1;

});

}).then(function(userDataRequests) {

userDataRequests.forEach(function(userDataRequest) {

cache.delete(userDataRequest);

});

});

});

}

});

* + 1. Правила доступа к базе данных

Так как разрабатываемой приложение не имеет серверной части, остро встаёт вопрос о разграничении доступа пользователей к отдельным частям и функциям системы. Приложение работает полностью в клиентском окружении и реализует свои функции посредством javascript-сценариев, которые в открытом виде передаются с сервера на клиент, а это значит, что не составляет труда получить доступ к данным программы, а также изменить её поведение и, тем самым, поставить безопасность пользователей приложения под угрозу.

К счастью, Firebase позволяет задавать правила доступа к различным сегментам своей базы данных. Правила эти описаны в объекте вида JSON, что открывает широкие возможности по гибкой настройке действительно защищённой системы. Таким образом получается мощный инструмент по разграничению прав доступа пользователей к функциям системы на основе ролей.

Структура доступа будет следующей: в специальный раздел базы данных будут занесены возможные роли пользователей, внутри каждой из которых будет список разделов базы данных с разрешением на конкретное действие.

Permissions: {

user: {

users: {

own: { // can read, update, delete own data

r: true,

u: true,

d: true

},

other: {} // can't read/write (not owned) data;

},

/\*

\*/

messages: {

c: true, // can create messages

},

message: {

own: { // can update & delete own messages

r: true,

u: true,

d: true

},

other: { // can read other messages

r: true

}

}

}

}

R,U,D, соответственно, сокращения от Read, Update, Delete. После этого мы присваиваем определённую роль конкретному пользователю, тем самым определяя рамки, в которых ему разрешено пользоваться системой. Для повышения безопасности, можно вовсе запретить доступ к этой таблице извне: данные можно будет менять лишь из консоли Firebase.

Приступим к организации подобной системы доступа. Основой будет функция проверки доступа:

isAllowed(obj, entity, permission, ownResources) {

auth != null && auth.uid &&

root.users[auth.uid].role &&

(

auth.uid !== obj.owner

?

root.permissions[root.users[auth.uid].role][entity].all === true ||

root.permissions[root.users[auth.uid].role][entity][permission] === true ||

root.permissions[root.users[auth.uid].role][entity]

.other[permission] === true

:

(!ownResources ? root.permissions[root.users[auth.uid].role][entity].own[permission] === true : false)

)

}

По сути своей, это лишь большое условное выражение с переменными параметрами. В аргументы этой функции передаётся объект контекста, запрашиваемый раздел для доступа, тип операции доступа и булево значение, определяющее получает ли пользователь доступ к собственным ресурсам. Далее нам необходимо, используя указанную фукнцию, определить разделы базы данных и проверку доступа к ним:

path / {

read() { false }

write() { false }

}

type Teams {

create() { isAllowed(this, 'teams', 'c', false) }

}

type Team {

read() { isAllowed(this, 'team', 'r', false) }

update() { isAllowed(this, 'team', 'u', false) }

delete() { isAllowed(this, 'team', 'd', false) }

}

path /projects is Teams {}

path /projects/{$pid} is Team {}

В первую очередь запретим доступ ко всем неопределённым разделам (запрещено всё, что не разрешено). После этого мы можем определить модели разделов базы данных. Внутри модели мы задаём метод определения прав доступа на выполнение конкретной операции: вызываем написанную ранее функцию с параметрами, определяющими операцию, к которой мы запрашиваем доступ в данный момент. В конце мы привязываем созданную нами модель данных к разделу базы данных.

* + 1. Транслирование видео- и аудиоматериалов

Пользователь с ролью «Преподаватель» имеет возможность проводить онлайн-трансляции своих занятий. Доступ к трансляции могут иметь только пользователи с ролью «Студент», имеющие подписку на текущий курс. Алгоритм транслирования представлен на рисунке 3.10.

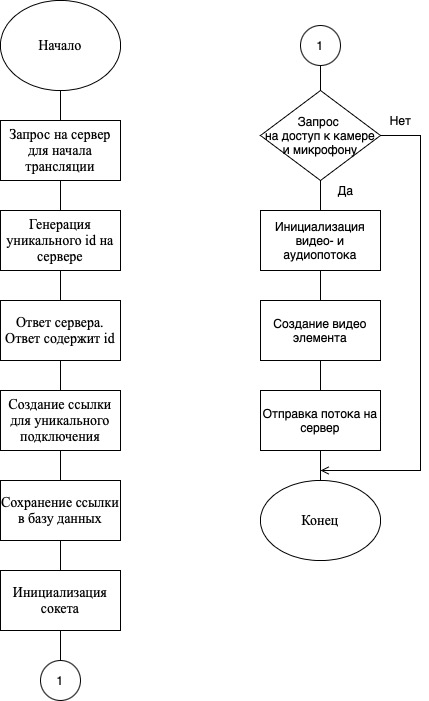


Рисунок 3.10 – Алгоритм трансляции видео и аудио

После открытия доступа к микрофону и камере устройства начинается передача потока на сервер, откуда подключившиеся пользователи смогут его получить. Данная функция реализуется при помощи peer-to-peer соединеия, что в дальнейшем позволит расширить функционал приложения внедрением видео связи.

* + 1. Захват изображения с камеры

Пользователь имеет возможность загрузить фотографию в свой профиль, чтобы позволить другим пользователям отличать его. Одним способом получения изображение является его загрузка из памяти устройства. Вторым способом загрузки является получения изображения непосредственно с камеры устройства средствами HTML5.

Функция navigator.getUserMedia позволяет получить объект, открывающий доступ к медиаресурсам клиента.

navigator.getUserMedia(

{

video: true

},

function(stream){

video.src = window.URL.createObjectURL(stream);

video.play();

},

function(err){

console.error(err);

}

);

Таким образом, если пользователь даст разрешение, приложение сможет получить доступ к камере устройства и сможем использовать её видеопоток по своему усмотрению.

Однако, теперь возникает необходимость получить кадр из этого видеопотока для дальнейшего использования:

function takeSnapshot(){

var hidden\_canvas = document.querySelector('canvas'),

video = document.querySelector('video.camera\_stream'),

image = document.querySelector('img.photo'),

width = video.videoWidth,

height = video.videoHeight,

context = hidden\_canvas.getContext('2d');

hidden\_canvas.width = width;

hidden\_canvas.height = height;

context.drawImage(video, 0, 0, width, height);

var imageDataURL = hidden\_canvas.toDataURL('image/png');

}

Теперь переменная imageDataURL содержит кадр, закодированный в Base64, который можно смело использовать на странице, либо загрузить на сервера Firebase следующим образом:

var storageRef = firebase.storage().ref();

storageRef.child("gmlta/profile.jpg").put(data).then(function(snapshot) {

// ...

});

После этого картинка может быть встроена в страницу пользователя как фотография профиля:

storageRef.child('gmlta/profile.jpg').getDownloadURL().then(function(url) {

var xhr = new XMLHttpRequest();

xhr.responseType = 'blob';

xhr.onload = function(event) {

var blob = xhr.response;

};

xhr.open('GET', url);

xhr.send();

var img = document.getElementById('user-photo');

img.src = url;

}).catch(function(error) {

// Handle any errors

});

* + 1. Организация общения между пользователями

Чат между участниками и организаторами также реализован при помощи Firebase (см. рисунок 3.11). Так как клиент подписывается на обновления базы данных в реальном времени, то мы моментально будем получать новые сообщения, присылаемые другими клиентами.

database.child("messages").on('value', function(snapshot) {

// ...

});

Функция будет вызвана при каждом изменении раздела messages. За отображение сообщений отвечает связка компонента <firebase-document> и шаблона с типом dom-repeat.

<template is="dom-repeat" items="messages">

<message-card data=[[item]] />

</template>

Благодаря разграничению прав доступа, мы организуем безопасный обмен сообщениями, ведь пользователи других каналов не смогут получить доступ к текущему каналу.

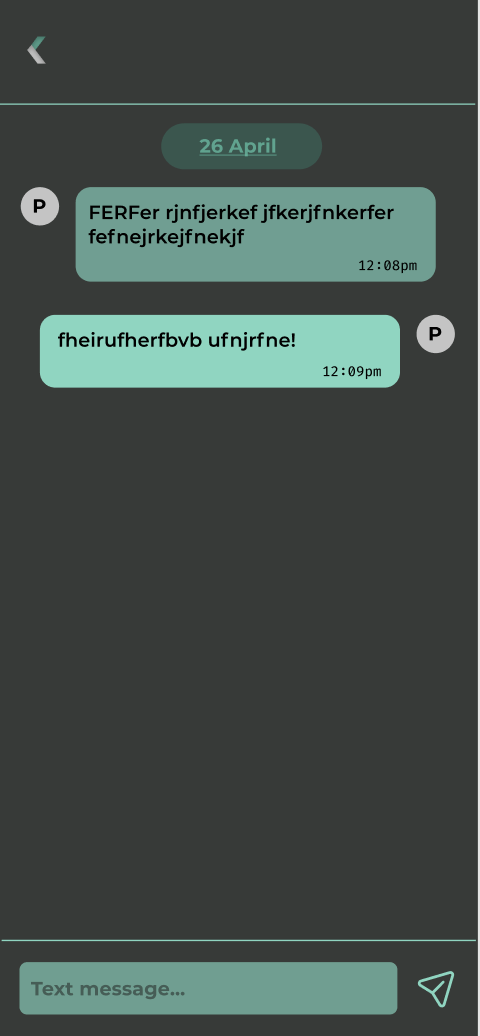


Рисунок 3.11 – Страница с компонентом обмена сообщениями

* 1. Развертывание программного средства

После завершения проектирования всех компонентов программного средства появляется вопрос о развертывании всей системы. Необходимо составить описание аппаратнопрограммных комплексов, которые понадобятся для обеспечения функционирования распределенного приложения. Для этого логичным выглядит составление диаграммы развертывания. Данная диаграмма представлена на рисунке 3.12. Она отражает следующие особенности:

* на узле оконечного устройства в качестве среды выполнения перечислен список браузеров, через которое можно запустить клиентскую часть приложения;
* для серверной части программного средства и базы данных показано их развертывание с помощью облачных сервисов;
* предполагается, что пользовательское оконечное устройство значительно удалено от серверов программной системы, доступ осуществляется через сеть Интернет, что и упрощенно показано на диаграмме.

Для хостинга системы предполагается использование сервисов Firebase Cloud Hosting. Необходимо зарегистрироваться на сайте системы и создать новый проект. Система предоставит учётные данные для работы проекта:

* ProjectURL;
* API key;
* DatabaseURL;
* DB key;
* Cloud messaging key.

Эти параметры необходимо внести в конфигурационный файл проекта, тем самым обеспечив связь проекта с серверами Firebase.



Рисунок 3.12 – Диаграмма развертывания программного средства

1. ТЕСТИРОВАНИЕ И ПРОВЕРКА РАБОТОСПОСОБНОСТИ ПРОГРАММНОГО СРЕДСТВА

Тестирование программного обеспечения – процесс анализа программного средства и сопутствующей документации с целью выявления дефектов и повышения качества продукта [10]. Вот уже несколько десятков лет его стабильно включают в планы разработки как одна из основных работ, причем выполняемая практически на всех этапах проектов. Важность своевременного выявления дефектов подчеркивается выявленной эмпирически зависимостью между временем допущения ошибки и стоимостью ее исправления: график данной функции круто возрастает.

Для проверки правильности работы программного модуля, реализации им всех функциональных требований и корректности совместной работы с другими модулями приложения был выбран метод функционального тестирования и разработан набор тест-кейсов. Функциональное тестирование является видом тестирования программного обеспечения, при котором тестируется каждая функция программы и делается вывод о правильности ее реализации. Проверка на правильность осуществляется в некоторых выбранных точках области определения функции, если проверить всю область определения невозможно. С помощью функционального тестирования было проверено отсутствие ошибок в работе функций или отсутствие реализации функций, ошибки интерфейса и ошибки внешних структур данных.

Помимо юнит-тестирования в данном разделе проведем динамическое ручное тестирование и автоматизированное тестирование. Его целью является подтверждение соответствия работы программного средства установленным в начале разработки требованиям. Успешное выполнение приведенных в данном разделе тестовых случаев должно подтвердить работоспособность программного средства в основных сценариях использования, а также устойчивость к неверным входным данным.

В рамках дипломного проектирования был составлен тестовый сценарий, состоящий из набора тест-кейсов, реализованных программно и примененных на различных этапах разработки, из набора автоматизированных тест-кейсов, реализованных при помощи технологии Selenium Webdriver, а также из набора мануальных тестов, реализованных непосредственно разработчиками в процессе проектирования.

Результаты проведенного тестирования программного средства представлены в виде таблиц 4.1 – 4.12.

Таблица 4.1 – Тест-кейс неудачной регистрации

|  |  |  |
| --- | --- | --- |
| Тест | Ожидаемый результат | Результат |
| **Попытка регистрации с неподтвержденным паролем**   1. Открыть приложение, перейдя по адресу http://0.0.0.0:3000 2. Проверить элементы на странице 3. Кликнуть по кнопке «Menu» 4. Проверить элементы на странице 5. Кликнуть по кнопке «Sign Up» 6. Проверить элементы на странице 7. Ввести значение «test\_user@icloud.com» в поле «Enter Email» 8. Ввести значение «qwerty123» в поле «Enter password» 9. Ввести значение «Qwerty123» в поле «Confirm password» 10. Кликнуть по кнопке «sign up» | 1. Открывается главная страница приложения. 2. Отображаются следующие элементы:  * кнопка «Menu», открывающая навигационную панель.  1. Открывается навигационная панель. 2. Отображаются следующие элементы:  * кнопка «</Courses>»; * кнопка «</Log in>», ведущая на страницу авторизации; * кнопка «</Sign up>», ведущая на страницу регистрации; * кнопка «</Contact us>»;  1. Открывается страница регистрации. 2. Отображаются следующие Элементы:  * поле «Enter email»; * поле «Enter password»; * поле «Confirm password»; * кнопка «sign up»; * список провайдеров.  1. Значение в поле «Enter email» введено. 2. Значение в поле «Enter password» введено. 3. Значение в поле «Confirm password» введено. 4. Появляется сообщение о несоответствии паролей «Sign up failed. Please check input data». | Успех |

Таблица 4.2 – Тест-кейс неудачной авторизации

|  |  |  |
| --- | --- | --- |
| Тест | Ожидаемый результат | Результат |
| **Авторизация несуществующего пользователя**   1. Открыть страницу авторизации 2. Проверить элементы на странице 3. Ввести значение «test\_user» в поле «Login» 4. Ввести значение «qwerty123» в поле «Password» 5. Кликнуть по кнопке «Log in» | 1. Открывается страница авторизации. 2. Отображаются следующие элементы:  * поле «Login»; * поле «Password»; * кнопка «Log in», подтверждающая отправку формы авторизации.  1. Значение в поле «Login» введено. 2. Значение в поле «Password» введено. 3. Появляется сообщение «User was not found!» | Успех |

Таблица 4.3 – Тест-кейс успешно пройденной регистрации

|  |  |  |
| --- | --- | --- |
| Тест | Ожидаемый результат | Результат |
| **Регистрация с помощью провайдера**   1. Открыть страницу регистрации 2. Проверить элементы на странице 3. Кликнуть по кнопке «Googlr» 4. Выбрать профиль для авторизации 5. Кликнуть по кнопке «Confirm» | 1. Открывается страница регистрации. 2. Отображаются следующие элементы:  * поле «Enter email»; * поле «Enter password»; * поле «Confirm password»; * кнопка «sign up»; * список провайдеров.  1. Появление поп-ап компонента google со списком авторизированных профилей. 2. Появление поля с кнопкой «Confirm». 3. Сообщение «You were successfully registered!». | Успех |

Таблица 4.4 – Тест-кейс успешно пройденной авторизации

|  |  |  |
| --- | --- | --- |
| Тест | Ожидаемый результат | Результат |
| **Авторизация существующего пользователя с верными данными**   1. Открыть страницу авторизации 2. Ввести значение «test\_user» в поле «Username» 3. Ввести значение «qwerty123» в поле «Password» 4. Кликнуть по кнопке «Register» | 1. Открывается страница авторизации. 2. Значение в поле «Username» введено. 3. Значение в поле «Password» введено. 4. Появляется сообщение об успешной авторизации «Welcome, test\_user!». | Успех |

Таблица 4.5 – Тест-кейс редактирование профиля пользователя

|  |  |  |
| --- | --- | --- |
| Тест | Ожидаемый результат | Результат |
| **Редактирование профиля**   1. Открыть страницу «Profile» 2. Проверить наличие элементов на странице 3. Кликнуть по кнопке «Edit» 4. Ввести в поле «Username» значение «Name» 5. Кликнуть по кнопке «Save» | 1. Открывается страница интеграции с банком. 2. Отображаются следующие элементы:  * поле «Login»; * поле «Password»; * кнопка «Integrate»; * выпадающий список банков.  1. Кнопка «Save» активна. 2. Значение в поле «Username» введено. 3. Появляется сообщение о успешном редактировании информации. Кнопка «Save» не отображается. | Успех |

Таблица 4.6 – Тест-кейс создания нового курса

|  |  |  |
| --- | --- | --- |
| Тест | Ожидаемый результат | Результат |
| **Создание курса**   1. Открыть главную страницу администратора 2. Проверить элементы на странице 3. Кликнуть по кнопке «Courses» 4. Проверить наличие элементов 5. Кликнуть по кнопке «Add course» 6. Проверить наличие элементов 7. Ввести в поле «CourseName» значение «test\_name» 8. Ввести в поле «Tutuor» значение «tutor@icloud.com» 9. Ввести в поле «Period» значение «03/21-05/12» 10. Ввести в поле «Description» значение «descrption»   Ввести в поле «Sheduel» значение «Mn: 3:20pm, Tue: 1:12pm»   1. Кликнуть по кнопке «Add» | 1. Открывается главная страница администратора. 2. Отображаются следующие элементы:  * кнопка «Courses»; * кнопка «Students»; * кнопка «Professors»;  1. Открывается страница «Courses». 2. Отображаются следующие элементы:  * список курсов; * кнопка «Add course».  1. Появляется страница «Add course». 2. Отображаются следующие элементы:  * поле «CourseName»; * поле «Tutor»; * поле «Period»; * поле «Sheduel»; * кнопка «Add».  1. Значение в поле «CourseName» введено. 2. Значение в поле «Tutor» введено. 3. Значение в поле «Period» введено. 4. Значение в поле «Sheduel» введено. 5. Страница «Add course» закрылась. Появился новая запись в списке курсов. | Успех |

Таблица 4.7 – Тест-кейс наличия элементов личного кабинета пользователя

|  |  |  |
| --- | --- | --- |
| Тест | Ожидаемый результат | Результат |
| **Наличие элементов детализированной страница курса для пользователя «Преподаватель»**   1. Открыть приложение и авторизоваться как пользователь «Преподаватель» 2. Проверить наличие элементов на странице 3. Выбрать любой курс из представленного списка 4. Проверить наличие элементов на странице | 1. Открывается главная страница личного кабинета пользователя. 2. Отображаются следующие элементы:  * кнопка «Menu»; * список курсов.  1. Открытие детализированной страницы курса. 2. Отображаются следующие элементы:  * кнопка «Back», позволяет вернуться на предыдущую страницу; * кнопка «Add file», позволяет загрузить новый текстовый файл; * кнопка «Messenger», открывает беседу выбранного курса; * кнопка «Start record», позволяет включить онлайн-трансляцию занятия; * кнопка «Save record», позволяет завершить трансляцию и сохранить ее в хранилище; * кнопка «Delete», позволяет удалить файл из списка материалов; * список документов; * лейбл «CourseName», отображает название выбранного курса; * элемент видео отображения. | Успех |

Таблица 4.8 – Тест-кейс наличия элементов личного кабинета пользователя

|  |  |  |
| --- | --- | --- |
| Тест | Ожидаемый результат | Результат |
| **Наличие элементов детализированной страница курса для пользователя «Студент»**   1. Открыть приложение и авторизоваться как пользователь «Студент» 2. Проверить наличие элементов на странице 3. Выбрать любой курс из представленного списка 4. Проверить наличие элементов на странице | 1. Открывается главная страница личного кабинета пользователя. 2. Отображаются следующие элементы:  * кнопка «Menu»; * список курсов.  1. Открытие детализированной страницы курса. 2. Отображаются следующие элементы:  * кнопка «Back», позволяет вернуться на предыдущую страницу; * кнопка «Messenger», открывает беседу выбранного курса; * кнопка «Start video», позволяет включить просмотр онлайн-трансляции занятия; * кнопка «Stop record», позволяет завершить просмотр трансляции; * список документов, при нажатии на любой элемент списка, начинается скачивание файла; * лейбл «CourseName», отображает название выбранного курса; * элемент видео отображения. | Успех |

Таблица 4.9 – Тест-кейс онлайн-трансляции

|  |  |  |
| --- | --- | --- |
| Тест | Ожидаемый результат | Результат |
| **Совершение онлайн-трансляции**   1. Открыть приложение и авторизоваться как пользователь «Преподаватель» 2. Проверить наличие элементов на странице 3. Выбрать любой курс из представленного списка 4. Проверить наличие элементов на странице 5. Кликнуть по кнопке «Start record» 6. Кликнуть по кнопке «Разрешить» | 1. Открывается главная страница личного кабинета пользователя. 2. Отображаются следующие элементы:  * кнопка «Menu»; * список курсов.  1. Открытие детализированной страницы курса. 2. Отображаются следующие элементы:  * кнопка «Back», позволяет вернуться на предыдущую страницу; * кнопка «Add file», позволяет загрузить новый текстовый файл; * кнопка «Messenger», открывает беседу выбранного курса; * кнопка «Start record», позволяет включить онлайн-трансляцию занятия; * кнопка «Save record», позволяет завершить трансляцию и сохранить ее в хранилище; * кнопка «Delete», позволяет удалить файл из списка материалов; * список документов; * лейбл «CourseName», отображает название выбранного курса; * элемент видео отображения.  1. Появляется модальное окно браузера с запросом на доступ к камере и микрофону. 2. Появляется изображение с камеры в элементе отображения видео. Сообщение «Recording». | Успех |

Таблица 4.10 – Тест-кейс сохранения онлайн-трансляции

|  |  |  |
| --- | --- | --- |
| Тест | Ожидаемый результат | Результат |
| **Сохранение онлайн-трансляции**   1. Открыть приложение и авторизоваться как пользователь «Преподаватель» 2. Проверить наличие элементов на странице 3. Выбрать любой курс из представленного списка 4. Проверить наличие элементов на странице 5. Кликнуть по кнопке «Start record» 6. Кликнуть по кнопке «Разрешить» 7. Кликнуть по кнопке «Save record» | 1. Открывается главная страница личного кабинета пользователя. 2. Отображаются следующие элементы:  * кнопка «Menu»; * список курсов.  1. Открытие детализированной страницы курса. 2. Отображаются следующие элементы:  * кнопка «Back», позволяет вернуться на предыдущую страницу; * кнопка «Add file», позволяет загрузить новый текстовый файл; * кнопка «Messenger», открывает беседу выбранного курса; * кнопка «Start record», позволяет включить онлайн-трансляцию занятия; * кнопка «Save record», позволяет завершить трансляцию и сохранить ее в хранилище; * кнопка «Delete», позволяет удалить файл из списка материалов; * список документов; * лейбл «CourseName», отображает название выбранного курса; * элемент видео отображения.  1. Появляется модальное окно браузера с запросом на доступ к камере и микрофону. 2. Появляется изображение с камеры в элементе отображения видео. Сообщение «Recording». 3. Появляется сообщение «Stop recording. Saving…». 4. Список документов курса обновляется. Появляется новая запись. | Успех |

Таблица 4.11 – Тест-кейс отправки сообщения

|  |  |  |
| --- | --- | --- |
| Тест | Ожидаемый результат | Результат |
| **Отправка сообщения**   1. Открыть приложение и авторизоваться как пользователь «Студент» 2. Проверить наличие элементов на странице 3. Выбрать любой курс из представленного списка 4. Проверить наличие элементов на странице 5. Кликнуть по кнопке «Messenger» 6. Проверить наличие элементов на странице 7. Ввести в поле «Input-message» значение «new\_message» 8. Кликнуть по клавише «Send» | 1. Открывается главная страница личного кабинета пользователя. 2. Отображаются следующие элементы:  * кнопка «Menu»; * список курсов.  1. Открытие детализированной страницы курса. 2. Отображаются следующие элементы:  * кнопка «Back», позволяет вернуться на предыдущую страницу; * кнопка «Messenger», открывает беседу выбранного курса; * кнопка «Start video», позволяет включить просмотр онлайн-трансляции занятия; * кнопка «Stop record», позволяет завершить просмотр трансляции; * список документов, при нажатии на любой элемент списка, начинается скачивание файла; * лейбл «CourseName», отображает название выбранного курса; * элемент видео отображения.  1. Открытие страницы мессенджера 2. Отображаются следующие элементы:  * список отправленных сообщений; * кнопка «Send»; * поле ввода «Input-message»;  1. Значение «new\_message» в поле ввода «Input-message». 2. Отображение сообщения в списке отправленных сообщений. Поле ввода «Input-message» очищено. | Успех |

Таблица 4.12 – Тест-кейс добавления фотографии в профиль пользователя

|  |  |  |
| --- | --- | --- |
| Тест | Ожидаемый результат | Результат |
| **Добавление фотографии в профиль пользователя**   1. Открыть приложение и авторизоваться как пользователь «Студент» 2. Проверить наличие элементов на странице 3. Кликнуть по кнопке «Menu» 4. Проверить наличие элементов в выпадающем меню 5. Кликнуть по кнопке «Profile» 6. Проверить наличие элементов на странице 7. Кликнуть по кнопке «Edit» 8. Кликнуть по иконке изображения 9. Кликнуть по кнопке «Разрешить» 10. Выбрать фото из галереи 11. Кликнуть по кнопке «Save» | 1. Открывается главная страница личного кабинета пользователя. 2. Отображаются следующие элементы:  * кнопка «Menu»; * список курсов.  1. Открывается выпадающее меню. 2. Отображаются следующие элементы:  * кнопка «Courses»; * кнопка «Profile»; * кнопка «Log out»; * кнопка «Contact us».  1. Открывается страница с информацией пользователя. 2. Отображаются следующие элементы:  * кнопка «Edit»; * поле «Username»; * поле «Email»; * поле «Group»; * кнопка «Menu»; * поле «Course»; * иконка «Photo».  1. Появляется кнопка «Save». 2. Появляется модальное окно браузера с запросом на доступ ко внутренней памяти телефона. 3. Открытие галлереи. 4. Загрузка фотографии в приложение. 5. Иконка содержит новое изображение; кнопка «Save» пропадает. | Успех |

В результате итогового тестирования серьезных дефектов выявлено не было. Все тестовые случаи были пройдены успешно. Фактический результат совпал с ожидаемым результатом в каждом из тестовых случаев.

1. РУКОВОДСТВО ПО УСТАНОВКЕ И ИСПОЛЬЗОВАНИЮ

В данном разделе приведены основные сведения по работе с программным средством. Приложение данного дипломного проекта (а точнее, его клиентская часть) не требует установки и настройки на конечных устройствах пользователя, поскольку представляет собой веб-приложение.

Для хостинга системы предполагается использование сервисов Firebase Cloud Hosting. Необходимо зарегистрироваться на сайте системы и создать новый проект. Система предоставит учётные данные для работы проекта:

* ProjectURL;
* API key;
* DatabaseURL;
* DB key.

Эти параметры необходимо внести в конфигурационный файл проекта,

тем самым обеспечив связь проекта с серверами Firebase.

Для запуска сценариев сборки проекта используется окружение Node. Необходимо установить на рабочей машине Node [11]. После этого при помощи командной строки менеджера пакетов Node (node package manager, NPM) и запустить в корне проекта команду npm install, которая выполнит установку зависимостей Node. После этого необходимо установить в систему менеджер front-end зависимостей bower командой npm install –g bower, которая установит собственные зависимости.

После этого необходимо установить систему сборки проектов Gulp командой install –g gulp, которая позволит выполнять сценарии сборки проекта.

После установки системы сборки пользователь может запустить проект несколькими способами:

* локальный старт командой gulp serve;
* удалённая установка на сервера Firebase командой gulp deploy.

В обоих случаях после установки будет открыто окно браузера с главной страницей приложения (см. рисунок 5.1):



Рисунок 5.1 – Приветственная страница сайта

На главном экране пользователь получает доступ к информации об проекте и к списку доступных курсов, который кэшируется и может быть открыт даже если отсутствует доступ в интернет (см. рисунок 5.2).



Рисунок 5.2 – Приветственная страница сайта (каталог курсов)

В приложении используется паттерн Navigation drawer: навигационное меню, выдвигаемое сверху (см. рисунок 5.3). Из него пользователь может перейти в любой другой раздел приложения. Некоторые элементы могут быть недоступны из-за ограничений доступа пользователя (см. рисунок 5.4).



Рисунок 5.3 – Навигационное меню неавторизированного пользователя

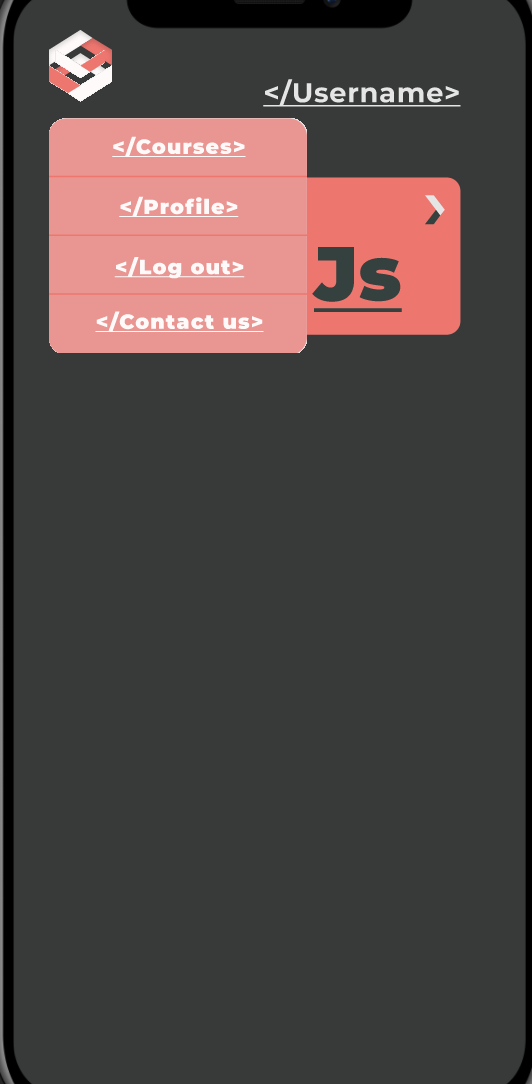


Рисунок 5.4 – Навигационное меню пользователя «Преподаватель»

В случае, если пользователь еще не зарегистрирован в системе, ему следует кликнуть по кнопке навигационного меню «Sign Up» и перейти на страницу регистрации, где ему предстоит ввести имя пользователя, пароль и адрес электронной почты, после чего, после подтверждения пароля и в случае удачной регистрации, он вернется на приветственную страницу. Также пользователь имеет возможность авторизоваться с помощью сторонних провайдеров Скриншот страницы регистрации приведен на рисунке 5.5.

После регистрации пользователю необходимо авторизоваться, перейдя на страницу авторизации. Ее изображение представлено на рисунке 5.6. Там пользователю необходимо будет ввести имя пользователя, указанное при регистрации и пароль. После этого в случае удачной авторизации пользователь попадет на главную страницу своего личного кабинета.

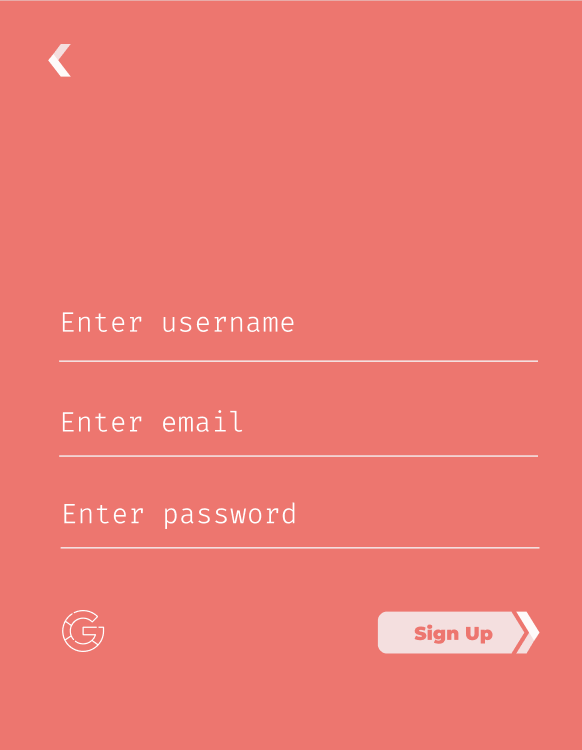


Рисунок 5.5 – Страница регистрации

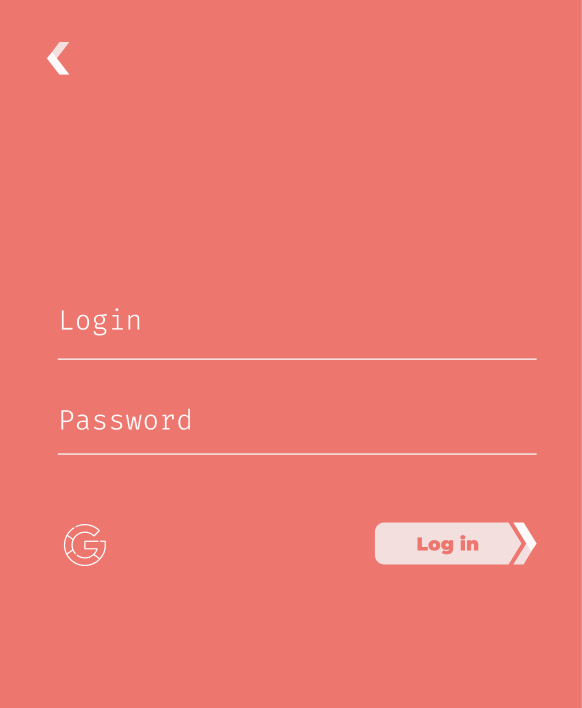


Рисунок 5.6 – Страница авторизации

На главной странице личного кабинета отображается его никнейм и список его активных курсов. При нажатии на кнопку «Username» будет произведён переход на страницу с личной информацией пользователя, где можно будет настроить свой профиль: выбрать фотографию и имя пользователя, ввести личные и контактные данные, ссылки в соцсетях и другие параметры, относящиеся к конкретному пользователю(см. рисунок 5.7). Информация доступна к редактированию только после нажатия кнопки «Edit». Добавление фотографии пользователя происходит при нажатии на иконку профиля (см. рисунок 5.7), загрузка фотографии происходит из внутренней памяти телефона, а затем загружается в облачное хранилище Firebase. Для сохранения отредактированной информации нажать на кнопку «Save».



Рисунок 5.7 – Иконка для загрузки фотографии профиля

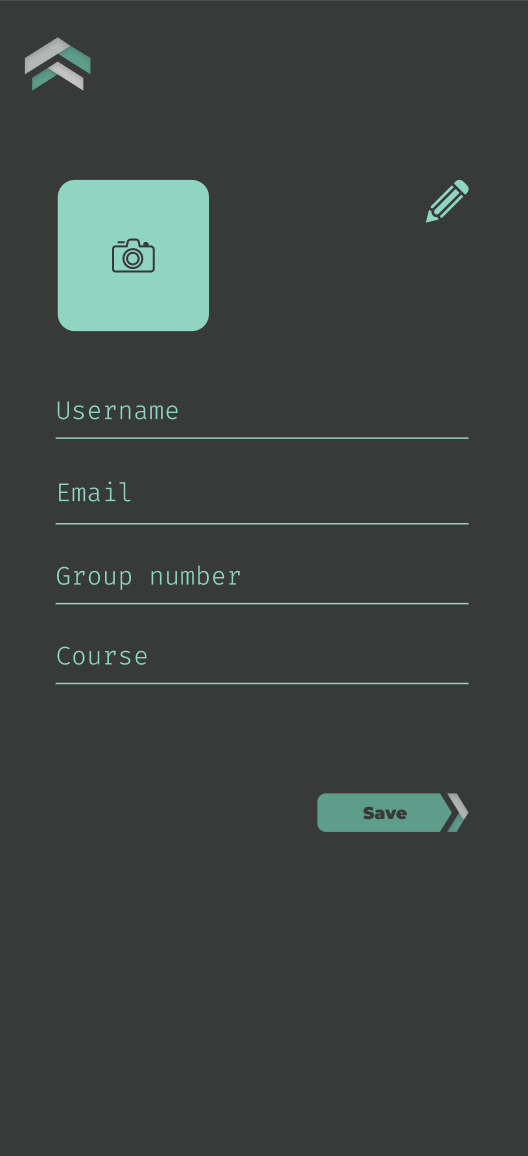


Рисунок 5.8 – Страница профиля пользователя (отображение и редактирование информации)

При открытии навигационного меню и нажатии кнопки «Profile» происходит возврат на главную страницу личного кабинета со списком активных курсов пользователя. При нажатии на любой из активных курсов происходит переход в классную комнату курса. Для пользователей различных ролей данная страница выглядит по-разному. При авторизации пользователя с ролью «Преподаватель» доступны следующие функции: удаление файлов, добавление нового файла, просмотр, онлайн-трансляция, сохранение записи и переход в основной чат курса (см. рисунок 5.9). Для того чтобы начать онлайн трансляцию следует нажать на кнопку «Start recording» в правом нижнем углу элемента отображения видео, затем остановить трансляцию с помощью «Stop&Save». Для пользователей с ролью «Студент» доступны просмотр файла и онлайн-трансляции, а также переход в основной чат (см. рисунок 5.10). В обоих слуаях кнопка «Back» обработана и возвращает пользователя к списку активных курсов.



Рисунок 5.9 – Страница курса для пользователя «Преподаватель»



Рисунок 5.10 – Страница курса для пользователя «Студент»

Нажатие на кнопку «Messenger» открывает пользователю общий чат выбранного курса. На странице чата пользователю доступна вся история сообщений, а так же возможность отправить свое. Для каждого сообщения отображается время отправки и имя отправителя, сообщения отсортированы от самого раннего к самому последнему, таким образом нижнее сообщение будет самым свежим. Для удобства, сообщения, отправленные пользователям, акцентированы более ярким цветом. Сообщение вводится в текстовое поле «Input-message» и отправляется по нажатию клавиши «Send». Cтраница общего чата представлена на рисунке 5.11. Кнопка «Back» возвращает пользователя на страницу курса

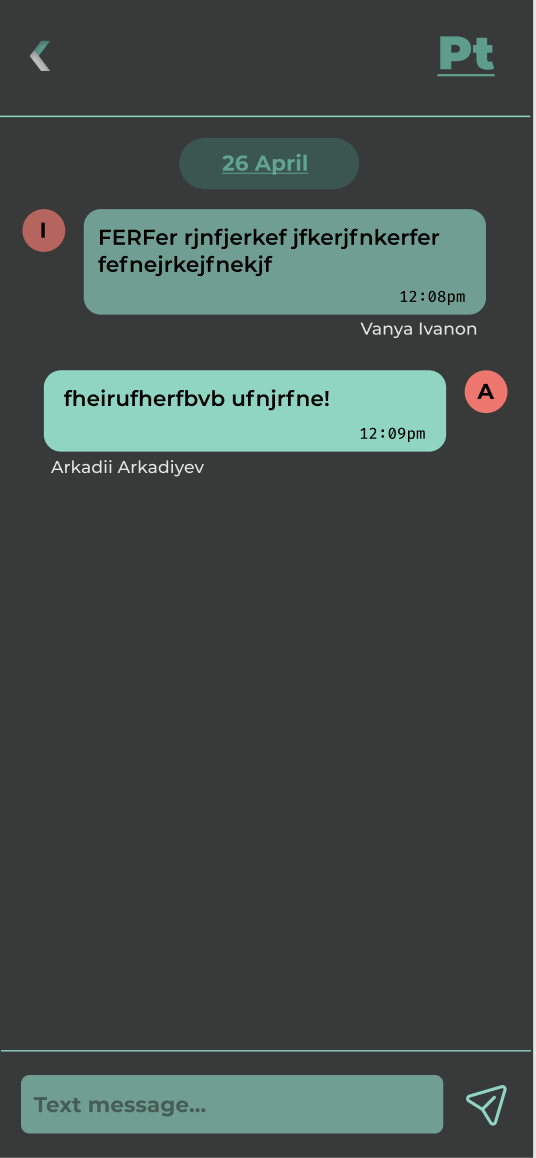


Рисунок 5.11 – Страница чата выбранного курса

1. ТЕХНИКО-ЭКОНОМИЧЕСКОЕ ОБОСНОВАНИЕ РАЗРАБОТКИ И РЕАЛИЗАЦИИ НА РЫНКЕ ИГРОВОГО ВЕБ-ПРИЛОЖЕНИЯ ДЛЯ МОДЕЛИРОВАНИЯ СТАРТАПОВ

## Характеристика игрового веб-приложения для стартапов

Игровое веб-приложение представляет собой интерактивную веб-платформу с элементами геймификации, где пользователи могут создавать виртуальные стартапы, проектировать бизнес-процессы и тестировать их успешность в условиях, приближенных к реальному рынку. Продукт ориентирован на жителей Республики Беларусь и направлен на обучение основам предпринимательства через игровые механики [9].

Основными функциями разработанного продукта являются:

1 Конструктор стартапов: создание бизнес-моделей, с использованием шаблонов, и инструментов визуализации (блок-схемы, диаграммы). Настройка параметров: целевая аудитория, бюджет, маркетинговые риски, риски.

2 **Симуляция развития бизнеса**: динамическая оценка успешности стартапа на основе введенных данных (алгоритмы анализа рентабельности, конкуренции, рыночного спроса). Генерация сценариев (кризисы, инвестиции, изменения законодательства) для проверки устойчивости модели.

3 Геймификация: системы достижений, баллов и уровней за выполнение задач (найм виртуальной команды, выход на безубыточность). Рейтинги пользователей и возможность соревнований в режиме «стартап-баттлов».

4 Образовательный модуль: база знаний с кейсами успешных белорусских стартапов (например, MSQRD, PandaDoc). Интерактивные уроки по основам бизнес-планирования и правовым аспектам РБ.

5 Социальное взаимодействие: возможность делиться проектами, получать обратную связь от сообщества, объединяться в виртуальные команды.

Основная цель приложения — помочь начинающим предпринимателям, студентам и энтузиастам развивать навыки стратегического планирования и анализа, минимизируя риски реальных финансовых потерь. Любой зарегистрированный пользователь может создать виртуальный стартап, выбрать отрасль (например, IT, розничная торговля, услуги), определить целевую аудиторию и выделить бюджет [10]. Система генерирует сценарии развития стартапа на основе введенных данных: рыночный спрос, конкуренция, внешние факторы (например, изменения законодательства РБ, экономические кризисы). Пользователи получают обратную связь в виде аналитических отчетов: прогноз прибыли/убытков, оценка рисков, рекомендации по оптимизации процессов. Встроенная геймификация (баллы, уровни, рейтинги) мотивирует участников совершенствовать свои проекты и соревноваться с другими пользователями.

Данное приложение предназначено для владельцев ПК, ноутбуков, а также телефонных устройств, которые имеют операционную систему Android или iOS с доступом к сети Интернет.

Основной аудиторией являются люди, заинтересованные в предпринимательской деятельности, студенты экономических, управленческих и IT-специальностей (например, БГУ, БГУИР), школьники 10–11 классов (в рамках профориентационных программ); бизнес-тренеры и консультанты (как инструмент для обучения). Наша платформа ориентируется на жителей республики Беларусь.

Реализация платформы на рынке планируется при помощи размещения её на облачном хостинге с доменом в зоне «.by». Реализованная платформа будет бесплатной и монетизироваться за счет встроенной рекламы.

* 1. Расчет инвестиций в разработку программного средства для   
     его реализации на рынке

Инвестициями являются затраты на разработку веб-платформы. Затраты на заработную плату команды, которая работает над проектом, исходят из состава команды и ее численности, размеров месячной заработной платы каждого участника команды и трудоемкости самой разработки.

Чтобы реализовать проект «Программное средство для моделирования бизнес-процессов в веб-среде» потребуется месяц работы команды разработки, состоящей из программиста, тестировщика, UI/UX-дизайнера и маркетолога.

Расчет основной заработной платы участников команды осуществляется по формуле:

где – это коэффициент премий и иных стимулирующих выплат; – это категории исполнителей, занятых разработкой программного средства; – часовой оклад исполнителя i-й категории, р; – трудоемкость работ, выполняемых исполнителем i-й категории, ч.

Таблица 1 – Расчет затрат на основную заработную плату команды разработчиков

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Категория исполнителя | Месячный оклад, р. | Часовой оклад, р. | Трудоемкость работ, ч. | Итого, р. |
| Программист | 5000,00 | 29,76 | 168 | 5000,00 |
| Тестировщик | 2500,00 | 14,88 | 168 | 2500,00 |
| UI/UX-дизайнер | 2000,00 | 11,90 | 72 | 857,00 |
| Маркетолог | 2000,00 | 11,90 | 72 | 857,00 |
| *Итого* | | | | 9214,00 |
| Премия и иные стимулирующие выплаты (по данным предприятия или 50-100%) | | | | 4607,00 |
| *Всего* затрат на основную заработную плату разработчиков | | | | 13821,00 |

Затраты на дополнительную заработную плату команды разработчиков определяется по формуле:

где – это норматив дополнительной заработной платы.

Норматив дополнительной заработной платы составляет 15%, подставляем полученные значения в формулу 7.2:

Отчисления на социальные нужды рассчитываются по формуле:

где – это норматив отчислений в ФСЗН и Белгострах.

В соответствии с действующим законодательством по состоянию на январь 2025 г. норматив отчислений в ФСЗН и Белгострах составляет 35%. Подставим полученные значения в формулу 7.3:

Затраты на прочие расходы рассчитываются по следующей формуле:

где – это норматив прочих расходов.

Примем показатель за 35%. Подставим полученные значения в формулу 7.4:

Общая сумма инвестиций на разработку находится по следующей формуле:

Подставим в формулу 7.5 полученные значения:

Таким образом получили общую сумму инвестиций на разработку.

* 1. Расчет экономического эффекта от реализации программного   
     средства на рынке

Целевой аудиторией являются начинающие предприниматели и стартаперы РБ; студенты университетов; школьники 10-11 классов; бизнес-тренеры и консультанты. На данный момент в Республике Беларусь данным Министерства экономики зарегистрировано **~45 тыс. субъектов малого и среднего бизнеса**, из которых около **3–5 тыс.** можно отнести к стартапам (IT, услуги, инновации). Программы поддержки («Стартап-страна», ПВТ) ежегодно охватывают **~1,5 тыс. проектов**. Активные хакатоны и конкурсы (например, Space Hackathon, Hi-Tech Park) привлекают **~2 тыс. участников в год**. При использовании платформы пользователям будет показываться нативная реклама. Цена одного показа рекламы составляет 0,003 белорусских рубля. Предполагаемая аудитория составляет 50000 пользователей. Предполагаемый доход от показа рекламы составляет 55000,00 рублей в год [11].

Прирост чистой прибыли, полученную командой разработки от реализации программного средства на рынке, можно рассчитать по формуле:

где – предполагаемый доход от рекламы за год, р.; НДС – сумма налога на добавленную стоимость, р.; – ставка налога на прибыль согласно действующему законодательству, % (по состоянию на февраль 2025 г. – 25%)

Налог на добавленную стоимость определяется по формуле

где – ставка налога на добавленную стоимость в соответствии с действующим законодательством, % (по состоянию на февраль 2025 г. – 20%).

Используя имеющиеся значения, проведем расчет НДС (расчёт налога на прибыль):

Теперь, зная налог на добавленную стоимость, подсчитаем прирост чистой прибыли, которую может получить разработчик от продажи программного продукта. Сделать это можно с помощью следующей формулы:

где – количество копий (лицензий) программного средства, реализуемого за год, шт.

– отпускная цена копии (лицензии) программного средства, р.

– сумма налога на добавленную стоимость.

– ставка налога на прибыль, %.

– рентабельность продаж копий (лицензий)

Зная размер налога на добавленную стоимость, можно провести расчет прироста численной прибыли, которую получит команда разработки от реализации программного продукта. Подсчитано с учётом того под заработок от рекламы, а не от продажи лицензий. Соответственно уменьшаемое () р. [заработок от рекламы]. Ставка налога на прибыль по состоянию на 03.03.25 составляет 20 %. Используя имеющиеся данные, необходимо посчитать прирост чистой прибыли команды разработки:

Таким образом прирост чистой прибыли от реализации веб-платформы составит рублей в год.

* 1. Расчет показателей экономической эффективности разработки и   
     реализации программного средства на рынке

Оценка экономической эффективности разработки и реализации программного средства на рынке зависит от результатов сравнения инвестиций в его разработку и полученного годового прироста чистой прибыли.  
Разработка и внедрение программного обеспечения экономически эффективны, если годовой прирост чистой прибыли превышает инвестиции в разработку, оценка экономической эффективности инвестиций в разработку программного средства осуществляется с помощью расчета рентабельности инвестиций по формуле

где – прирост чистой прибыли, полученной от реализации программного средства на рынке, р.; – затраты на разработку и реализацию программного средства, р.

Необходимо рассчитать оценку экономической эффективности инвестиций используя уже имеющиеся данные. Проведем расчет оценки эффективности используя формулу (7.7)

Разработка веб-платформы экономически эффективна, показав рентабельность в 39,45%, что значительно выше ставки рефинансирования (11,5%). Положительная оценка эффективности подтверждает окупаемость инвестиций. Однако, достижение этих результатов зависит от критического сценария: ежедневного посещения платформы и просмотра рекламы всеми 50 000 потенциальных пользователей. Всегда существует риск недостижения целевой аудитории или низкой популярности на рынке [12].

Успешное развитие программного средства для моделирования бизнес-процессов в веб-среде может привести к значительному расширению клиентской базы и увеличению рыночной доли. Дальнейшее совершенствование продукта, например, путем добавления новых функций, интеграции с другими платформами или улучшения пользовательского интерфейса, может привлечь больше пользователей и повысить его конкурентоспособность. Однако, любые изменения и улучшения потребуют дополнительных инвестиций, которые могут выражаться в увеличении времени разработки, привлечении дополнительных специалистов, или приобретении новых технологий. Внедрение новых функций, таких как расширенная аналитика, автоматизация процессов или интеграция с системами управления предприятием (ERP), может существенно повысить ценность продукта для пользователей, но одновременно потребует перерасчета всех экономических показателей проекта, включая рентабельность и срок окупаемости. Поэтому, перед принятием решения о внедрении новых функций или улучшений, необходимо провести детальный анализ затрат и потенциальной прибыли, чтобы убедиться в экономической целесообразности таких изменений и их соответствии общей стратегии развития программного средства. Только после тщательного анализа и планирования можно гарантировать, что инвестиции в развитие продукта будут эффективными и приведут к желаемому росту прибыли.

ЗАКЛЮЧЕНИЕ

Исследование ситуации в области веб-разработки подтвердило, что веб-приложения продолжают занимать лидирующие. Динамика развития веб-технологий, внедрение новых интерфейсов для безопасного взаимодействия с сервисами, а также рост мощностей серверных и клиентских технологий дают основания ожидать продолжение этого тренда в будущем. Поддержка таких крупных игроков, как Google, позволяет уверенно работать в области веб-разработки, ориентируясь на стабильное будущее и востребованность специалистов в этой сфере.

В рамках работы было разработано веб-приложение для моделирования и управления бизнес-процессами, ориентированное на стартапы. Система автоматизирует процесс планирования, управления и развития бизнеса, с фокусом на два типа: физический бизнес (например, заводы и фабрики) и инвестиционный (например, криптовалюты, инвестиции, ставки). Платформа предоставляет пользователю гибкость в настройке процессов, отслеживании изменений и оптимизации бизнес-стратегий.

Система позволяет вводить данные о текущем состоянии бизнеса, выбирать стратегии развития, нанимать сотрудников и моделировать рост бизнеса в реальном времени. Для физического бизнеса пользователи смогут наблюдать, как их предприятие растет, начиная с малого и увеличиваясь в размерах, создавая новые отделы и расширяя инфраструктуру. В случае инвестиционного бизнеса, пользователи будут управлять виртуальными средствами, что позволит им без риска для реальных денег тестировать различные стратегии на рынке криптовалют, акций или ставок.

Проект ориентирован на удобство и гибкость, предлагая удобный веб-интерфейс для пользователей. Все данные о бизнес-процессах и бизнес-стратегиях будут визуализироваться в виде графиков и диаграмм, которые при этом не являются статичными, а динамично изменяются на основе введенных данных и принятных решений. Важной особенностью является то, что эти визуальные эффекты и графики скрыты под капотом, а для пользователя доступна лишь удобная и понятная информация о развитии его бизнеса.

В процессе разработки были использованы современные веб-технологии, включая Python с FastAPI для серверной части, React для клиентской части, PostgreSQL для базы данных и WebSockets для обмена данными в реальном времени. Также проект включает интеграцию инструментов тестирования и контроля качества кода, таких как Pytest, Jest, Flake8, Black, ESLint и Prettier.

Проект ориентирован исключительно на веб-платформу, что позволяет сосредоточиться на создании стабильной и высококачественной веб-системы. В перспективе проект может быть расширен и адаптирован под другие бизнес-области, а также использован в образовательных целях для демонстрации и моделирования различных сценариев бизнес-решений.

СПИСОК ИСПОЛЬЗОВАННОЙ ЛИТЕРАТУРЫ

1. Макконнелл, С. Совершенный код. Мастер-класс / Пер. с англ. / С. Макконнелл. – М. : Издательско-торговый дом «Русская редакция», 2010. – 896 с.
2. Garlan, David. An Introduction to Software Architecture / David Garlan, Mary Shaw / Ed. by V Ambriola, G Tortora. – New Jersey : World Scientific Publishing Company, 1994. – Vol. I. – Режим доступа: https://www.cs.cmu.edu/afs/cs/project/vit/ftp/pdf/intro\_softarch.pdf. – Дата доступа: 01.04.24.
3. Getting Started with Progressive Web Apps [Электронный ресурс]. – Электронные данные. – Режим доступа: <https://developers.google.com/web/updates/2015/12/getting-started-pwa/> – Дата доступа: 23.03.25.
4. Куликов, С. С. Тестирование программного обеспечения. Базовый курс / С. С. Куликов. – EPAM Systems, RD Dep., 2017. – http://svyatoslav.biz/software\_testing\_book/.
5. React [Электронный ресурс]. – Электронные данные. – Режим доступа: https://react.org/. – Дата доступа: 18.03.25.
6. What is an interstitial ad? [Электронный ресурс]. – Режим доступа: https://www.adjust.com/glossary/interstitial-ad/. – Дата доступа: 27.02.2024.
7. Стартап-экосистема Беларуси [Электронный ресурс]. – Режим доступа: https:// startupbelarus.by. – Дата доступа: 03.03.2024.
8. Национальный статистический комитет Республики Беларусь [Электронный ресурс]. – Режим доступа: https://www.belstat.gov.by. – Дата доступа: 14.03.2024.
9. Парк высоких технологий (ПВТ). Официальный сайт. [Электронный ресурс]. – Режим доступа: https://park.by – Дата доступа: 04.03.2024.
10. Rabota.by [Электронный ресурс]. – Режим доступа:  https://rabota.by/?hhtmFrom=vacancy\_search\_list. – Дата доступа: 22.02.2024.
11. Налог на добавленную стоимость [Электронный ресурс]. – Режим доступа: https://myfin.by/wiki/term/nalog-na-dobavlennuyu-stoimost. – Дата доступа: 04.03.2024.
12. Что такое ROI? [Электронный ресурс]. – Режим доступа: https://soldimarketing.ru/analitika/roi/. – Дата доступа: 15.03.2024.