МИНОБРНАУКИ РОССИИ
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ
ВЫСШЕГО ОБРАЗОВАНИЯ
«ВОРОНЕЖСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»
(ФГБОУ ВО «ВГУ»)

Факультет компьютерных наук

Кафедра информационных технологий управления

Курсовая работа

Веб-приложение для ведения домашнего бюджета

Направление 09.03.02 «Информационные системы и технологии»

Профиль «Информационные технологии управления»

Преподаватель _____ В.С. Тарасов, ст. преподаватель

Обучающийся _____А.А. Пустовалов, 3 курс, д/о

Обучающийся _____В.Г. Новиков, 3 курс, д/о

Обучающийся _____Е.О. Бордюжа, 3 курс, д/о

Воронеж 2023

Содержание

Введение

Ведение личного и совместного бюджета является важной повседневной задачей для многих людей, которые хотят контролировать свои расходы и вести учет своих доходов. В настоящее время существует множество приложений и сервисов для управления бюджетом, но часто они либо не удовлетворяют всем потребностям пользователей, либо не позволяют контролировать совместный бюджет нескольким пользователям.

В данной курсовой работе рассмотрен процесс разработки веб-приложения для мониторинга личного и совместного бюджета, которое будет условно бесплатным и удобным в использовании. Для реализации данного приложения будут использованы современные технологии веб-разработки, такие как React, Node.js, PostgreSQL.

Целью данной курсовой работы является практическое применение знаний и навыков веб-разработки для создания полноценного коммерческого приложения для ведения бюджета, которое будет полезно для широкого круга пользователей. Работа будет содержать детальное описание всех этапов процесса разработки, включая анализ требований, проектирование приложения, реализацию, тестирование.

1 Постановка задачи

Задачей данной курсовой работы является разработка веб-приложения для ведения домашнего бюджета. Онлайн-сервис позволит систематизировано вести учет расходов и доходов, отслеживать статистику по каждой категории трат.

Следует реализовать следующие пункты:

- Регистрация нового пользователя;
- Авторизация;
- Формирование и просмотр статистики по расходам;
- Возможность добавления пользователя для ведения совместного бюджета;
- Запись трат и входящих денежных средств по категориям.

2 Анализ предметной области

2.1 Глоссарий

- Авторизация – это процесс проверки прав пользователя на осуществление определенных действий на сайте;
- База данных – это упорядоченный набор структурированной информации или данных, которые хранятся в электронном виде в компьютерной системе;
- Бургер-меню – это иконка в пользовательском интерфейсе, представляющая собой три горизонтальные полоски, напоминающие бургер, которая при нажатии открывает боковое меню с навигационными или дополнительными опциями;
- Бюджет – это расходы и доходы конкретного человека или группы лиц;
- Валидация – это процесс проверки данных на соответствие заданным правилам или условиям для обеспечения их корректности и целостности;
- Веб-приложение, веб-сервис, интернет-сервис, онлайн-сервис, проект – это программное обеспечение, которое размещено на удаленном сервере и доступно через браузеры в интернете;
- Демо-режим – это режим работы приложения, в котором доступна ограниченная функциональность;
- Клиент – это объект, запрашивающий информацию по сети;
- Личный кабинет, профиль – это раздел сервиса, в котором пользователь может получить доступ к персональным данным;
- Мониторинг – это отслеживание расходов и доходов по категориям;
- Развертывание – это все действия, которые делают систему готовой к использованию;

- Регистрация – это способ сообщить сервису данные о себе и в обмен получить доступ к дополнительным ресурсам на сайте, которые недоступны гостям;
- Резиновая верстка – это подход к веб-разработке, в рамках которого создаются масштабируемые сайты, способные подстраиваться под разрешение текущего экрана;
- Сервер – это отдельный класс компьютерных устройств, предназначенных для обработки запросов от различных узлов сети;
- СУБД – это система управления базы данных;
- Фреймворк – это программная среда, облегчающая разработку и объединение разных компонентов большого программного проекта;
- Хеширование – это процесс преобразования входных данных в неповторимую строку фиксированной длины, которая служит в качестве уникального идентификатора для этих данных;
- Хостинг – это услуга, предоставляемая провайдером, которая позволяет размещать веб-сайты, приложения или другие ресурсы на удаленных серверах, обеспечивая доступ к ним через интернет;
- Frontend – это клиентская сторона пользовательского интерфейса к программно-аппаратной части сервиса;
- Backend – это программно-аппаратная часть сервиса;
- REST API – это стиль архитектуры программного обеспечения для построения распределенных масштабируемых веб-сервисов;
- React – это JavaScript библиотека для создания пользовательских интерфейсов;
- SPA – это одностраничное веб-приложение, которое работает на одной HTML-странице, обновляя данные на ней;
- GitHub – это крупнейший веб-сервис для хостинга IT-проектов и их совместной разработки;

– Header – это верхняя часть веб-страницы, которая содержит информацию о его содержании или навигационные элементы;
– HTTP – это протокол передачи данных, используемый для обмена информацией между клиентом и сервером в сети интернет;
– HTTPS – это защищенная версия протокола HTTP, которая обеспечивает шифрование данных для безопасной передачи информации между клиентом и сервером;
– HTML – это стандартизированный язык разметки документов для просмотра веб-страниц в браузере;
– SQL-запросы – это наборы команд для работы с реляционными (табличными) базами данных;
– SQL-инъекция – это один из способов взлома сайта.

2.2 Сценарии пользователей

1. Пользователь: Александр, 32 года.

   Описание: имеет постоянный доход, занимается бизнесом и желает отслеживать свои расходы.

   Пользовательская история: Александр хочет вести учет своих расходов на личные нужды и бизнес, чтобы определить, какие категории занимают большую часть его бюджета и где можно сэкономить.

2. Пользователь: Анастасия, 26 лет.

   Описание: студентка, живет в общежитии, получает стипендию, не имеет постоянного дохода.

   Пользовательская история: Анастасия хочет контролировать траты, чтобы не истощить свою стипендию слишком быстро и иметь возможность купить то, что ей действительно нужно.

3. Пользователь: Дмитрий, 40 лет.

   Описание: имеет постоянный доход, семейный человек.

Пользовательская история: Дмитрий и его жена хотят вести совместный бюджет, отслеживать свои расходы и доходы, чтобы узнать, сколько они тратят на различные категории и насколько могут сэкономить. Они хотят иметь возможность просмотреть свою статистику расходов за месяц и планировать свой бюджет на будущее.

2.3 Цели создания веб-приложения

Данное приложение создается для коммерческих целей Для этого пользователю необходимо предоставить функциональность, которая позволит отслеживать движение личных денежных средств, просматривать статистику по расходам и доходам, а также вести совместный бюджет.

Для достижения данной цели были выделены следующие задачи:

- Проектирование и развертывание базы данных;
- Разработка frontend части веб-приложения;
- Реализация бизнес-логики приложения на сервере;
- Реализация связи между клиентом и сервером с применением подхода REST API;
- Развертывание приложения на серверной части.

2.4 Сфера применения

Приложения для ведения бюджета имеют большое значение для многих людей, поскольку позволяют им контролировать свои расходы и доходы, следить за своим финансовым положением и планировать свои расходы. Это может быть особенно полезно для тех, кто имеет ограниченный бюджет или хочет экономить на своих расходах.

Использование приложений для учета бюджета может также помочь людям научиться лучше управлять своими деньгами и развивать финансовую грамотность. Это может включать в себя понимание, как управлять своими расходами, как правильно планировать свои финансы и как принимать обоснованные финансовые решения.

Некоторые из наиболее распространенных сфер применения включают в себя:

- Личные финансы: приложения для учета бюджета могут помочь людям контролировать свои личные финансы, планировать свои траты, отслеживать свои доходы и расходы, а также устанавливать финансовые цели и следить за их выполнением;
- Семейный бюджет: приложения для учета бюджета могут помочь семьям планировать свой бюджет, учитывая расходы на питание, жилье, транспорт и другие категории трат;
- Финансовое планирование: приложения для учета бюджета могут помочь людям планировать свои финансовые цели и следить за их выполнением, а также помочь им принимать взвешенные финансовые решения;
- Обучение финансовой грамотности: приложения для учета бюджета могут быть полезны для обучения финансовой грамотности, а также помочь людям научиться планировать свои финансы и контролировать расходы.

2.5 Обзор аналогов

2.5.1 Мобильное приложение Monefy

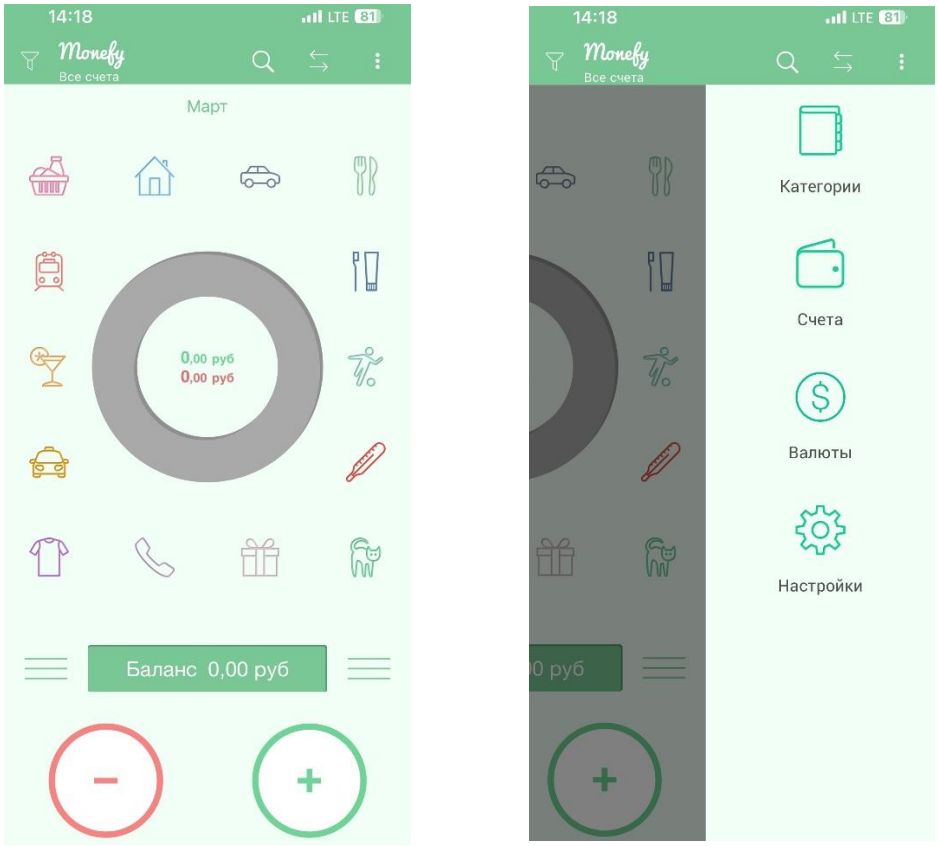


Рисунок 1 - Приложение Monefy

Сильные стороны приложения:

- Есть функция «совместного доступа», благодаря которой можно вести общий бюджет с разных устройств;
- Есть авторизация с паролем;
- Есть встроенный калькулятор для автоматического подсчета бюджета;
- Гибкие настройки: различные категории доходов и расходов, поддержка различных валют;
- Реализован выбор периода для анализа от дня до года;
- Есть возможность пользоваться приложением без авторизации.

Слабые стороны приложения:

- Приложение работает только на мобильных устройствах, нет веб-версии;

– При отсутствии подписки: реклама, невозможность создавать новые пользовательские категории трат;
– Нет возможности попробовать приложение в демо-режиме;
– Нет системы уведомлений.

2.5.2 Приложение Toshl finance

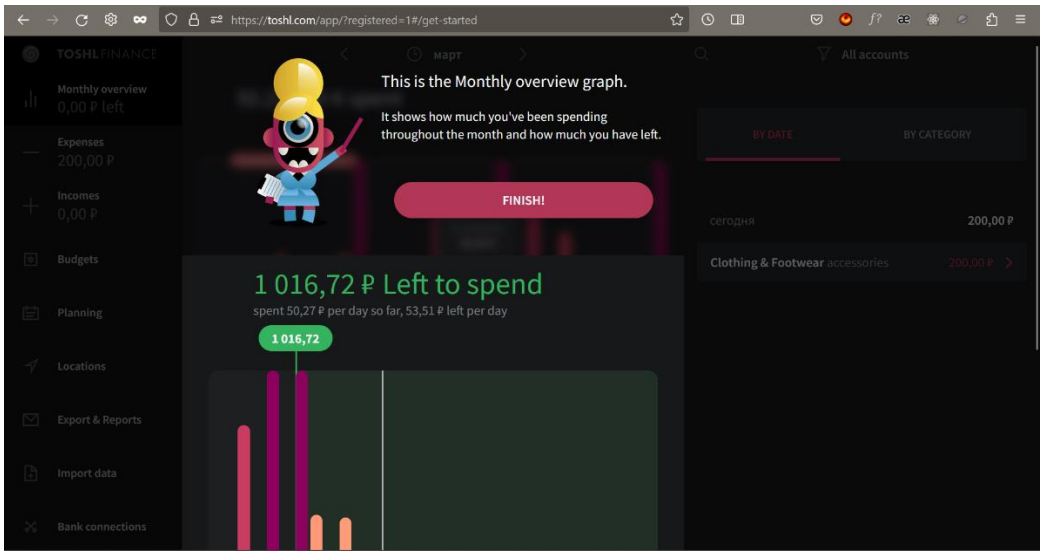


Рисунок 2 - Приложение Toshl finance

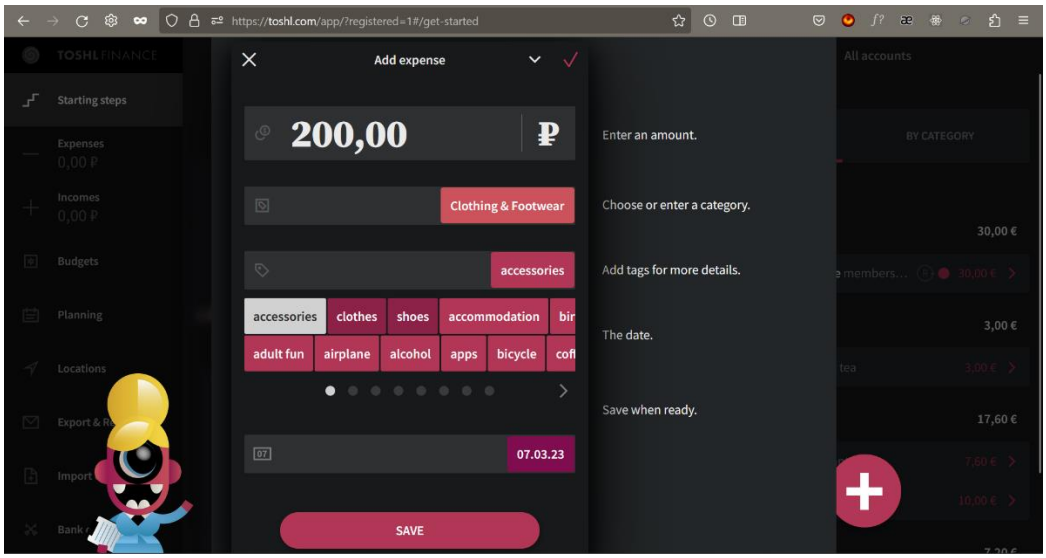


Рисунок 3 - Приложение Toshl finance

Сильные стороны приложения:

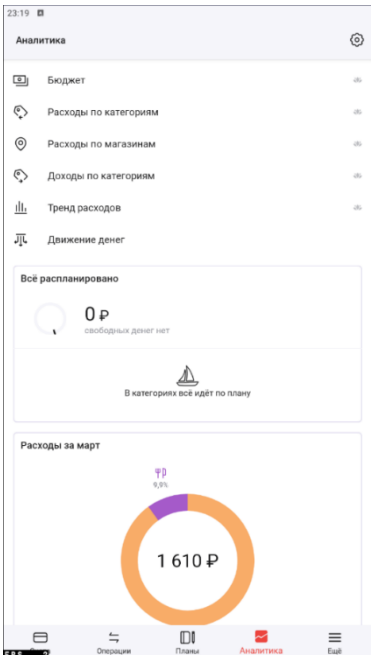
– Есть мобильная и веб-версия приложения;

– Поддержка множества видов валют;
– Реализованы информативные графики с историями доходов и расходов;
– Существует клиентская служба поддержи, а также есть возможность связаться с разработчиками и сообщить о возникших ошибках в работе приложения.

Слабые стороны приложения:

– Постоянно меняются языки с русского на английский и обратно в категориях и пометках, причем как в приложении, так и в веб версии;
– Приложение выдает сбои в обновлениях состояния карт и данных о тратах;
– При обновлениях приложения сбрасываются все настройки пользователя;
– Регулярная реклама о подписке;
– Нет общего бюджета;
– Нет возможности попробовать приложение в демо-режиме;
– Нет возможности пользоваться приложением без авторизации.

2.5.3 Приложение Дзен-мани



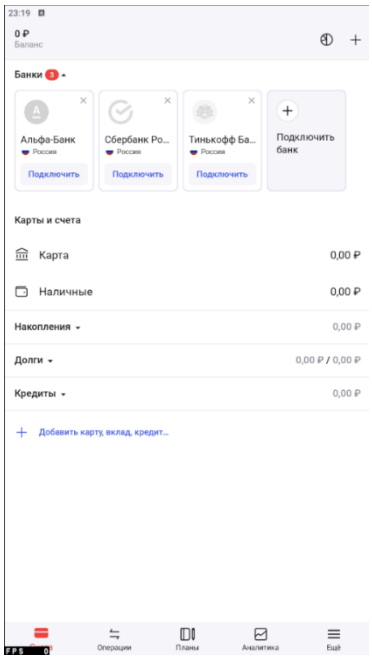


Рисунок 4 - Приложение Дзен-мани

Сильные стороны приложения:

– Реализована система уведомлений;
– Есть общий бюджет;
– Есть авторизация с паролем.

Слабые стороны приложения:

– Приложение работает только на мобильных устройствах, нет веб-версии;
– Нет возможности пользоваться приложением без авторизации;
– Недоступна функция аналитики за прошлый месяц;
– Нет возможности попробовать приложение в демо-режиме;
– Нет возможности сгруппировать покупки по категориям.

2.5.4 Приложение Coinkeeper 3

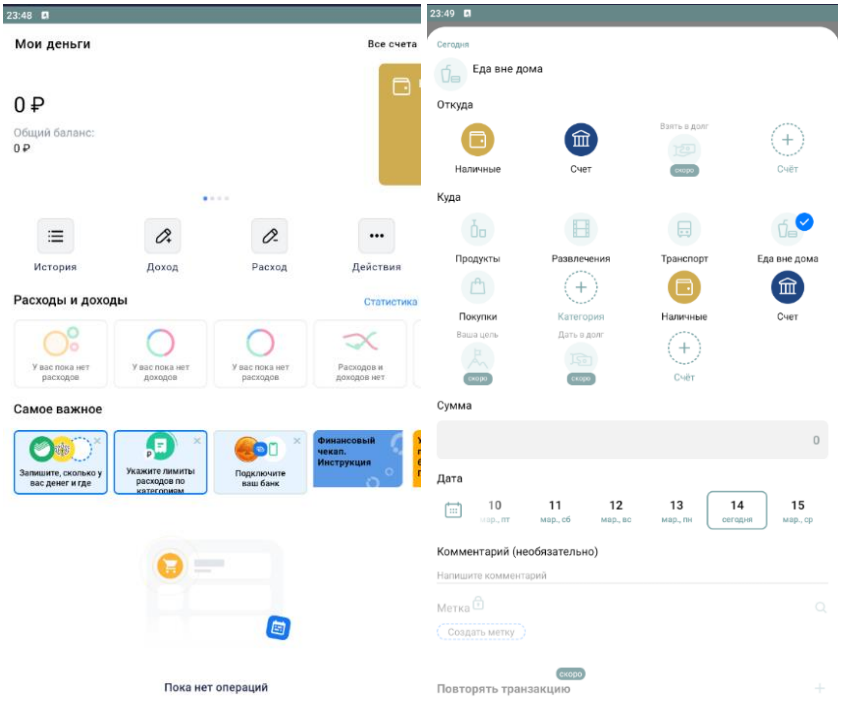


Рисунок 5 - Приложение Coinkeeper 3

Сильные стороны приложения:

– Есть мобильная версия приложения;

– Есть возможность пользоваться приложением без авторизации;
– Присутствуют курсы и инструменты финансовой грамотности.

Слабые стороны приложения:

– Нет веб-версии приложения;
– Ограниченная функциональность без подписки (нельзя добавить новые категории трат);
– Нет общего бюджета;
– Нет возможности попробовать приложение в демо-режиме;
– Плохая техподдержка (многие пользователи жалуются, что происходит полное игнорирование со стороны разработчиков).

2.5.5 Выводы из обзора аналогичных решений

Сравнение аналогов показало следующий результат, приведенный в таблице 1:

Таблица 1 - Сравнение аналогов

Критерии/Приложения	Monefy	Toshl finance	Дзен-мани	Coinkeeper 3
Мобильная версия	+	+	+	+
Веб-версия	-	+	-	-
Авторизация	+	+	+	+
Общий бюджет	-	-	+	-
Уведомления	-	-	+	-
Статистика	+	+	-	-
Демо-режим	-	-	-	-
Подписка	-	+	-	+
Инструменты финансовой грамотности	-	-	-	+
Техническая поддержка	-	+	-	-

2.6 Требования к веб-приложению в целом

Приложение должно удовлетворять следующим требованиям:

- Обеспечить работу в Google Chrome, Firefox и Microsoft Edge;
- Предоставить возможность регистрации и авторизации;
- Предоставить возможность уже авторизованному пользователю пригласить друга для ведения совместного бюджета;
- Предоставить статистику для анализа расходов и доходов по категориям. Авторизованный пользователь может смотреть, сколько денег он тратит и зарабатывает в каждой категории, например, в категории «Продукты», «Здоровье» или «Развлечения»;
- На данном этапе разработки не будет интеграции с платежными сервисами, поэтому все финансовые операции, к примеру оформление премиум подписки, будут реализованы в виде заглушек.

2.6.1 Требования к функциональности

Разрабатываемое веб-приложение должно обладать следующей функциональностью:

- Возможность регистрации и авторизации для пользователей;
- Добавление записей расходов и доходов по категориям;
- Просмотр статистики расходов и доходов по каждой категории за определенный период;
- Реализация возможности приглашения друга для ведения совместного бюджета;
- Предоставление пользователю информации о финансовой грамотности.

2.7 Продуктовые воронки

Рассмотрим количество шагов, которое необходимо пройти авторизованному пользователю для внесения данных и получения статистики доходов или расходов. При открытии приложения авторизованному пользователю будет показан главный экран, где пользователь может внести

данные о тратах или поступлениях, нажав на «плюс» в нижнем правом углу. Для получения статистики пользователь должен выбрать в боковом меню раздел «Статистика» или на главном экране нажать на кнопку «Статистика».

Таким образом, необходимо всего 3 шага для пользования основными функциями приложения.

2.8 Пользователи системы

В системе существуют такие группы пользователей как: неавторизованный, авторизованный, премиум-пользователь и участник группы.

Разрабатываемое приложение предоставляет возможность вести совместный бюджет группе пользователей. Вместимость группы по умолчанию составляет два человека, но если хотя бы один участник группы является премиум-пользователем, то вместимость группы увеличивается до пяти человек, что является максимальной вместимостью.

Неавторизованный пользователь – посетитель веб-сайта, узнавший о сервисе из поисковой выдачи или любым другим способом. Для неавторизованного пользователя должна быть реализована следующая функциональность:

- Регистрация в веб-приложении;
- Вход в демо-режим, в котором доступна ограниченная функциональность веб-приложения, а именно: просмотр трат, внесенных заранее (данные траты являются предзаписанными неизменяемыми данными), возможность фильтрации таких трат.

Авторизованный пользователь – пользователь, прошедший процесс авторизации. Для авторизованного пользователя должна быть реализована следующая функциональность:

- Приглашение одного друга в группу для ведения совместного бюджета;

– Добавление доходов и расходов. При внесении трат и поступлений можно выбрать одну из предложенных категорий. Без премиум-подписки пользователь ограничен количеством таких категорий;
– Оформление премиум подписки;
– Просмотр статистики трат;
– Фильтрация трат по категориям;
– Возможность выхода из аккаунта.

Премиум-пользователь – авторизированный пользователь, который приобрел подписку на сервис, открывающую дополнительные функциональные возможности. Для премиум-пользователя должна быть реализована та же функциональность, что и для авторизованного, а также:

– Увеличение количества возможных участников в группе, в которой состоит премиум-пользователь, до пяти человек;
– Неограниченное количество используемых категорий.

Участник группы – авторизованный пользователь, ведущий совместный бюджет. Разрешается состоять не более чем в одной группе. Данному пользователю доступна следующая функциональность:

– Добавление трат в общий бюджет;
– Добавление трат в личный бюджет. Траты в личном бюджете являются приватными, то есть они не видны другим пользователям группы;
– Просмотр статистики трат;
– Фильтрация трат по категориям;
– Возможность выхода из группы.

3 Анализ задачи

3.1 Диаграмма прецедентов

Диаграмма прецедентов помогает описать, как пользователи будут взаимодействовать с системой. На рисунке 6 представлена диаграмма прецедентов для авторизованного пользователя, а на рисунке 7 – для неавторизованного пользователя.

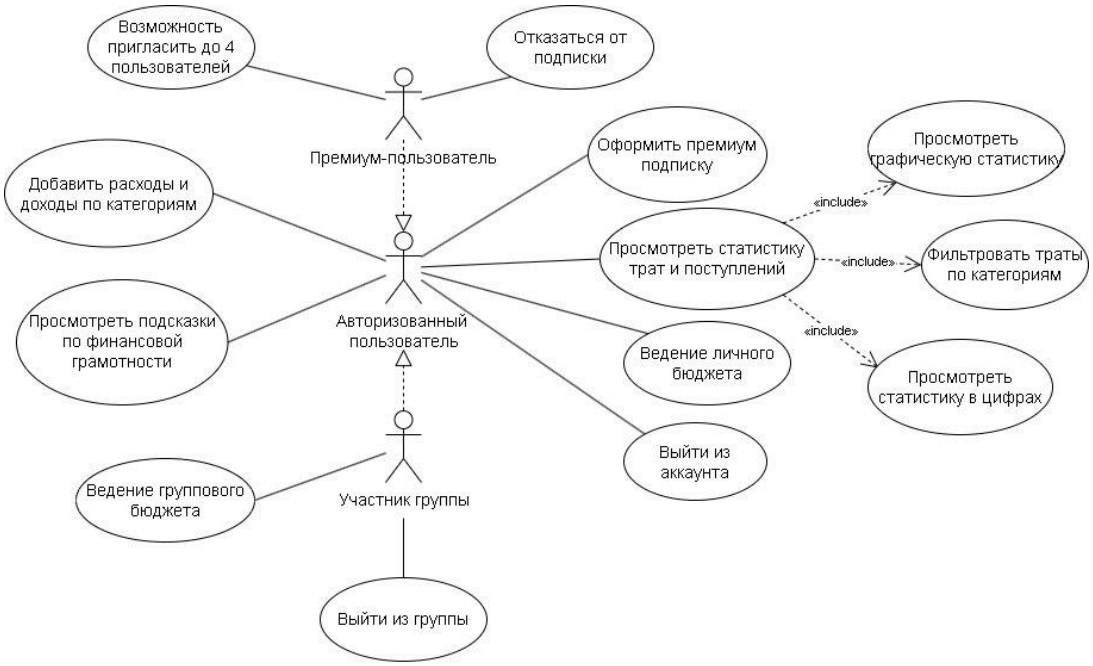


Рисунок 6 - Диаграмма прецедентов

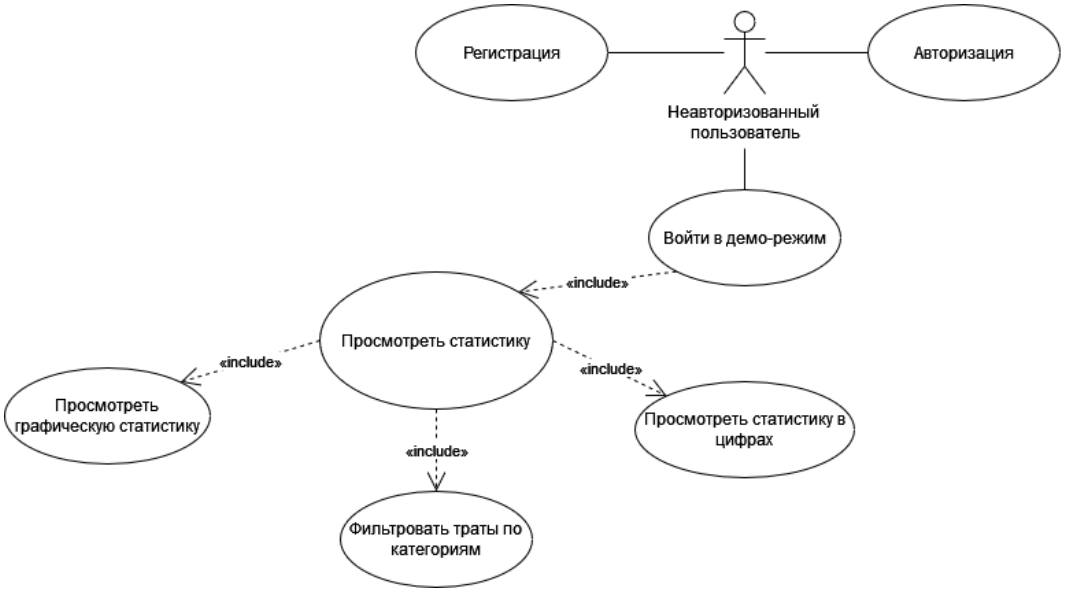


Рисунок 7 - Диаграмма прецедентов

3.2 Диаграмма последовательностей

Диаграмма последовательностей используется для уточнения диаграмм прецедентов, более детального описания логики сценариев использования. Диаграмма последовательности для неавторизованного пользователя представлена на рисунке 8, а для авторизованного – на рисунке 9.

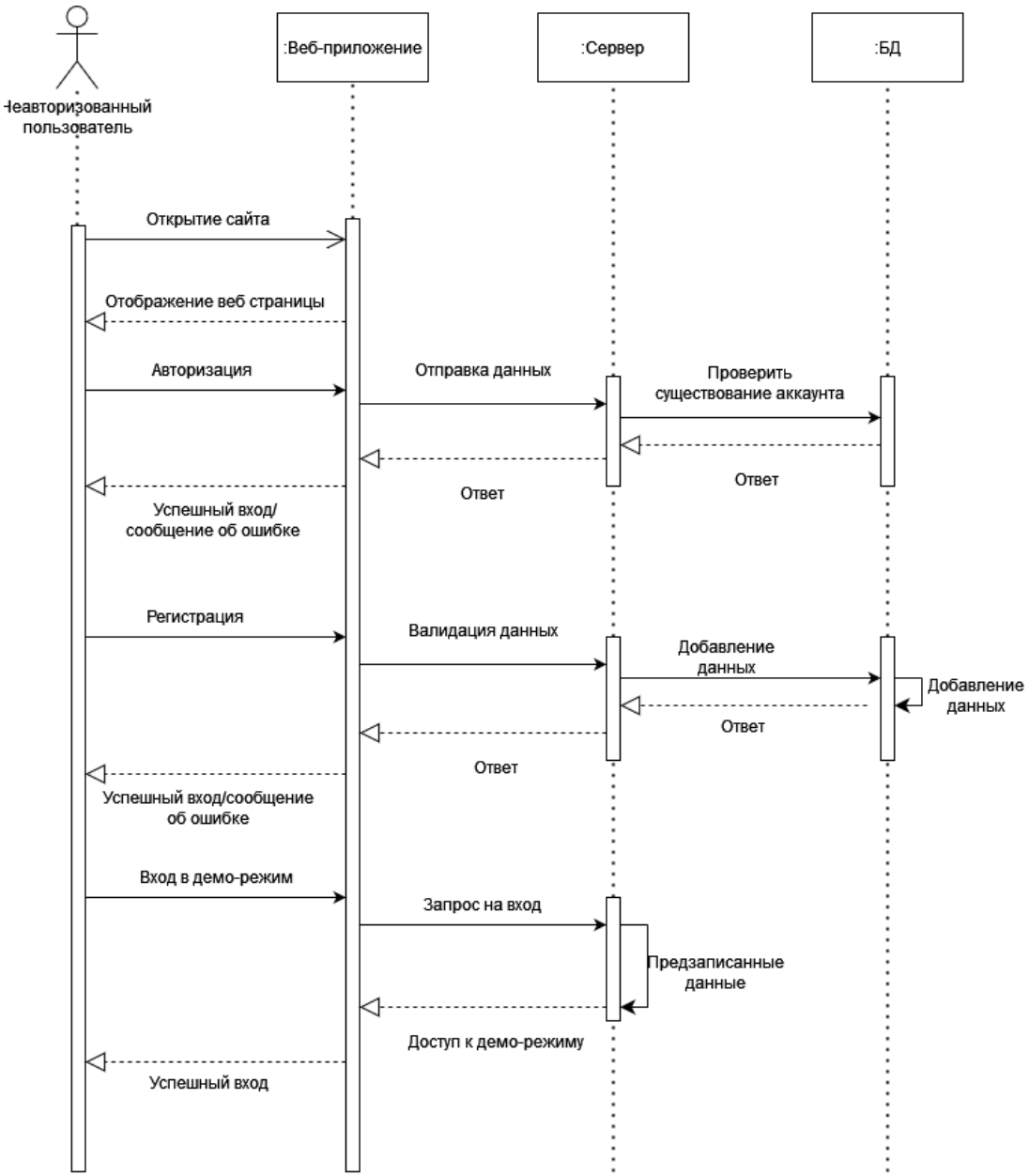


Рисунок 8 - Диаграмма последовательности

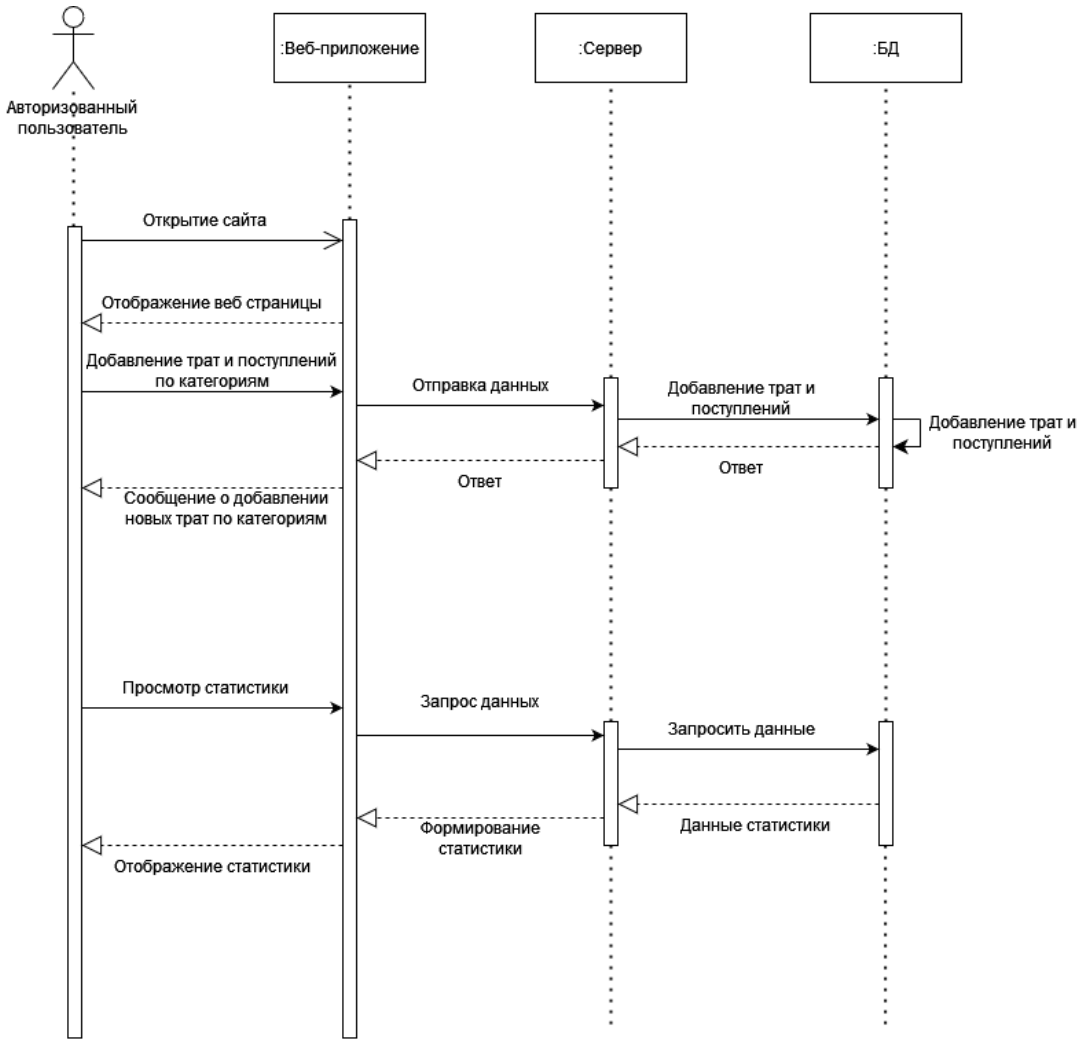


Рисунок 9 - Диаграмма последовательности

3.3 Диаграмма состояний

Диаграмма состояний показывает все возможные изменения в состоянии конкретного объекта на протяжении всего его жизненного цикла (рисунок 10-11).

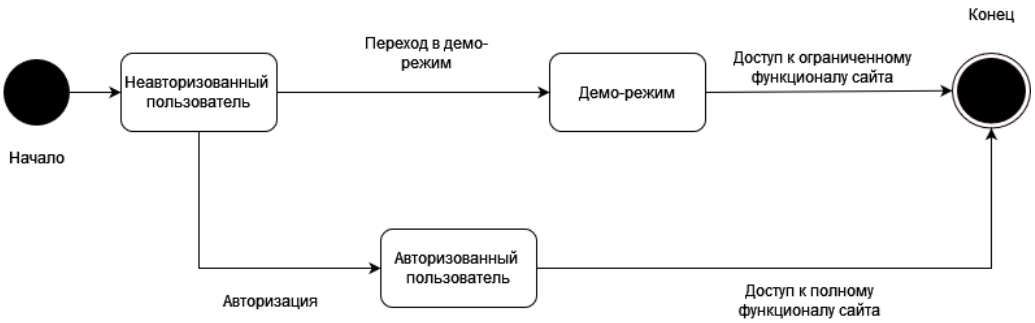


Рисунок 10 - Диаграмма состояния пользователя

Рисунок 11 - Диаграмма состояния группы

3.4 Контекстная диаграмма (IDEF0)

Диаграммы по методологии IDEF0 (рисунок 12-13). Эта методология функционального моделирования и графическая нотация, которая предоставляет возможность описать бизнес-процессы и определить их структуру, что позволяет более детально показать функциональность проекта и разработать наиболее оптимальный план действий при его реализации.

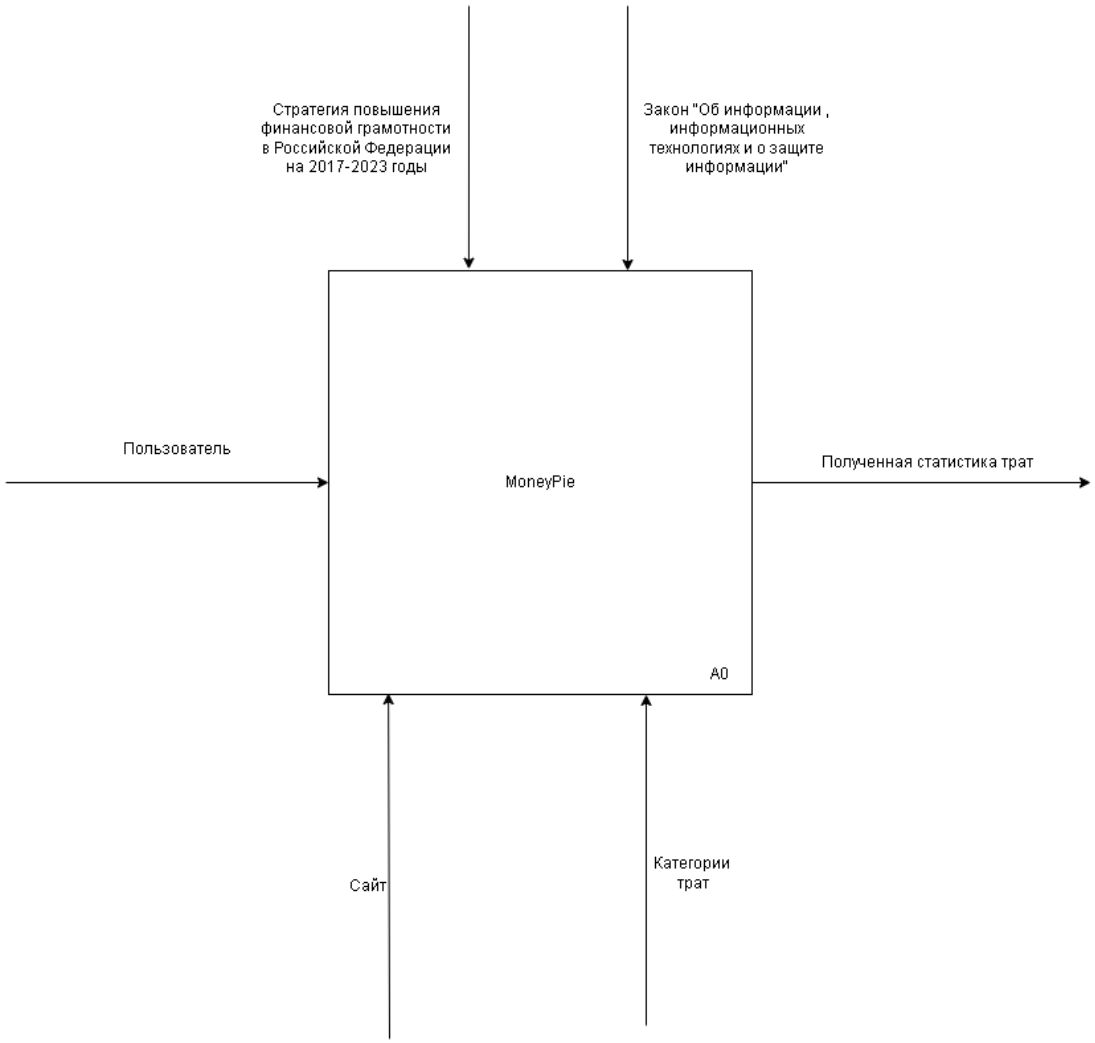


Рисунок 12 - IDEF0 уровень 0

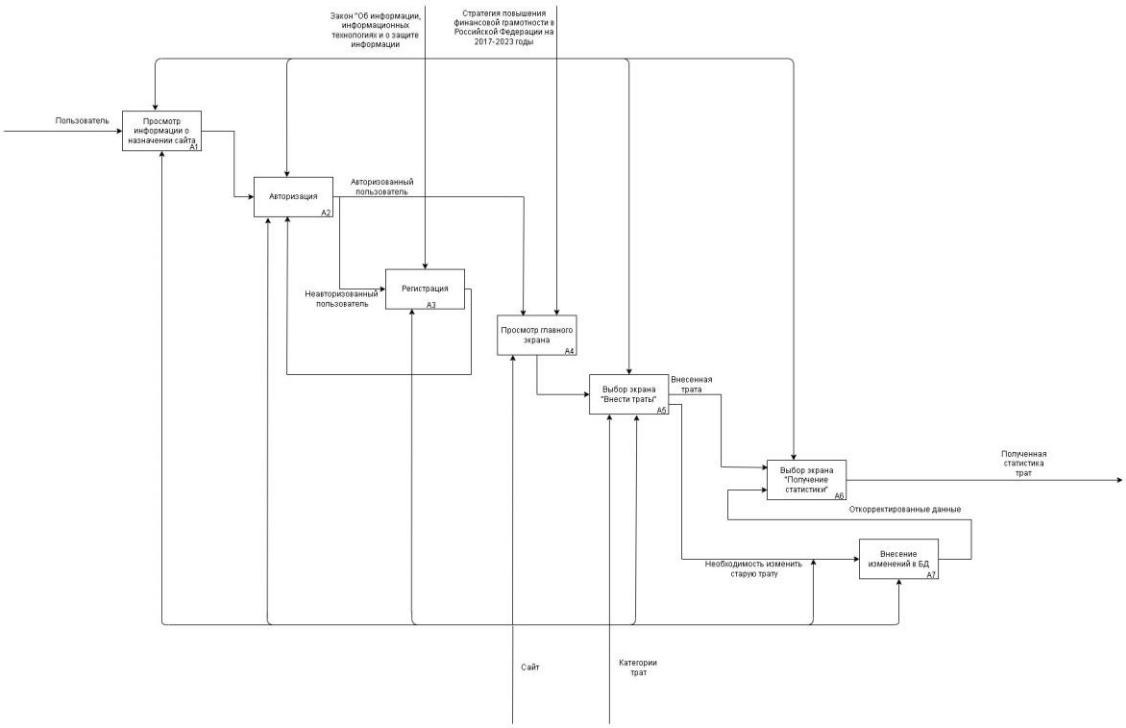


Рисунок 13 - IDEF0 уровень 1

3.5 Диаграмма сотрудничества

Диаграмма сотрудничества определяет набор взаимодействующих ролей, используемых вместе, чтобы показать функциональность приложения (рисунок 14-16).

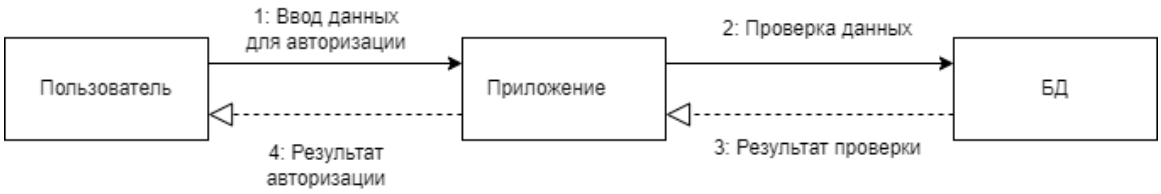


Рисунок 14 - Диаграмма сотрудничества

Рисунок 15 - Диаграмма сотрудничества

Рисунок 16 - Диаграмма сотрудничества

3.6 Диаграмма активностей

Диаграмма активностей отражает динамические аспекты поведения системы и наглядно показывает, как поток управления переходит от одной деятельности к другой (рисунок 17).

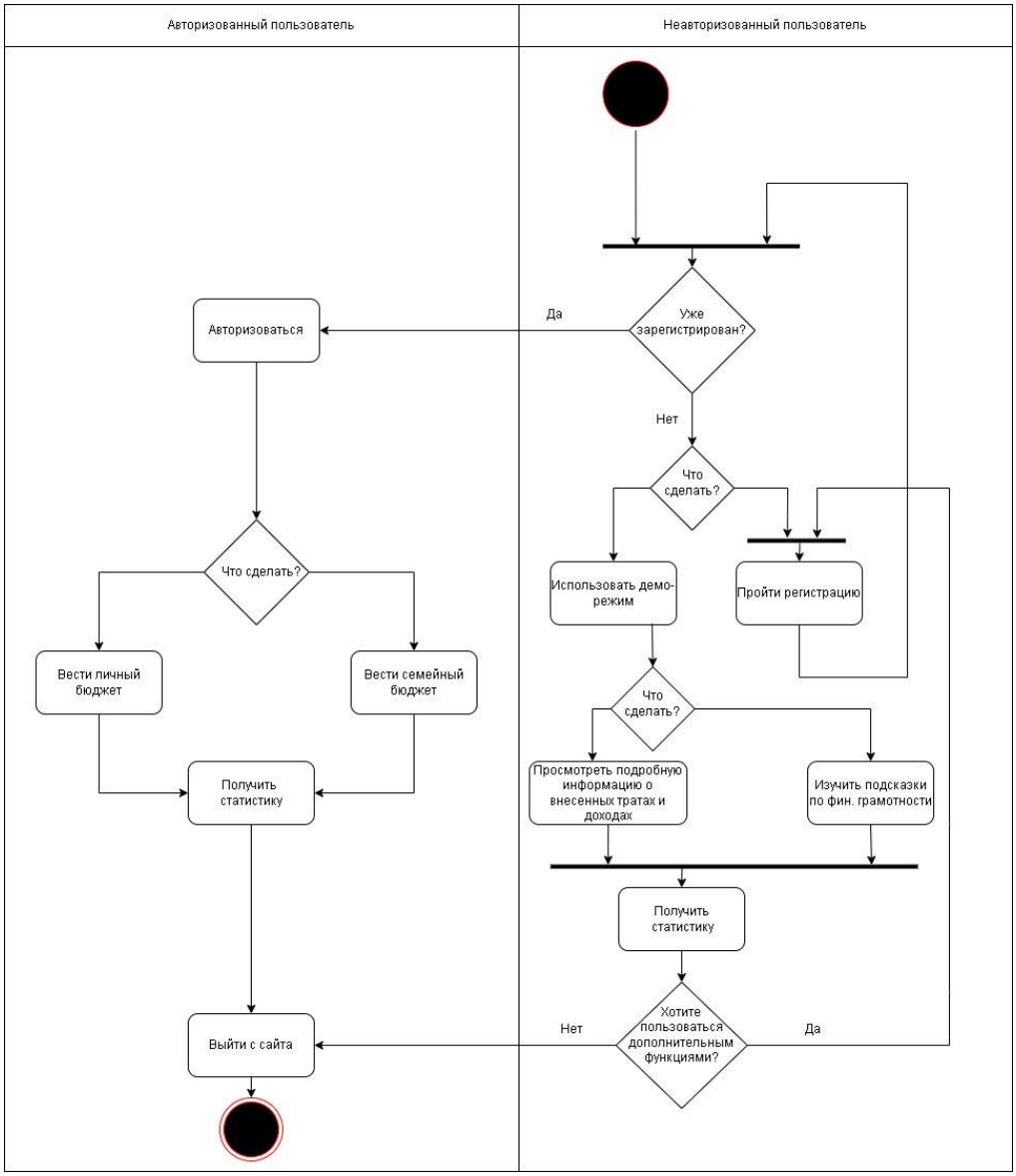


Рисунок 17 - Диаграмма активностей

3.7 Диаграмма развертывания

Диаграмма развертывания показывает архитектуру исполнения системы, включая такие узлы, как программные среды исполнения, а также промежуточное программное обеспечение, соединяющее их (рисунок 18).

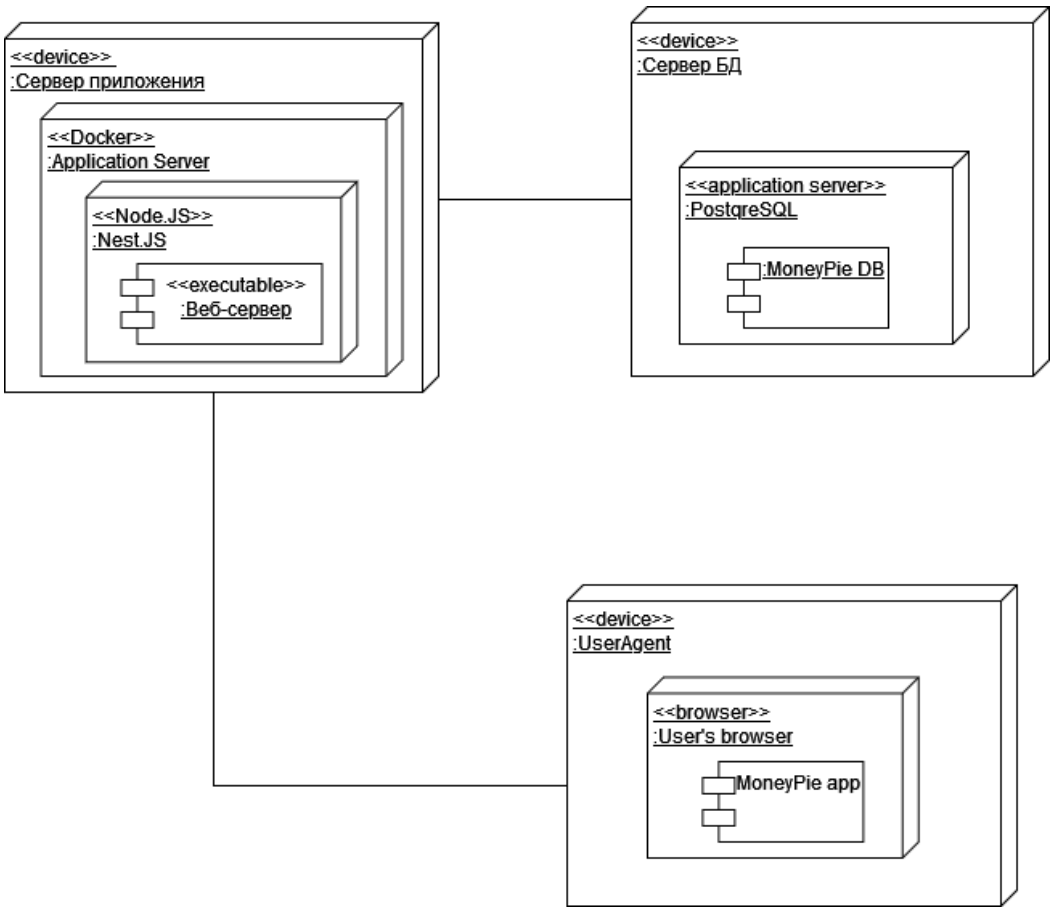


Рисунок 18 - Диаграмма развертывания

3.8 ER-диаграмма базы данных

ER-диаграмма базы данных используется для моделирования и визуализации связей между сущностями в базе данных. Она помогает определить структуру базы данных, идентифицировать сущности, их атрибуты и связи между ними. ER-диаграмма предоставляет общее представление о данных, которые будут храниться в базе данных, и служит основой для создания схемы базы данных (рисунок 19).

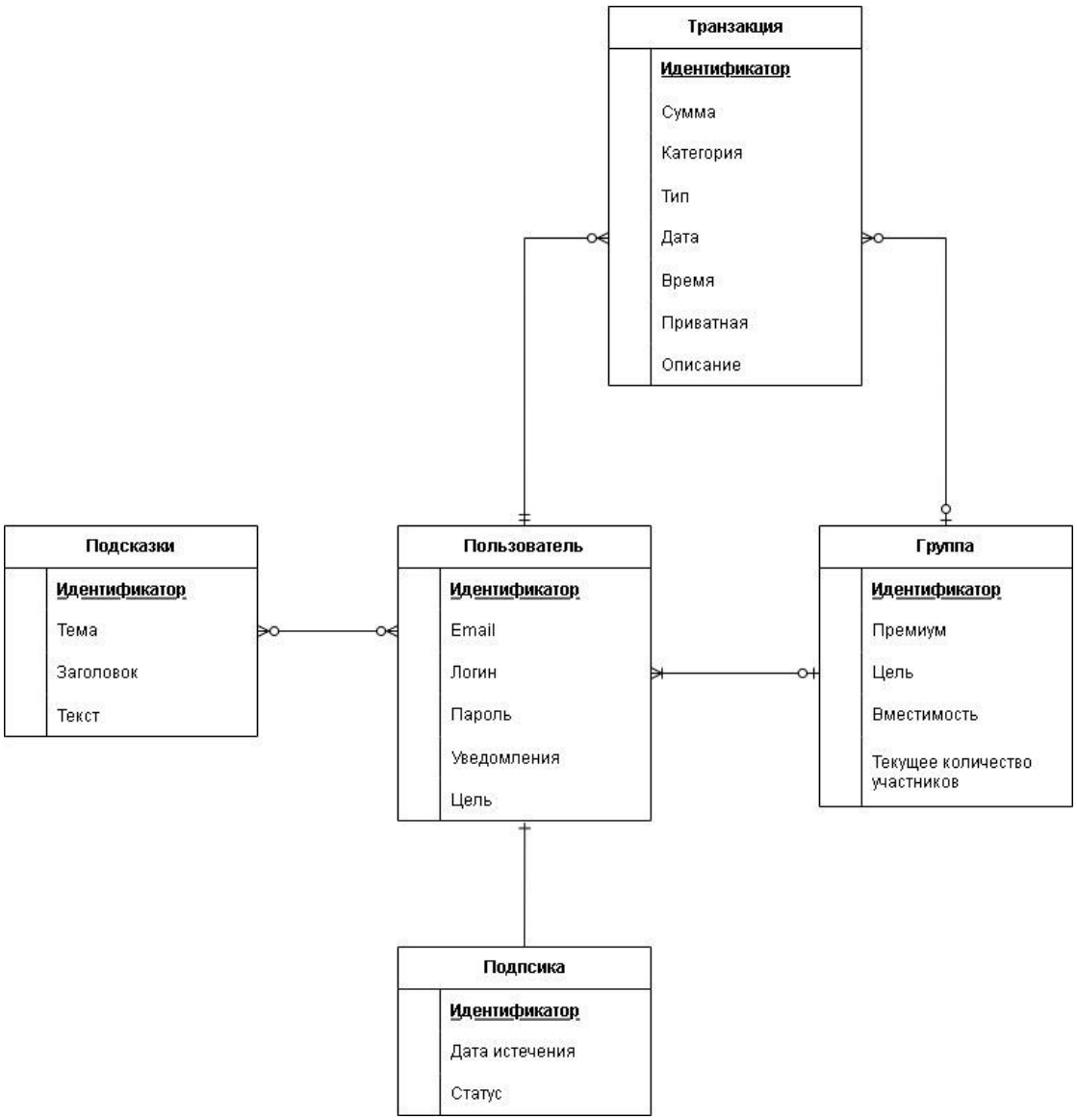


Рисунок 19 - ER-диаграмма базы данных

3.9 Физическая схема базы данных

Физическая схема базы данных, с другой стороны, определяет способ физической организации данных в базе данных. Она включает в себя информацию о таблицах, столбцах, индексах, ограничениях и других структурах, необходимых для физического хранения данных. Физическая схема базы данных обычно учитывает производительность, оптимизацию запросов и другие аспекты, связанные с физическим хранением данных (рисунок 20).

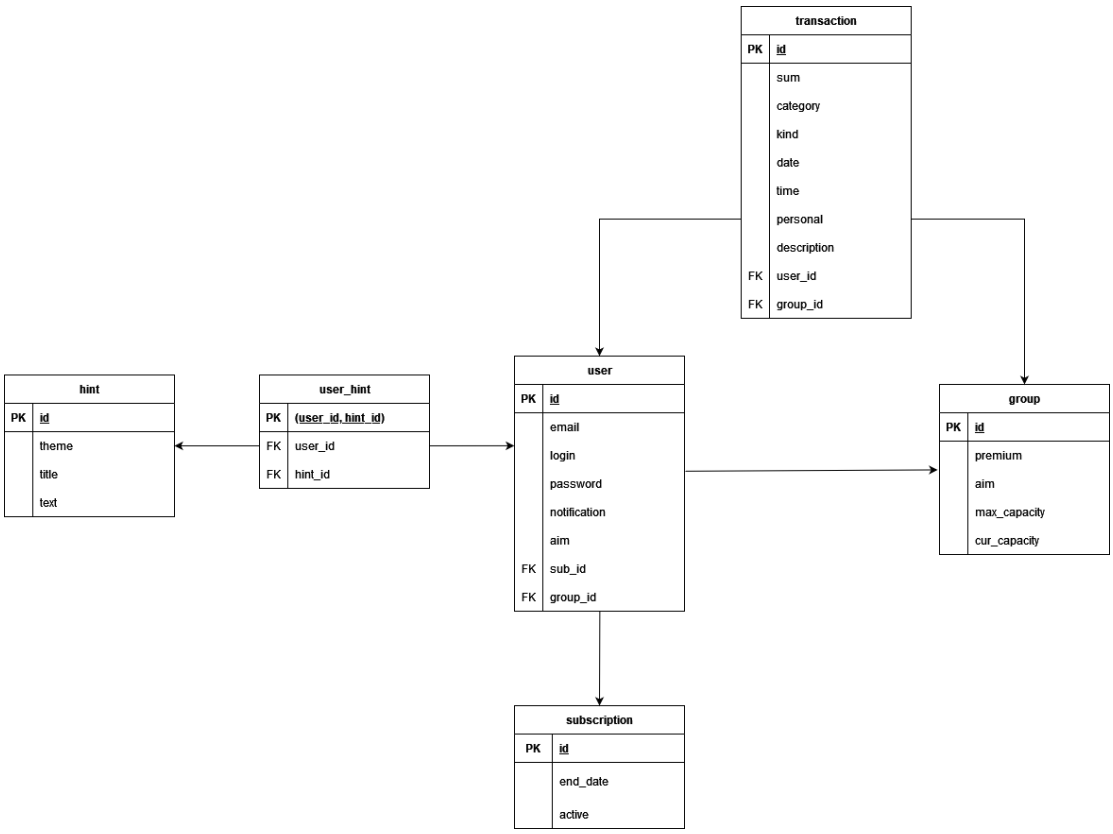


Рисунок 20 - Физическая схема базы данных

4 Реализация

4.1 Разработка frontend

При разработке клиентской части приложения использованы следующие технологии:

- JavaScript (JS) – язык программирования, который выполняется внутри браузера и позволяет внедрять в сайт различные функции на стороне клиента [1];
- TypeScript (TS) – язык программирования, представленный Microsoft и позиционируемый как средство разработки веб-приложений, расширяющее возможности JavaScript. TypeScript предлагает статическую типизацию, что позволяет выявить и предотвратить ошибки типизации на этапе разработки. Язык был выбран, чтобы создать более надежный код, уменьшить количество ошибок и упростить сопровождение и масштабирование приложения [1];
- React – библиотека с открытым исходным кодом, позволяющая создавать одностраничные приложения. Работает как с JS, так и с TS. Библиотека основана на компонентном подходе, что позволяет разбить интерфейс на небольшие, переиспользуемые компоненты, облегчая разработку и поддержку кода;
- Next.js является фреймворком для разработки веб-приложений на базе React [2]. Он предоставляет мощные инструменты для серверного рендеринга, статической генерации и управления маршрутизацией, что повышает производительность и SEO-оптимизацию веб-сервиса.

Разработка React-приложений с использованием Next.js предлагает ряд особенностей и преимуществ, которые способствуют более эффективной и производительной разработке [3]. Например, благодаря Next.js можно предварительно сгенерировать статические HTML-страницы для улучшения производительности и кэширования. Также Next.js имеет встроенные

инструменты для управления маршрутизацией, что предоставляет возможность создавать страницы с помощью файловой системы, размещая React-компоненты в папках с соответствующими именами. Это упрощает организацию и добавление новых страниц в приложение.

Базовая структура приложения, разработанного с использованием Next.js и React, обычно имеет следующие основные элементы:

- pages: в этой папке размещаются файлы с React-компонентами, которые представляют собой отдельные страницы. Каждый файл в этой папке автоматически становится маршрутом и доступен по соответствующему URL-адресу;
- components: в этой папке содержатся компоненты, которые используются на разных страницах. Здесь содержатся как общие элементы страниц, такие как кнопки, заголовки и модальные окна, так и отдельные модули, то есть готовые компоненты-шаблоны, созданные из других компонентов;
- styles: здесь размещаются файлы со стилями приложения, такие как CSS-файлы или файлы стилей для CSS-in-JS библиотек;
- public: в этой папке находятся статические ресурсы, такие как изображения, иконки и другие файлы, которые будут доступны напрямую из браузера.

На рисунке 21 изображена структура веб-приложения на клиентской стороне.

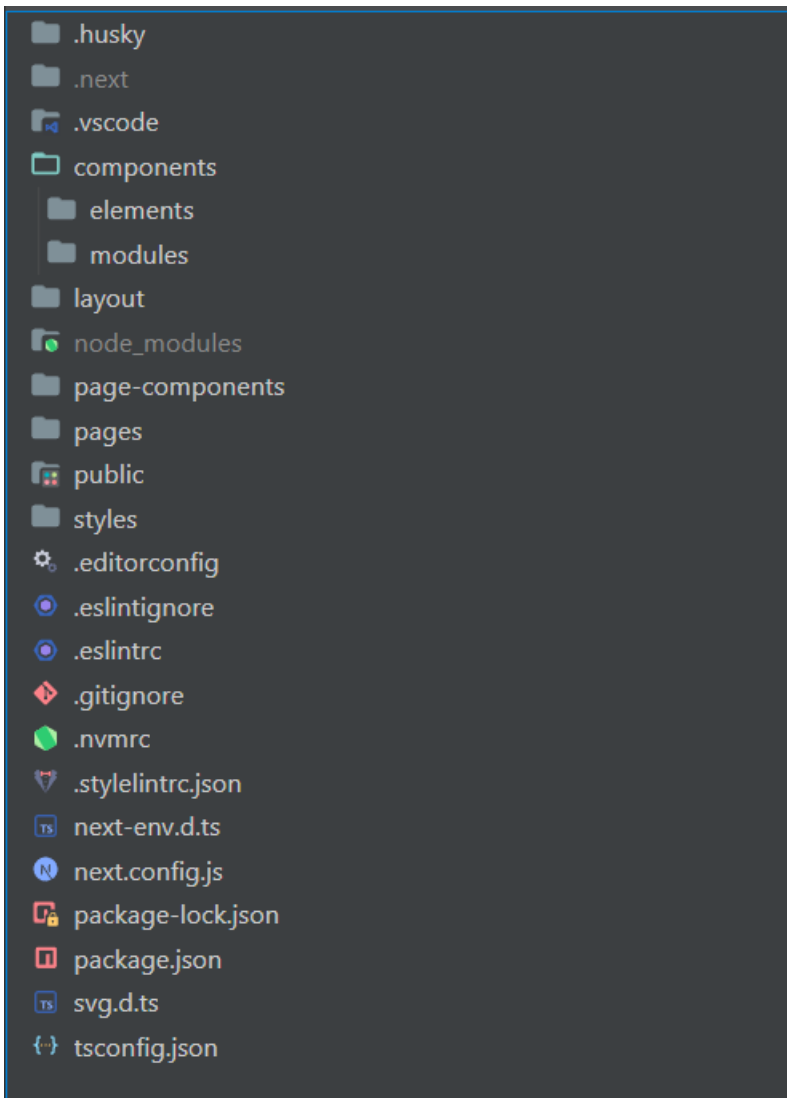


Рисунок 21 - Структура React-приложения

Здесь помимо элементов, которые были описаны выше, можно заметить также: layout, page-components и различные файлы конфигурации.

Папка layout содержит компоненты, отвечающие за общую структуру и компоновку страниц приложения. В ней находятся шапка и боковая панель, которые присутствуют на каждой странице веб-сервиса (рисунок 22).

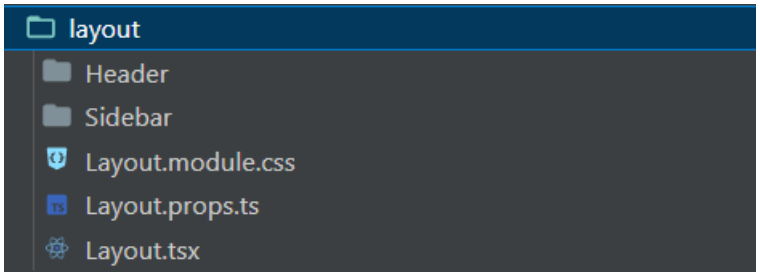


Рисунок 22 - Структура папки layout

Папка page-components содержит компоненты, специфичные для определенных страниц. Это как уникальные компоненты, которые предназначены для определенных разделов приложения, так и компоненты, имеющие особую логику и отображение на определенных страницах (рисунок 23).

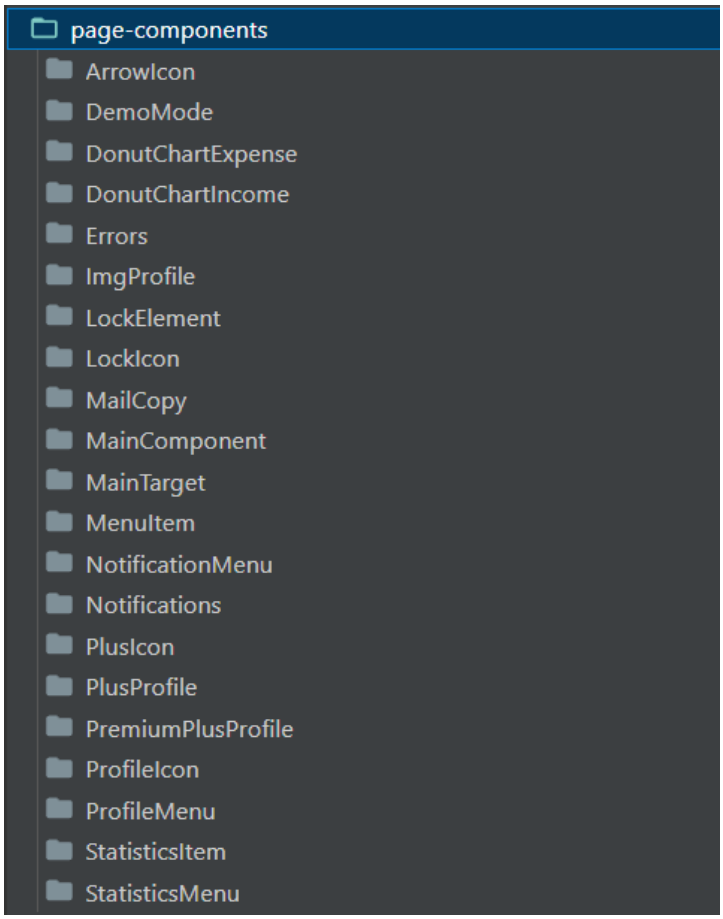


Рисунок 23 - Структура папки page-components

Файлы конфигурации в данной структуре используются для определения и настройки различных аспектов разработки и сборки веб-приложения. Например, файлы «.editorconfig» и «.eslintrc» служат для поддержания единого стиля кодирования и обнаружения потенциальных проблем в JavaScript или TypeScript коде. Файлы «next.config.js» и «package.json» позволяют настроить и управлять зависимостями, скриптами и другими параметрами проекта, в то время как файлы «.gitignore» и «.nvmrc»

облегчают работу с системой контроля версий Git и установкой нужной версии Node.js, используемый при разработки серверной части приложения, соответственно.

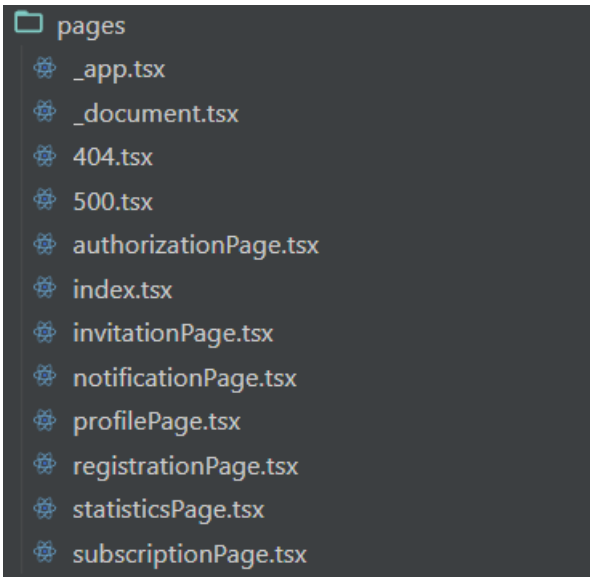


Рисунок 24 - Содержимое папки pages

На рисунке 24 представлены страницы веб-приложения. Файл index.tsx является основной страницей (главным экраном). Он представляет собой React-компонент, который отображается при доступе к корневому URL-адресу приложения.

Страница index.tsx служит точкой входа для пользователя, где он получает общую информацию о приложении и может перейти на другие страницы или разделы веб-сервиса.

4.2 Разработка backend

При разработке серверной части приложения использованы следующие технологии:

– Основной язык – TypeScript, исполняемый на сервере в среде Node.js в связке с фреймворком Nest.js. Nest.js – это платформа для создания серверных приложений на Node.js. Сам фреймворк построен с использованием TypeScript и полностью поддерживает его (при

этом позволяет разработчикам использовать чистый JavaScript). Позволяет реализовать MVC архитектуру;

– Приложение будет оперировать реляционной БД, в качестве СУБД будет использоваться СУБД с открытым исходным кодом PostgreSQL [4];
– В качестве инструмента развертки приложения будет использоваться Docker, который позволяет автоматизировать процесс развертывания и управления приложениями [5];
– Для документации разрабатываемого REST API будет использоваться Swagger, предоставляющий набор инструментов, который позволяет автоматически описывать API на основе его кода [6].

Структура приложения приведена на рисунке 25

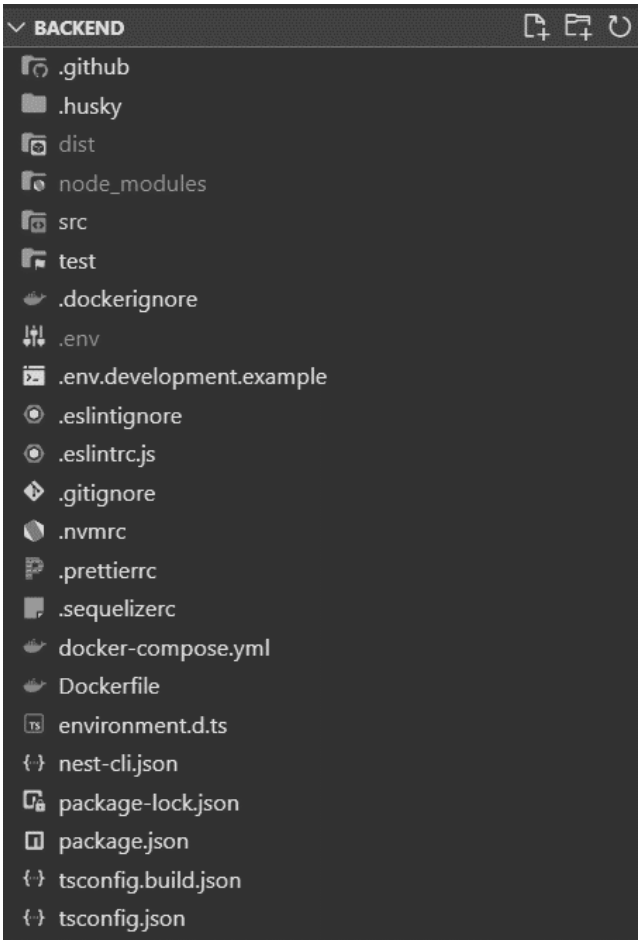


Рисунок 25 - Структура серверной части приложения

Иерархия пакетов модулей изображена на рисунке 26.

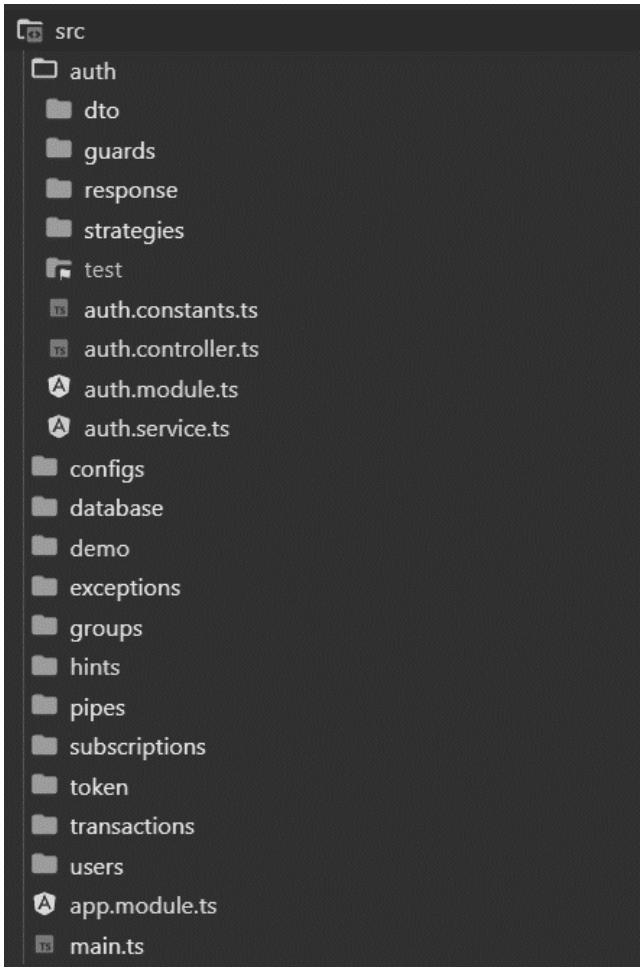


Рисунок 26 - Иерархия пакетов модулей

Структура проекта включает конфигурационные файлы и директории, которые содержат модели, контроллеры, сервисы и другие компоненты, относящиеся к различным функциональным областям, таким как аутентификация, авторизация, обработка запросов, база данных, исключения и т.д.

Например, основные конфигурационные файлы:

- «.env.development.example» содержит переменные окружения, которые используется в разработке.;
- «.sequelizerc» включает в себя конфигурацию для Sequelize as ORM для работы с базой данных;

– «nest-cli.json» предоставляет конфигурацию для инструмента командной строки Nest.js, который позволяет управлять проектом;
– «tsconfig.json» определяет настройки компилятора TypeScript.

Все эти файлы помогают настроить и управлять различными аспектами проекта.

Файл, в котором описывается инструкции для сборки образа Docker, содержащего все необходимое для запуска приложения – «Dockerfile».

Файл, в котором находится конфигурация для развертывания приложения с использованием Docker Compose – «docker-compose.yml». Здесь указываются сервисы, сети, тома и другие настройки.

Основной модуль приложения, в котором объединяются все модули, контроллеры, провайдеры и другие компоненты – «app.module.ts».

Основной файл приложения, который запускает сервер Nest.js и настраивает его конфигурацию – «main.ts».

На примере модуля «auth» рассмотрим содержание пакетов:

– «dto» содержит объекты передачи данных (DTO – Data Transfer Objects) для аутентификации, авторизации и связанных операций;
– «response» имеет объекты ответов (response) для аутентификации и авторизации, которые используются для формирования ответов API;
– «strategies» содержит стратегии аутентификации, которые определяют, как будет выполняться процесс аутентификации (например, через JWT, OAuth, Passport и т. д.);
– «auth.constants.ts» включает в себя константы, связанные с аутентификацией и авторизацией, такие как типы стратегий аутентификации, роли пользователей и другие важные значения.

Механизмы, используемые для защиты информации и обеспечения целостности данных в приложении – гварды (Guards) и валидационные пайпы (ValidationPipes)

Гварды (Guards) предоставляют механизмы для контроля доступа к определенным ресурсам или действиям в приложении. Они могут проверять различные условия, такие как аутентификация пользователя, авторизация для выполнения определенных операций или проверка прав доступа. Гварды позволяют запретить доступ, перенаправить пользователя на другую страницу или выполнить другие действия в зависимости от результатов проверки условий. Они помогают обеспечить безопасность в различных частях приложения.

Валидационные пайпы (ValidationPipes) используются для проверки и валидации входных данных, прежде чем они будут использованы в приложении. Они могут выполнять проверку формата данных, обязательности полей, проверку диапазона значений и другие проверки, чтобы гарантировать, что данные соответствуют заданным требованиям.

Сервер был развернут (загружен) на хостинге AdminVPS. База данных находится на сервере. После создания базы данных были предоставлены данные для подключения, такие как хост, порт, имя базы данных, имя пользователя и пароль. Эта информация используется для настройки подключения к базе данных из приложения. Пароли пользователей хранятся в базе данных в хэшированном виде с помощью хэш-функции из библиотеки «bcrypt».

5 Навигация по приложению

5.1 Главный экран

При первом запуске приложения пользователь попадает на главный экран в демо-режиме, после чего может нажать на кнопку «Зарегистрироваться» и перейти на экран регистрации (рисунок 27).

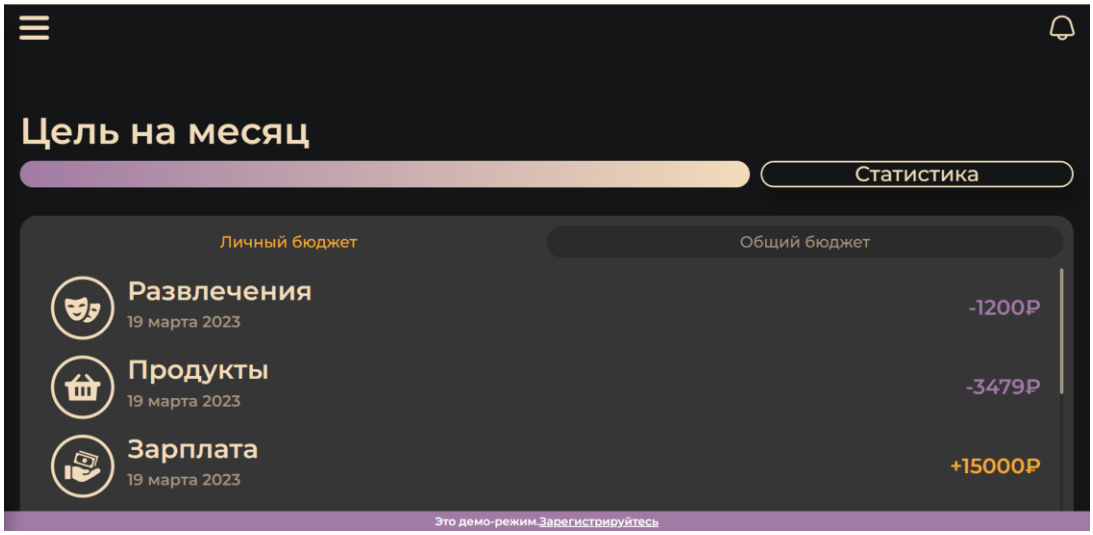


Рисунок 27 - Главный экран в демо-режиме

Если пользователь прошел регистрацию или авторизацию, то попадает на главный экран со всей доступной функциональностью (например, нажатие на «Плюс» в нижнем углу экрана для добавления трат или поступлений. Отсюда можно перейти на другие страницы приложения, нажав на «Бургер-меню» в левом углу header, при котором открывается боковое меню с разделами веб-сервиса, или нажать на «Центр уведомлений» в правом углу header для перехода на экран уведомлений (рисунки 28-29).

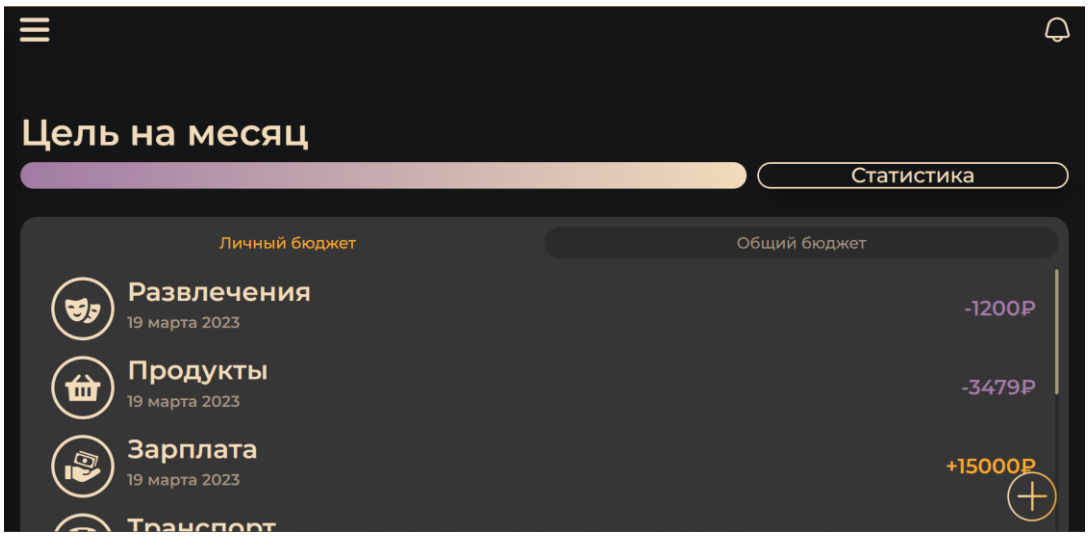


Рисунок 28 - Главный экран

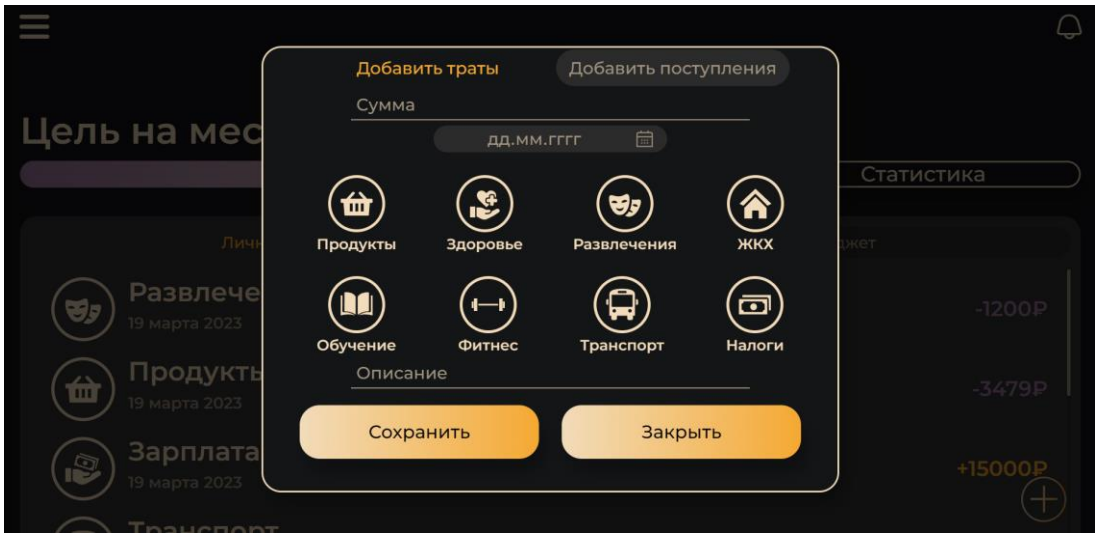


Рисунок 29 - Добавление трат и поступлений

5.2 Экран авторизации

На рисунке 30 показан экран авторизации.

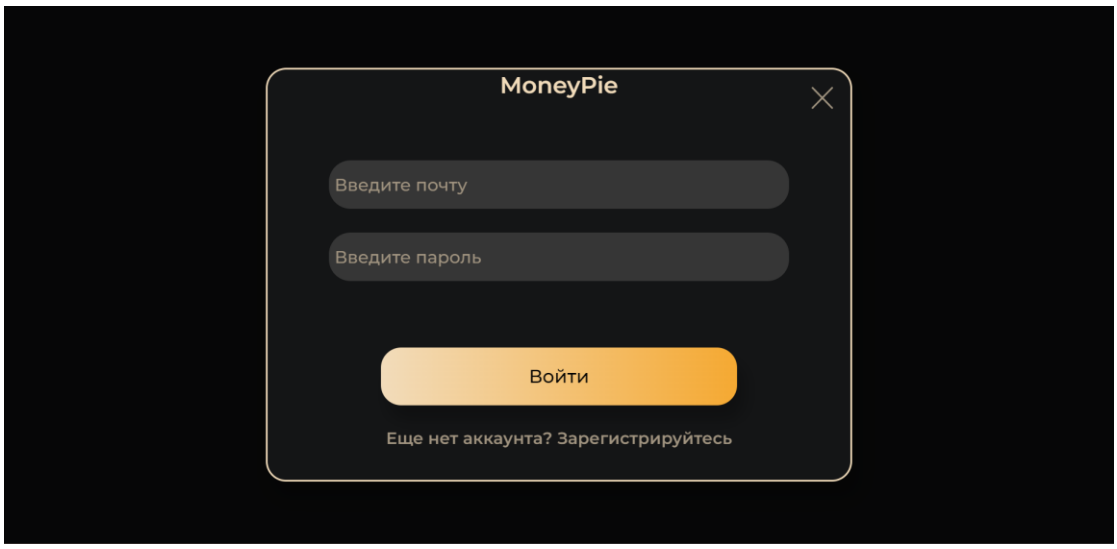


Рисунок 30 - Экран авторизации

5.3 Экран регистрации

На рисунке 31 изображен экран регистрации.

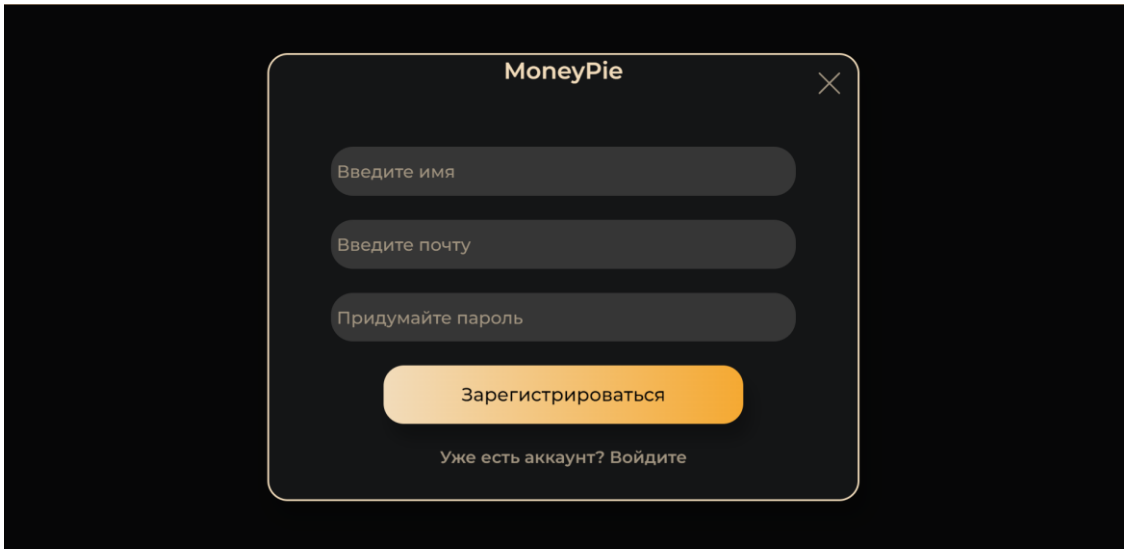


Рисунок 31 - Экран регистрации

5.4 Боковое меню

При нажатии на «Бургер-меню» в header открывается меню с разделами приложение, такие как «Профиль», «Главная страница», «Статистика». В нижней части бокового меню можно нажать на кнопку «Узнать о приложении», чтобы перейти на страницу с информацией о проекте, а также оформить премиум-подписку (рисунок 32).

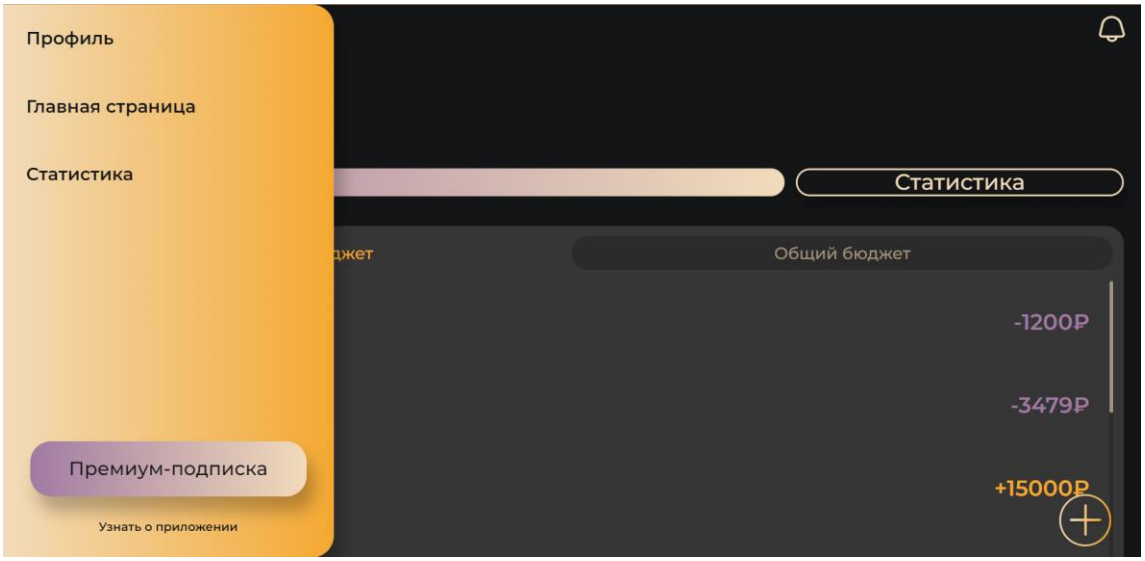


Рисунок 32 - Боковое меню

5.5 Экран профиля

На экране профиля можно просмотреть информацию о пользователе, пригласить друга, отключить уведомления, управлять подпиской, выйти из группы и выйти из аккаунта (рисунок 33).

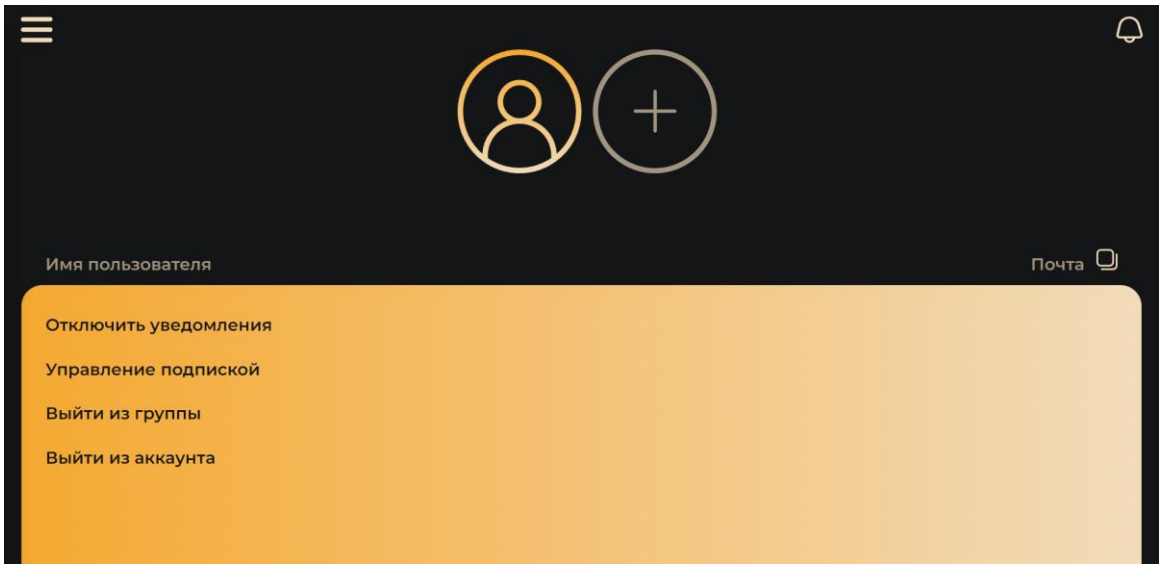


Рисунок 33 - Экран профиля

5.6 Экран уведомлений

На данном экране можно просмотреть подсказки и советы по финансовой грамотности (рисунок 34).

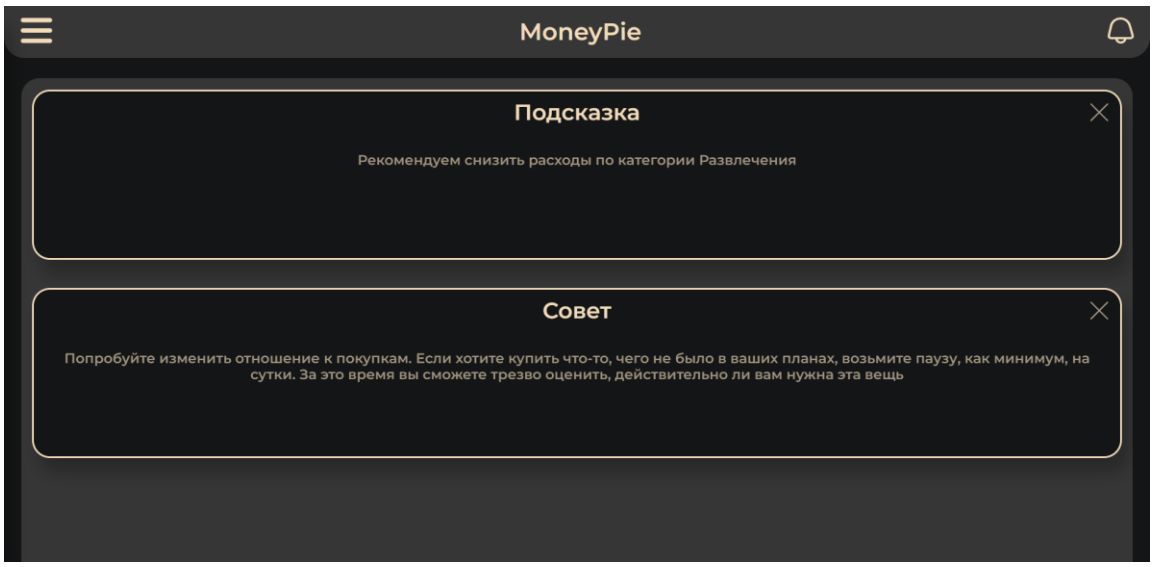


Рисунок 34 - Экран уведомлений

5.7 Экран со статистикой

На экране со статистикой можно просмотреть график расходов и доходов по категориям (рисунок 35-36), а также развернуть нижнее меню с подробной информацией (рисунок 37).

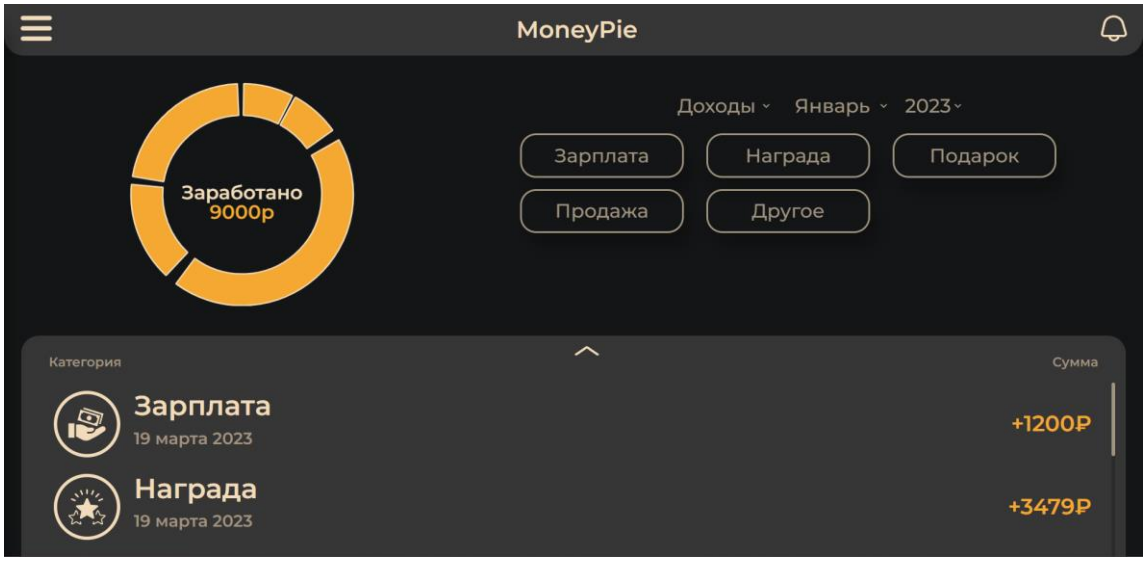


Рисунок 35 - Экран со статистикой по доходам

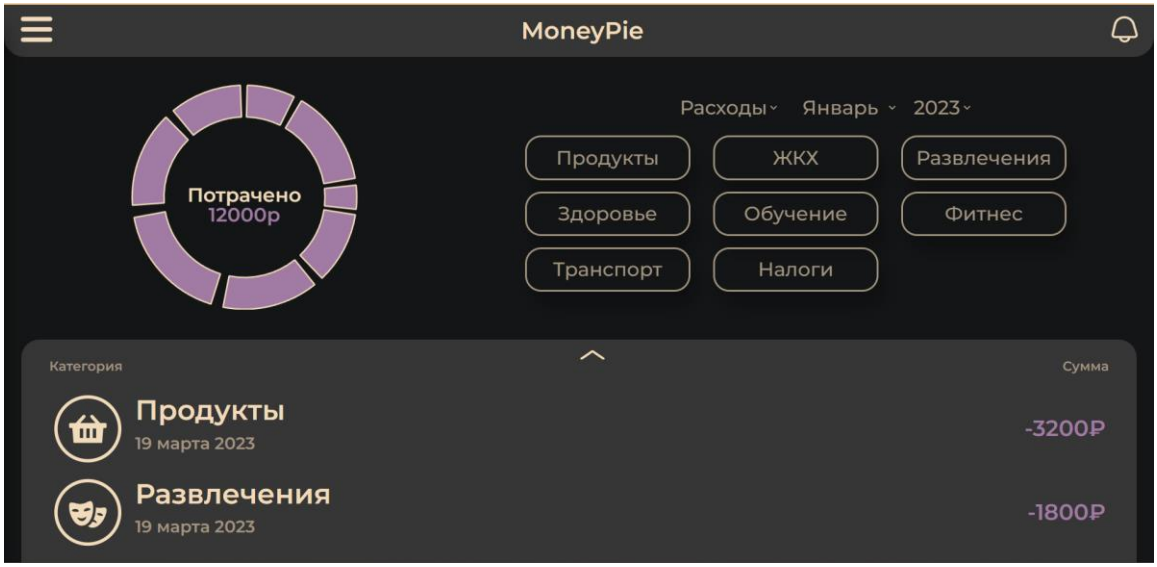


Рисунок 36 - Экран со статистикой по расходам

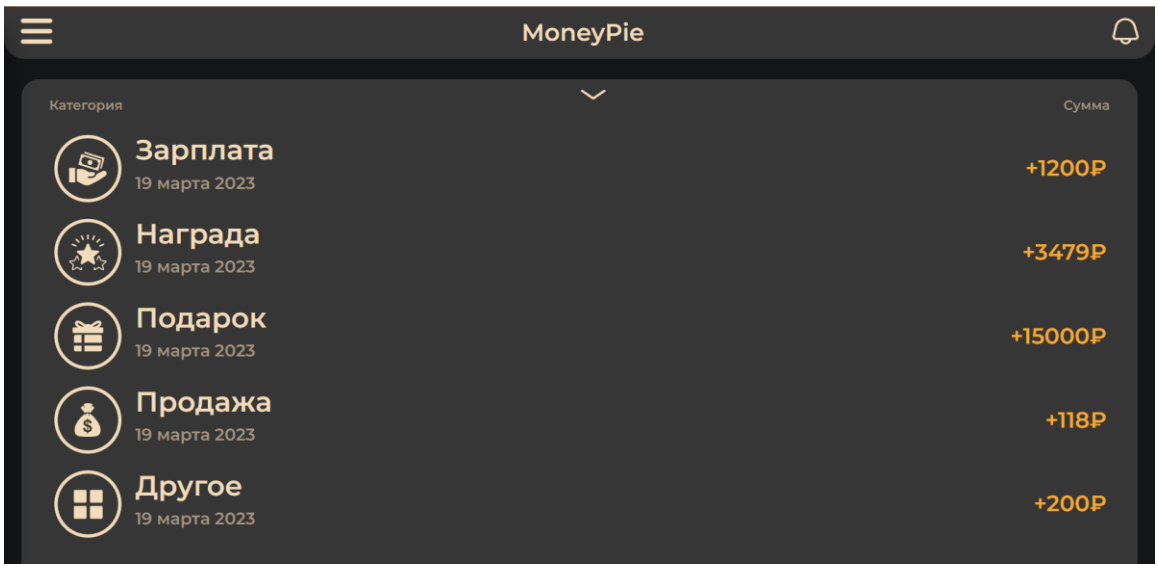


Рисунок 37 - Меню с подробной информацией

5.8 Экран приглашения друга

На экране приглашения друга можно ввести почту пользователя, которого нужно пригласить, а также нажать на кнопку «Отправить», чтобы отправить запрос на приглашение (рисунок 38).

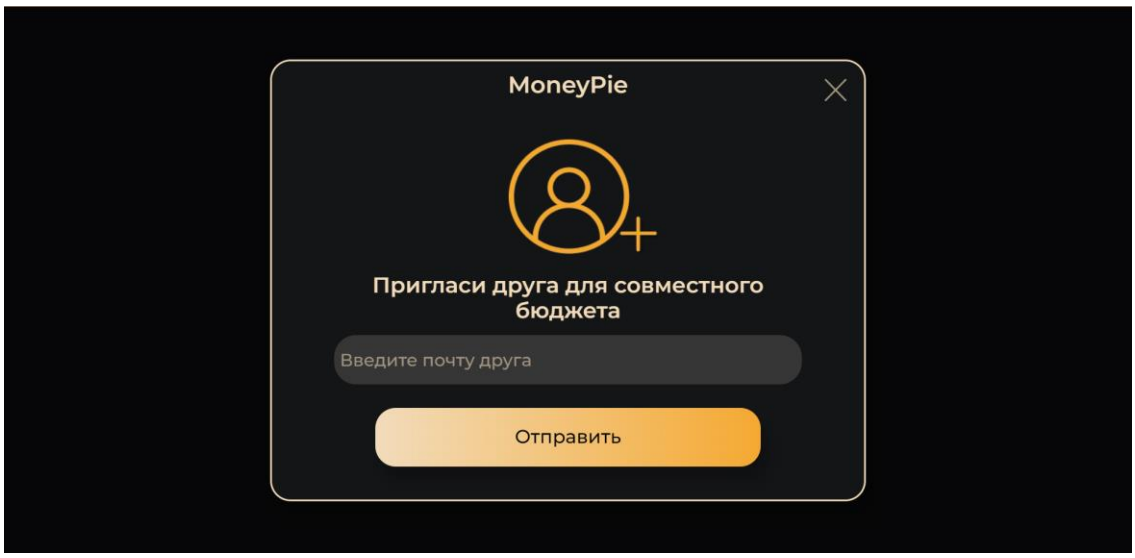


Рисунок 38 - Экран приглашения друга

5.9 Экран премиум-подписки

На экране премиум-подписки можно оформить подписку, нажав на кнопку «Купить» или оформить позднее, нажав на кнопку «Оформить подписку позже» (рисунок 39).

Рисунок 39 - Экран премиум-подписки

6 Тестирование

При проведении тестирования разработанной системы были охвачены следующие типы тестов:

- Дымовое тестирование;
- Модульное тестирование (unit-тесты);
- Функциональные тесты (тестирование пользовательского интерфейса);
- Нефункциональные тесты (usability тестирование).

6.1 Дымовое тестирование

Дымовое тестирование выполняется для проверки основных функций и стабильности программного обеспечения после его обновления или внесения значительных изменений.

Цель дымового тестирования заключается в том, чтобы убедиться, что основные функции приложения или системы работают должным образом и не возникают критические ошибки после внесения изменений. Это важный этап тестирования, который проводится перед более подробными и глубокими тестами, чтобы выявить наиболее критические проблемы в короткие сроки.

При проведении дымового тестирования выполняются базовые сценарии использования или тестовые случаи, которые охватывают основные функции и ключевые пути в приложении. Тесты обычно являются автоматизированными и могут включать такие проверки, как запуск приложения, проверка отображения интерфейса, выполнение базовых операций и проверка результата.

В таблице 2 представлены результаты дымового тестирования для основных сценариев.

Таблица 2 - Результаты дымового тестирования

Тестовый сценарий	Результат теста
Вход в демо-режим	Пройден

Просмотр главной страницы в демо-режиме	Пройден
Просмотр страницы статистики в демо-режиме	Пройден
Просмотр страницы уведомлений в демо-режиме	Пройден
Регистрация	Пройден
Авторизация	Пройден
Просмотр главной страницы	Пройден
Добавление цели на месяц	Пройден
Просмотр личного бюджета	Пройден
Добавление друга в общий бюджет	Пройден
Просмотр общего бюджета	Пройден
Просмотр страницы статистики по доходам	Пройден
Просмотр страницы статистики по расходам	Пройден
Просмотр страницы уведомлений	Пройден
Просмотр страницы профиля	Пройден
Управление подпиской	Пройден
Добавление пользователей в группу	Пройден

6.2 Модульное тестирование

Модульное тестирование (unit-тесты), является методом тестирования программного обеспечения, при котором отдельные модули или компоненты программы проверяются на корректность и работоспособность.

Основная цель модульного тестирования заключается в том, чтобы убедиться, что каждый отдельный модуль программы функционирует должным образом в изоляции от других модулей. Во время модульного тестирования проверяется правильность работы функций, методов, классов или других самостоятельных единиц кода. Это позволяет выявить и исправить ошибки на ранних стадиях разработки, улучшить стабильность и надежность программного продукта.

В структуре серверной части приложения внутри каждого модуля, имеющего контроллер, находится директория «test», содержащая unit-тесты. Так, например, показаны тесты для модулей «auth» и «demo» на рисунке 40.

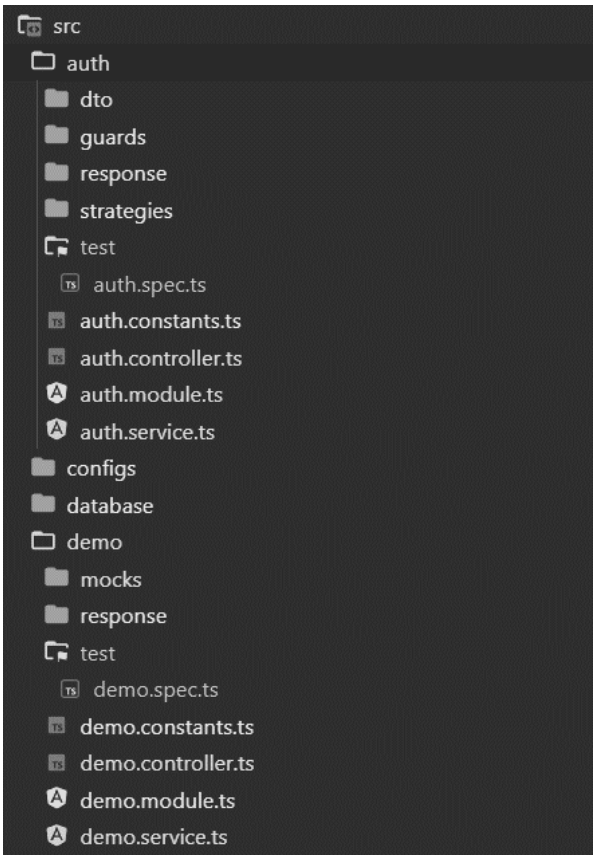


Рисунок 40 - Пример тестов для auth и demo

Ниже на рисунке 41 представлены результаты unit-тестов серверной части приложения для одного из модулей.

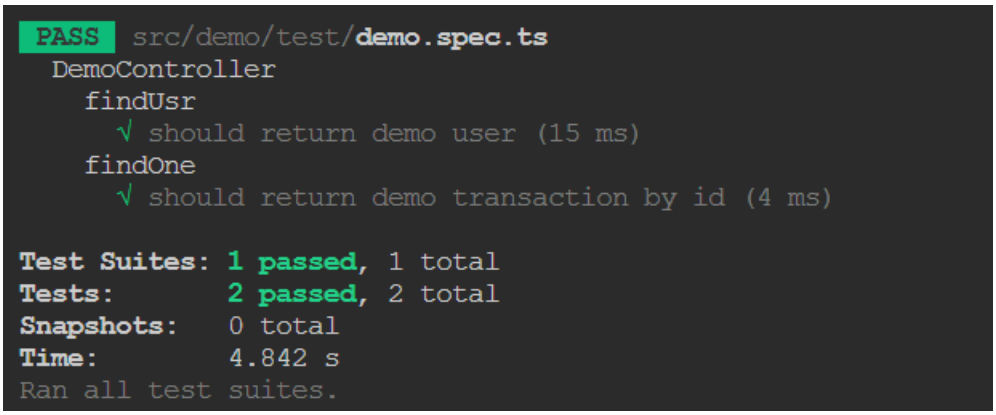


Рисунок 41 - Результаты unit-тестов

6.3 Функциональные тесты

Функциональное тестирование в виде тестирования пользовательского интерфейса (UI тестирование) является процессом проверки соответствия программного обеспечения его функциональным требованиям. Оно фокусируется на проверке работы системы с точки зрения ее внешнего поведения и взаимодействия с пользователем.

Цель функционального тестирования заключается в том, чтобы убедиться, что приложение или система работает в соответствии с заданными функциональными требованиями и ожиданиями пользователей. Это включает проверку правильности работы интерфейса, функциональности приложения, обработку пользовательского ввода, отображение данных и выполнение заданных операций.

В таблице 3 представлена часть результатов тестирования пользовательского интерфейса. В ней отражены тестовые сценарии для неавторизованного пользователя.

Таблица 3 - Результаты UI-тестирования для неавторизованного пользователя в демо-режиме

Тестовый сценарий	Ожидаемый результат	Статус теста
Нажатие на кнопку «Личный бюджет» на главной странице	Вывод тестовых трат и поступлений из базы данных	Пройден
Нажатие на кнопку «Общий бюджет» на главной странице	Вывод заблокированного экрана с кнопкой «Пригласить друга»	Пройден
Нажатие на кнопку «Пригласить друга» в общем бюджете	Переход на страницу регистрации (для демо-режима)	Пройден

Нажатие на кнопку «Зарегистрируйтесь» на баннере демо-режима	Переход на страницу регистрации	Пройден
Нажатие на кнопку «Статистика» на главной странице	Переход на страницу статистики	Пройден
Нажатие на кнопку «Доходы» в выпадающем меню	Вывод информации о доходах	Пройден
Нажатие на кнопку «Расходы» в выпадающем меню	Вывод информации о расходах	Пройден
Нажатие на кнопку с наименованием определенной категории на странице статистики	Выделение области соответствующей категории на графике	Пройден
Нажатие на иконку «Центр уведомлений» в header	Переход на страницу уведомлений	Пройден
Нажатие на иконку «Бургер-меню» в header	Открытие бокового меню	Пройден
Нажатие на кнопку «Профиль» в боковом меню	Переход на страницу регистрации (для демо-режима)	Пройден
Нажатие на кнопку «Главная страница» в боковом меню	Переход на главную страницу	Пройден
Нажатие на кнопку «Статистика» в боковом меню	Переход на страницу статистики	Пройден
Нажатие на ссылку «Узнать о приложении» в боковом меню	Переход на страницу проекта на GitHub	Пройден

Нажатие на ссылку «Уже есть аккаунт? Войдите» на странице регистрации	Переход на страницу авторизации	Пройден
Нажатие на ссылку «Еще нет аккаунта? Зарегистрируйтесь» на странице авторизации	Переход на страницу регистрации	Пройден
Нажатие на кнопку «Зарегистрироваться» на странице регистрации	Переход на главную страницу	Пройден
Нажатие на кнопку «Войти» на странице авторизации	Переход на главную страницу	Пройден

6.4 Нефункциональные тесты

Тестирование удобства пользования (usability тестирование) – это метод тестирования, который используется для оценки удобства, эффективности и удовлетворенности пользователей при взаимодействии с продуктом или системой. Оно направлено на оценку того, насколько легко и интуитивно пользователь может использовать продукт, а также на выявление проблем, с которыми пользователи могут столкнуться в процессе его использования [7].

Основная цель тестирования удобства пользования состоит в том, чтобы обеспечить оптимальный пользовательский опыт и повысить удовлетворенность пользователей. Это помогает разработчикам и дизайнерам идентифицировать слабые места в интерфейсе и функциональности продукта, а также принимать меры для их улучшения.

В таблице 4 представлены результаты тестирования для четырех опрошенных пользователей, перед которыми были поставлены указанные задания. Оценка успешности выполнения задания производилась с помощью подхода Нильсена, где выделяют три значения показателя [8]:

- Если пользователь справился с заданием без проблем — 100% (оценка 1);

– Если у пользователя возникли некоторые проблемы, но он сделал задание — 50% (оценка 0,5);
– Если пользователь не выполнил задание — 0% (оценка 0).

Таблица 4 - Результаты тестирования удобства пользования

Задание	Пользователь 1	Пользователь 2	Пользователь 3	Пользователь 4
Вход в демо-режим	1	1	1	1
Просмотр главной страницы в демо-режиме	1	1	1	1
Просмотр страницы статистики в демо-режиме	1	1	1	1
Просмотр страницы уведомлений в демо-режиме	1	1	1	1
Регистрация	1	0,5	1	1
Авторизация	1	1	1	1
Просмотр главной страницы	1	1	1	1
Добавление цели на месяц	1	1	1	1

Просмотр личного бюджета	1	1	1	1
Добавление друга в общий бюджет	1	1	0,5	1
Просмотр общего бюджета	1	0,5	1	1
Просмотр страницы статистики по доходам	1	1	1	1
Просмотр страницы статистики по расходам	1	1	1	1
Просмотр страницы уведомлений	1	1	1	1
Просмотр страницы профиля	1	1	1	1
Управление подпиской	0,5	1	1	1
Добавление пользователей в группу	0,5	1	1	0,5

Далее рассчитаем среднюю успешность по заданиям:

$$\frac{62 * 1 + 6 * 0{,}5}{68} * 100\% = \frac{65}{68} * 100\% \approx 96\%$$

В результате тестирования удобства пользования приложения была выявлена высокая успешность выполнения заданий, составляющая 96%. Это свидетельствует о том, что пользователи с легкостью осуществляют необходимые действия в приложении и успешно достигают своих целей. Такой высокий уровень успешности говорит о хорошей проработке интерфейса и интуитивной понятности функциональности приложения. Это позволяет утверждать, что приложение обладает высоким уровнем удобства пользования, что является важным фактором для обеспечения положительного пользовательского опыта.

Заключение

В заключении данной курсовой работы был рассмотрен процесс разработки веб-приложения для мониторинга личного и совместного бюджета. Целью работы было создание коммерческого приложения, которое будет полезно для широкого круга пользователей. В ходе выполнения проекта были выполнены следующие задачи:

- Проектирование и развертывание базы данных;
- Разработка frontend части веб-приложения;
- Реализация бизнес-логики на сервере;
- Реализация связи между клиентом и сервером с применением подхода REST API;
- Развертывание приложения на серверной части.

Веб-приложение предоставляет пользователю удобный и систематизированный способ учета расходов и доходов, а также возможность отслеживать статистику по каждой категории трат. Основная функциональность приложения включает:

- Регистрацию нового пользователя;
- Авторизацию;
- Формирование и просмотр статистики по расходам;
- Возможность добавления пользователя для ведения совместного бюджета;
- Запись трат и входящих денежных средств по категориям.

Для реализации данного приложения были использованы современные технологии веб-разработки, включая React, Node.js и PostgreSQL. Благодаря этим технологиям удалось создать удобный и функциональный интерфейс, обеспечить безопасность данных пользователей и эффективную работу веб-приложения.

В результате данной работы было создано веб-приложение, которое соответствует поставленным целям и задачам. Оно предоставляет

пользователям возможность контролировать свои финансы, управлять личным и совместным бюджетом, а также анализировать свои расходы и доходы. Приложение имеет потенциал для полноценного коммерческого использования и может быть полезным инструментом для многих пользователей, помогая им улучшить свою финансовую грамотность и достичь финансовых целей.

Список использованных источников

1. MDN Web Docs — JavaScript [Электронный ресурс]. — Режим доступа: https://developer.mozilla.org/en-US/docs/Web/JavaScript/Guide. — Заглавие с экрана. — (Дата обращения: 20.04.2023).
2. Next.js Documentation: [Электронный ресурс]. — Режим доступа: https://nextjs.org/docs. — Заглавие с экрана. — (Дата обращения: 25.04.2023).
3. Что такое React и как его освоить? [Электронный ресурс]. — Режим доступа: https://academy.yandex.ru/journal/chto-takoe-react-i-kak-ego-osvoit. — Заглавие с экрана. — (Дата обращения: 29.04.2023).
4. Система управления объектно-реляционными базами данных PostgreSQL [Электронный ресурс]. — Режим доступа: https://web-creator.ru/technologies/webdev/postgresql. — Заглавие с экрана. — (Дата обращения: 2.05.2023).
5. Моуэт, Э. Использование Docker: Разработка и внедрение программного обеспечения при помощи технологии контейнеров. — Перевод с английского. — Москва: Издательство: «ДМК Пресс», 2019. - 156 с.
6. Тестирование API с помощью Swagger: особенности и преимущества [Электронный ресурс]. — Режим доступа: https://blog.ithillel.ua/ru/articles/api-testing-with-swagger. — Заглавие с экрана. — (Дата обращения: 14.05.2023).
7. Вигерс, К., Битти, Дж. Разработка требований к программному обеспечению: Третье дополненное издание. Перевод с английского. — Москва: Издательство: «Русская редакция», 2014. - 120 с.
8. О формуле Нильсена и вероятности [Электронный ресурс]. — Режим доступа: https://v-shliachkov.medium.com/. — Заглавие с экрана. — (Дата обращения: 27.05.2023).