ВВЕДЕНИЕ

В наше время информационный прогресс достиг немыслимых масштабов и продолжает развиваться семимильными шагами. Новые технологии появляются в нашей жизни для автоматизации каких-либо процессов в каждой сфере жизнедеятельности, что облегчает дальнейшую работу.

На сегодняшний день проходит очень много маркетинговых кампаний, рекламирующих сферу инвестирования в такие активы как акции, облигации и т.д. Компании, представляющие свои брокерские услуги, ориентируются на удобство пользования их информационных систем (далее ИС) и качество своего продукта. Данные ИС решают следующий ряд задач:

- Возможность инвестирования в акции, облигации и другого рода активы со своего смартфона, ноутбука, персонального компьютера (далее ПК);
- Поиск интересующих компаний, в которые можно вложить свои средства;
- Мониторинг новостей и отчётностей компании;
- Покупка дополнительных тарифов для совершения более крупных сделок и покупки активов на иностранных биржах.

В совокупности данные решения открывают перед пользователем новые возможности в инвестировании. Достаточно открыть счёт у брокера, внести депозит и начать совершать сделки через своего брокера.

Сюда же входит и сфера инвестирования в криптовалюты. В связи с популяризацией данного направления, всё больше денег притекает в проекты, выпускающие свою криптовалюту и всё больше появляется бирж, предоставляющих услуги «брокера» в данной сфере.

Актуальность темы инвестирования в криптовалютные проекты заключается в том, что у разных компаний могут быть серьёзные условия для участия в поддержке их продукта. В каких-то проектах порог входа (суммы

минимально возможного инвестирования) идёт от 5.000-50.000$ и выше; где-то нужно быть резидентом заграничной страны, то есть, хотя бы иметь вид на жительство в определённой стране. Не каждый может позволить себе такую роскошь, особенно если речь идёт об инвестировании.

Примерами данных мероприятий являются ICO (Initial Coin Offering) – первичное размещение токенов/монет [23]. Это по своей сути привлечение денежных средств в какую-либо криптовалюту до её размещения на биржах. Из-за того, что на ICO активы продаются по заниженной цене, то условия для участия в данной сфере могут быть завышены для разных слоёв общества.

Поэтому было принято решение разработать ИС, объединяющую средства пользователей системы в один общий пул и предназначенную для дальнейшего инвестирования исключительно в криптовалютные проекты.

Целью курсового проекта является разработка информационной системы краудинвестинговой платформы "Invest Inspector".

Были определены следующие задачи, которые необходимо решить в ходе курсового проектирования: исследовать предметную область; провести аналитический обзор существующих решений; определить стек используемых при разработке технологий; спроектировать и разработать структуру базы данных; спроектировать и разработать структуру системы и алгоритмы работы её компонентов; реализовать приложение; провести тестирование и отладку.

В итоге курсового проектирования необходимо создать систему, практическая значимость которой заключается в предоставлении пользователю (далее участнику) выбора проекта для инвестирования, сборе средств участников и дальнейшем инвестировании собранных денег. Система должна предоставлять пользователю поиск актуальных проектов; регистрировать его, как участника; давать возможность отправлять заявки на участие в проектах; предоставлять готовое решение по инвестированию; давать возможность инвестировать средства в проекты.

1. ОБЩИЙ РАЗДЕЛ

1.1 Исследование предметной области

В мире инвестиций существует много инструментов, которые помогают вкладчикам на раннем этапе разобраться со всеми аспектами данного направления.

Анализ задачи - неотъемлемая часть в разработке проекта. С его помощью описывают пошаговую инструкцию, необходимую выполнить в ходе создания информационной системы. Определение задачи даёт понимание какие функции будет иметь система, каким образом будет разбита на функциональные модули, какие роли пользователей будет иметь для удовлетворения всех поставленных требований. Поэтому анализ задачи даёт огромное преимущество в реализации информационной системы или любого другого проекта.

Проанализировав поставленную задачу, было принято решение отобразить основные аспекты системы в виде UML - диаграмм.

UML (англ. Unified Modeling Language - унифицированный язык моделирования) - язык графического описания для объектного моделирования в области разработки программного обеспечения; система обозначений, которую можно применять для объектно-ориентированного анализа и проектирования [27].

На основе анализа предметной области, были определены роли, взаимодействующие с системой. Каждая роль выполняет определённую группу функций. Результат анализа представлен в виде диаграммы прецедентов, изображённой на рисунке 1.1.

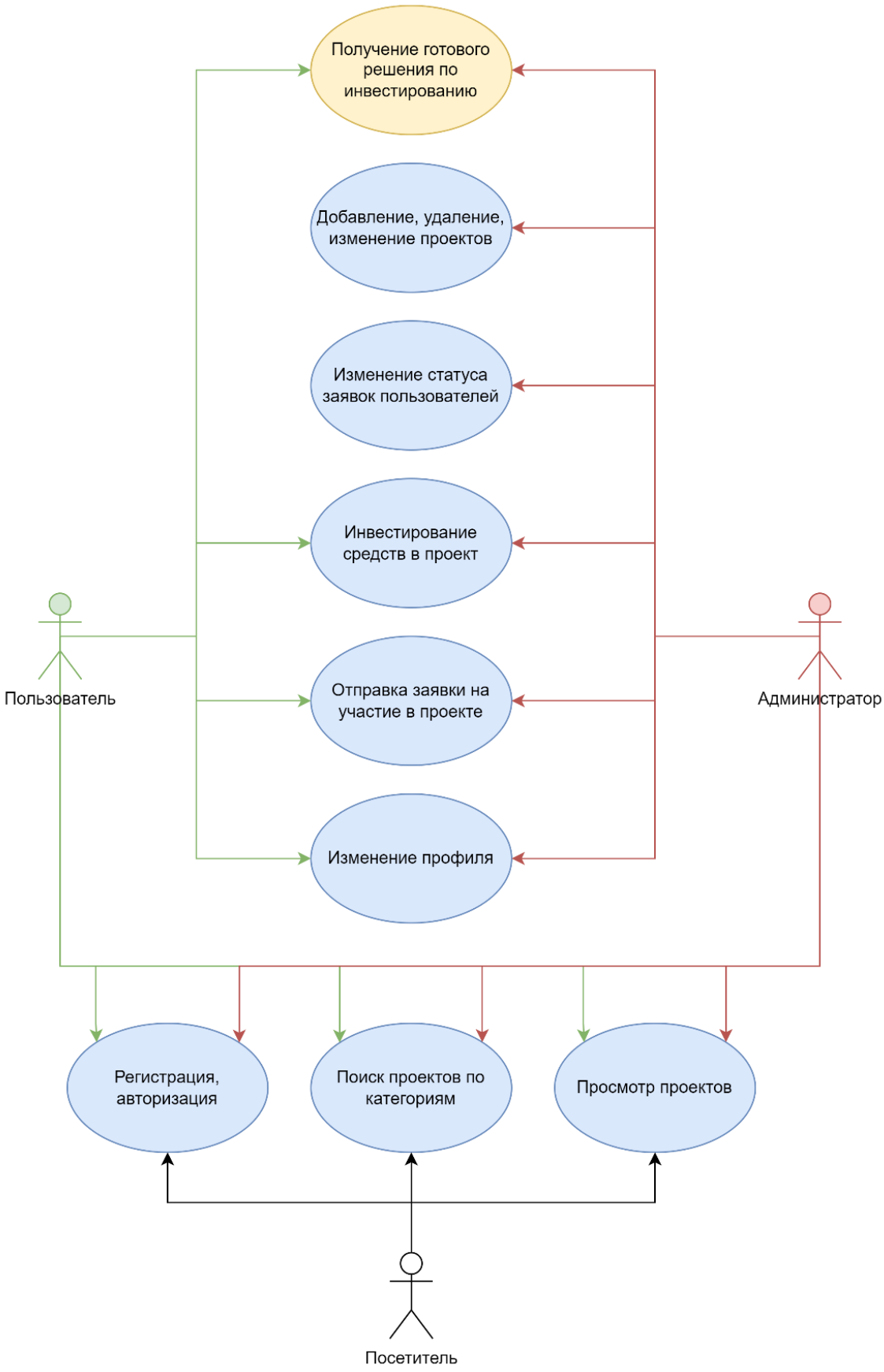

Рисунок 1.1 - UML-диаграмма вариантов использования

На диаграмме вариантов использования можно увидеть, что в системе будут присутствовать три роли: посетитель, пользователь и администратор. У посетителя доступ к основному функционалу ограничен: ему доступны только авторизация, регистрация, поиск и просмотр проектов. У обычного пользователя есть всё выше перечисленное и в дополнение - редактирование профиля, подача заявки на участие в проекте. У администратора доступно всё выше изложенное, а также добавление, удаление, изменение проектов и изменение статуса заявки пользователя. Также в проекте будет присутствовать возможность получения готового решения по инвестированию в проекты, исходя из собранных данных. Такая функция будет доступна только зарегистрированному пользователю и администратору.

Для отображения последовательности действий в информационной системе, была разработана диаграмма последовательности, представленная на рисунке 1.2.

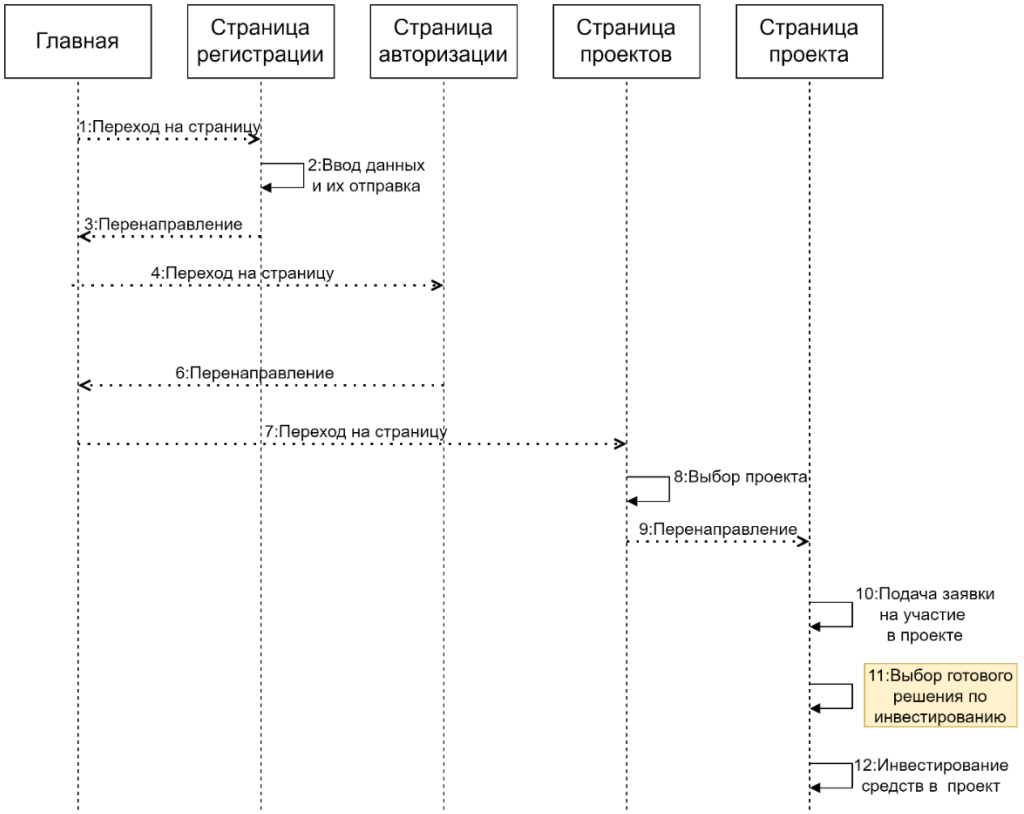

Рисунок 1.2 - UML-диаграмма последовательности

На диаграмме, представленной выше, изображён поток действий, который выполняется пользователем в результате работы с системой. Чтобы участвовать в проекте, он должен зарегистрироваться, авторизоваться по тем же данным, выбрать проект на странице «Проекты», подать заявку на странице выбранного проекта и инвестировать по своей стратегии или по предложенному готовому решению по инвестированию.

Для полного описания будущего функционала информационной системы, необходимо представить перечень функций и взаимодействий ролей системы в виде функциональной диаграммы, представленной на рисунке 1.3.

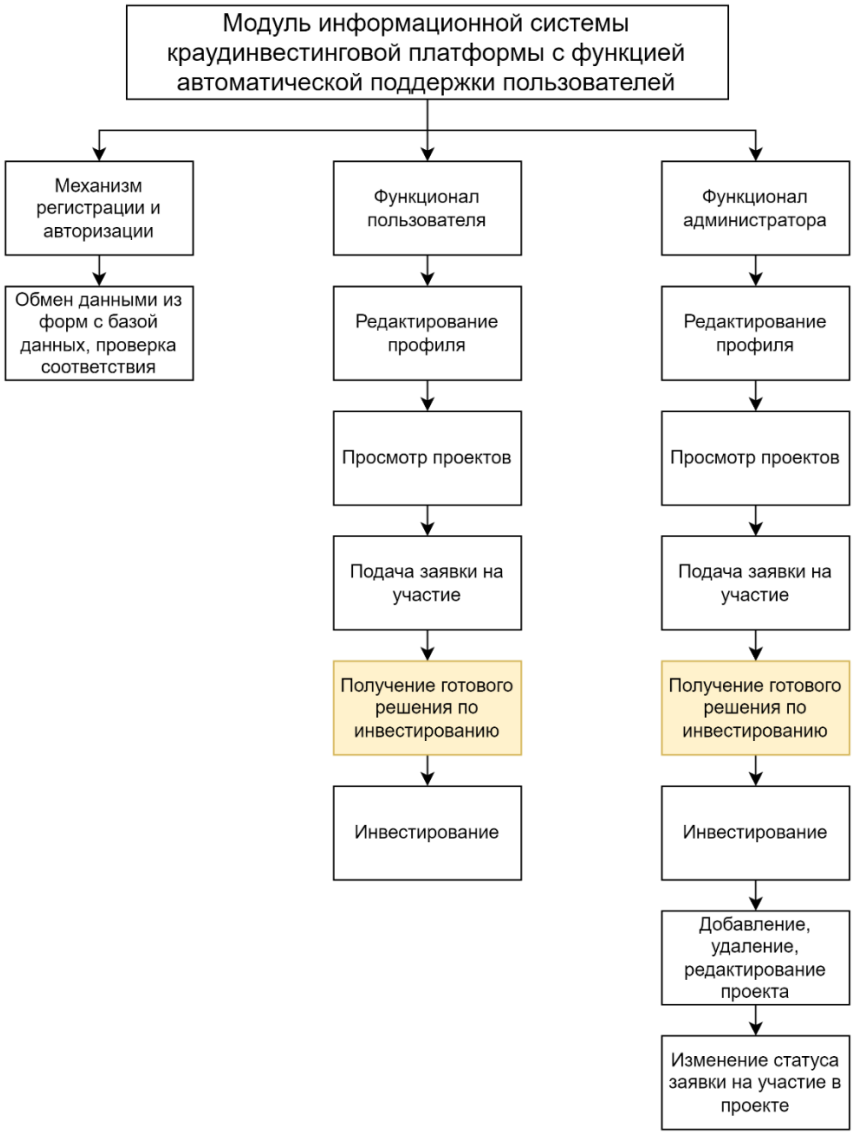

Рисунок 1.3 – Функциональная схема

Функциональная схема - схема, отображающая процессы, протекающие в отдельных функциональных цепях информационной системы [26]. Данная схема описывает: основной функционал пользователя и администратора; работу системы. Функционал администратора схож с функционалом пользователя, только имеет ещё две отдельные функции – добавление, удаление, редактирование проекта и изменение статуса заявки на участие в проекте.

Когда система взаимодействует с базой данных происходит выбор данных, изменение данных, удаление данных по запросам, поступающих из системы. Механизм авторизации проверяет соответствие введённых данных с данными из базы данных. Администратор системы может добавлять, изменять, удалять проекты. Клиент может просматривать услуги, отправлять заявки на проект и принимать в нём участие.

Подводя итоги данного раздела, можно сказать, что была исследована и проанализирована предметная область, на основе её исследования и анализа описаны следующие диаграммы: последовательности, вариантов использования и функциональная. UML-диаграмма последовательности дала понять, каким образом осуществляется порядок действий пользователя при работе с информационной системой. UML диаграмма вариантов использования отображает роли пользователей в системе и их возможности взаимодействия с ней. Функциональная UML диаграмма отобразила основные возможности ролей в системе, а также внешние взаимодействия системы с базой данных. В системе будут участвовать пользователи трёх видов: гость – посетитель приложения; сам пользователь – участник, зарегистрировавшийся в системе; администратор – имеет все права, отвечает за весь функционал в проекте. В проекте будет реализована функция автоматической поддержки пользователей, которая подразумевает под собой предоставление готового решения по инвестированию, которое поможет пользователю в выборе своей инвестиционной стратегии.

1.2 Аналитический обзор существующих решений

Сегодня на рынке криптовалют можно найти много инструментов и решений для упрощения процесса вложения средств в перспективные проекты.

Рассмотрим несколько популярных средств, которые предоставляют нам данную возможность: https://daomaker.com/, венчурные инвестиционные фонды.

Dao Maker – это платформа, предоставляющая возможность венчурных инвестиций обычным пользователям.

Что предлагает данная платформа изображено на рисунке 1.4.

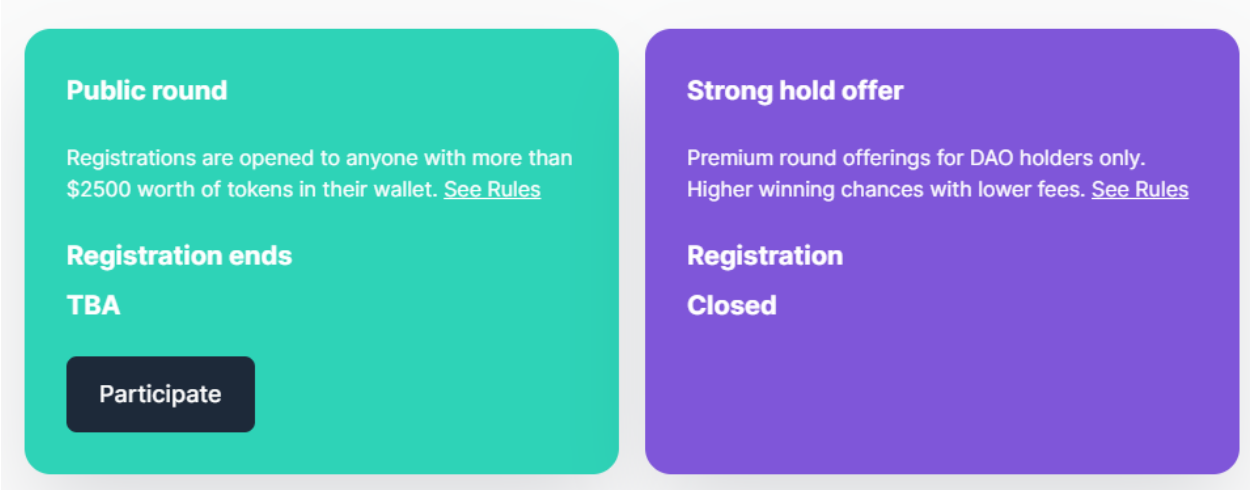

Рисунок 1.4 – Условия платформы Dao Maker

Платформа предоставляет два пути участия в инвестировании – иметь на криптовалютном кошельке средства на сумму более 2500$ или удерживать у себя на кошельке 2000 токенов DAO, что равняется примерно 2800$. А также необходимо пройти процесс верификации для получения полных возможностей платформы. Такие условия инвестирования подойдут не каждому пользователю.

Следующая площадка для рассмотрения – CoinList. Эта площадка предоставляет возможность инвестировать средства в ICO (Рисунок 1.5).

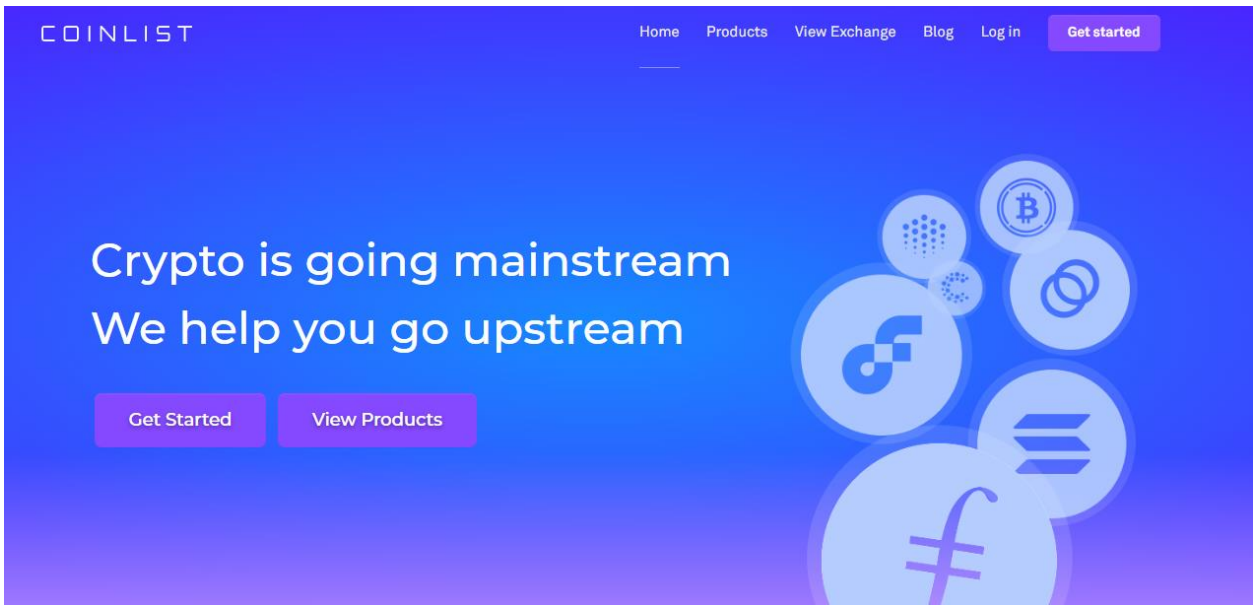

Рисунок 1.5 – Главный экран приложения CoinList

Для инвестирования через данную площадку нет необходимости в инвестировании дополнительных средств. Но они требуют пройти процесс верификации, который можете затянуться на несколько месяцев и более.

И последний проект из списка – Binance Launchpad – крупнейшая площадка для инвестирования в новые криптовалютные проекты, но также требует подтвердить свою личность по удостоверяющему документу и не предоставляет возможность инвестировать от 1$ (рисунок 1.6).

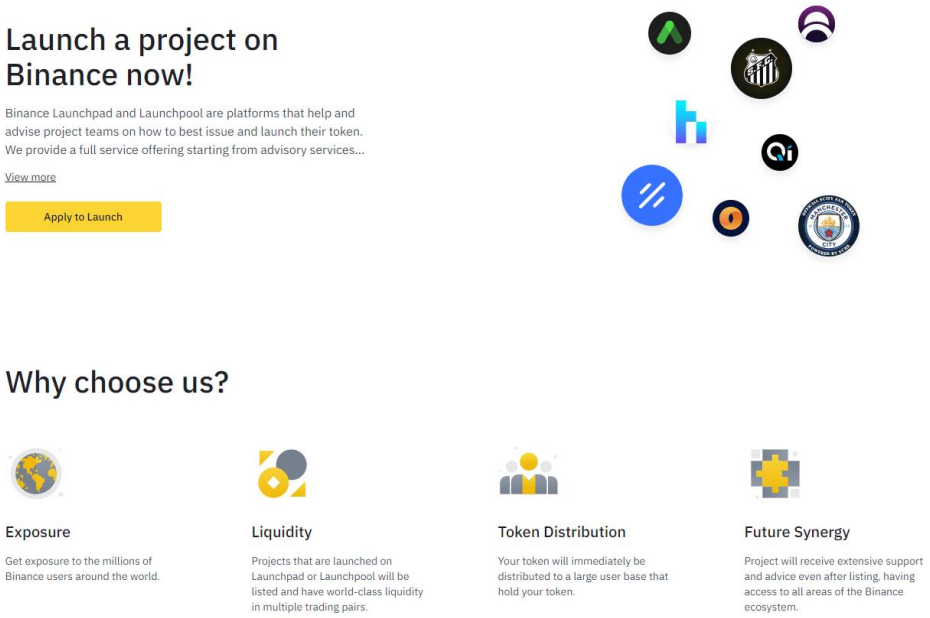

Рисунок 1.6 – Главный экран площадки Binance Launchpad

Чтобы выявить достоинства и недостатки вышерассмотренных инвестиционных решений по сравнению друг с другом, был проведен сравнительный анализ этих продуктов, результаты которого представлены в таблице 1.1.

Таблица 1.1 – Сравнительный анализ существующих решений

Сравнение	**Coinlist**	**DAOmaker**	**Binance Launchpad**
Возможность инвестирования с минимальным вложением в 1$	—	+	+
Отсутствие необходимости в прохождении верификации	—	—	—
Предоставление инвестиционных рекомендаций	—	—	—
Отсутствие необходимость во вложении дополнительных средств	+	—	—
Быстрое начало в пользовании сервисом	—	—	—

В ходе сравнительного анализа возможностей уже существующих информационных систем был сделан вывод о необходимости разработки собственной информационной системы, так как многие из них могут не удовлетворять потребностям возможной целевой аудитории.

1.3 Обоснование выбора стека технологий

1.3.1 Выбор языка (языков) программирования

Язык программирования – язык, предназначенный для написания компьютерных программ.

Рассмотрим такие языки программирования как PHP, Python, C#. Были выбраны для сравнения именно они, так как они более популяризованы.

PHP – скриптовый язык общего назначения, интенсивно применяемый для разработки веб-приложений [3]. В настоящее время поддерживается подавляющим большинством хостинг-провайдеров и является одним из лидеров среди языков, применяющихся для создания динамических веб-сайтов [2].

Преимущества PHP:

- Кроссплатформенность. PHP может быть запущен в любой операционной системе, включая юниксоиды;
- Поддержка веб-серверов. Сложно найти тот, который бы не работал с PHP;
- Разработка с помощью PHP дает много возможностей. При должном уровне владения, с помощью шаблонизатора можно создавать не только сценарии для веб-приложений, но и полноценные программы. Существуют решения, позволяющие создавать мобильные приложения на PHP.

Недостатки PHP:

- Узкопрофильность. Если вы выучили разработку с помощью PHP, то у вас одна дорога – в веб. И хотя возможности расширены различными реализациями, все же он «заточен» под программирование для Интернета;
- Безопасность. У PHP есть средства безопасности уровня системы и уровня web-приложения. Но, опять же, широкая используемость сыграла злую шутку: дыры в PHP находят

быстрее, чем разработчики успевают их закрывать. В PHP 7 множество проблем решено, но злоумышленник всегда впереди. В силу того, что массы знают «препроцессор», трудно предугадать всё [15].

Python – высокоуровневый язык программирования общего назначения с динамической типизацией и автоматическим управлением памятью, ориентированный на повышение производительности разработчика, читаемости кода и его качества, а также на обеспечение переносимости написанных на нём программ.

Преимущества Python:

- Идеален для старта в программировании;
- Простой;
- Множество доступных сред разработки;
- Универсальный;
- Востребованный на рынке.

Недостатки Python:

- Python не самый быстрый среди языков программирования. Скорость выполнения программ может быть ниже;
- Не самый удобный язык для мобильных разработок;
- Из-за гибкости типов данных потребление памяти Python не минимальное.

C# – это современный язык программирования, созданный компанией Microsoft. На нём можно писать программы любой сложности для любых платформ и операционных систем [1]. C# устроен так, чтобы программист мог писать меньше кода.

Преимущества C#:

- Поддержка подавляющего большинства продуктов Microsoft;
- Для небольших компаний и некоторых индивидуальных разработчиков бесплатными являются такие инструменты, Visual

Studio, облако Azure, Windows Server, Parallels Desktop для Mac Pro и многие другие;

- Большое количество синтаксического сахара, представляющего собой специальные конструкции, разработанные для понимания и написания кода. Они не имеют значения при компиляции.

Недостатки C#:

- Язык бесплатен только для небольших фирм, индивидуальных программистов, стартапов и учащихся;
- Приоритетная ориентированность на платформу Windows.

Исходя из того, что было рассмотрено, можно сделать вывод, что подходящим языком программирования для разработки данной системы, является PHP. Это удобный и не очень сложный язык.

1.3.2 Выбор системы(систем) управления базами данных

СУБД – специализированная программа, предназначенная для организации и ведения базы данных [29]. СУБД обеспечивает:

- Управление операциями над базой данных;
- Полное администрирование БД;
- Поддержка языков программирования для БД;
- Управление операциями над базой данных.

Примерами СУБД, которые нам подойдут, являются MS Access и MySQL [7]. Но чтобы не усложнять жизнь, было принято решение использовать Open Server.

Open Server – это портативный локальный WAMP/WNMP сервер, имеющий многофункциональную управляющую программу и большой выбор подключаемых компонентов [10].

PhpMyAdmin – веб-приложение с открытым кодом, написанное на языке PHP и представляющее собой веб-интерфейс для администрирования СУБД MySQL [9].

Была выбрана СУБД MySQL, так как она является более простым и удобным в использовании, и хорошо подходит для решения малых или средних приложений [8].

Выбрав Open Server в качестве программного решения был обеспечен не только набор полезных утилит, которые помогут в разработке, но, и локальный сервер со встроенной в него СУБД MySQL-8.0 [14].

1.3.3 Выбор дополнительных средств разработки

Фреймворк – программная платформа, определяющая структуру программной системы; программное обеспечение, облегчающее разработку и объединение разных компонентов большого программного проекта [5].

Фреймворк служит для того, чтобы облегчить труд человека и ускорят его показатели во много раз. Для разработки информационной системы был выбран фреймворк Laravel [11].

Boostrap – это открытый и бесплатный HTML, CSS и JS фреймворк, который используется веб-разработчиками для быстрой вёрстки адаптивных дизайнов сайтов и веб-приложений [12].

SDK – комплект средств разработки, который позволяет специалистам по программному обеспечению создавать приложения для определённого пакета программ, программного обеспечения базовых средств разработки и т.д.

Moralis – это SDK, предназначенный для помощи программистам создавать Web3 приложения, построенные в связке с блокчейнами [4].

JavaScript – язык программирования, использованный для разработки модулей интеграции [6].

2. СПЕЦИАЛЬНЫЙ РАЗДЕЛ

2.1 Требования к программному обеспечению

В ходе курсового проектирования необходимо разработать информационную систему краудинвестинговой платформы «Invest Inspector». К системе были сформулированы требования, которые представлены ниже. Система должна:

- Обеспечивать сбор, хранение и обработку информации согласно тематике проекта;
- Обеспечивать предоставление администратором новых проектов для вложений;
- Обеспечивать возможность поиска и просмотра проектов, подавать заявки на инвестирование в проекты;
- Предлагать пользователю готовое решение по инвестированию при помощи метода аддитивного критерия по следующим критериям: балансовый счёт пользователя, сумма всех инвестиций пользователя в проекты, собранная от пользователей сумма в проекте, сумма, собираемая для проекта, объём транзакций пользователя в блокчейне за неделю/месяц/год, количество проектов, в которых пользователь принял участие.
- Обеспечивать возможность участвовать в инвестировании всех пользователей с помощью крипто кошелька Metamask;
- Обеспечивать возможность редактировать профиль пользователя;
- Обеспечивать возможность администратору просматривать заявки пользователей и менять их статус;
- Обеспечивать сортировку заявок в системе;
- Предлагать пользователя удобный интерфейс;
- Поддерживать возможность доработки в случае расширения требований.

Вышеперечисленным требованиям отвечает информационная система, структура которой представлена на рисунке 2.1.

Рисунок 2.1 – Структура информационной системы

В данном случае предоставлена клиент-серверная архитектура системы. В представленной системе обмен информацией по Web-технологиям не отличается от информационного обмена, реализуемого по принципу «клиент-сервер», когда программа-сервер осуществляет обработку запросов, поступающих от программы-клиента.

Система представляет собой набор взаимосвязанных модулей, каждый из которых выполняет свою задачу:

- Система - главный модуль системы;
- Модуль регистрации и авторизации предназначен для регистрации и авторизации пользователей системы;
- Модуль статусов предназначен для администратора для просмотра заявок на проекты, а также изменения статуса заявки;
- Модуль редактирования профиля предоставляет пользователю доступ для редактирования своих данных в системе;

- Модуль добавления, удаления и изменения проекта предназначен для добавления проекта в систему и дальнейшего манипулирования с ним;
- Модуль подачи заявки предназначен для пользователей, которые должны подать заявку перед инвестированием в проект;
- Модуль инвестирования предназначен для инвестирования средств в проект;
- Модуль поиска предназначен для поиска проектов по категориям;
- Модуль готового решения по инвестированию предназначен для рекомендации пользователю суммы для инвестирования в проект;
- Модуль подключения крипто кошелька предназначен для подключения пользователя к системе с помощью крипто кошелька Metamask.

2.2 Проектирование и разработка структуры базы данных

В первую очередь был определён набор сущностей в базе данных. Диаграммы ER-типа, определённых сущностями таблиц, связями, атрибутами и ключами сущностей, представлены на рисунке 2.2 [17].

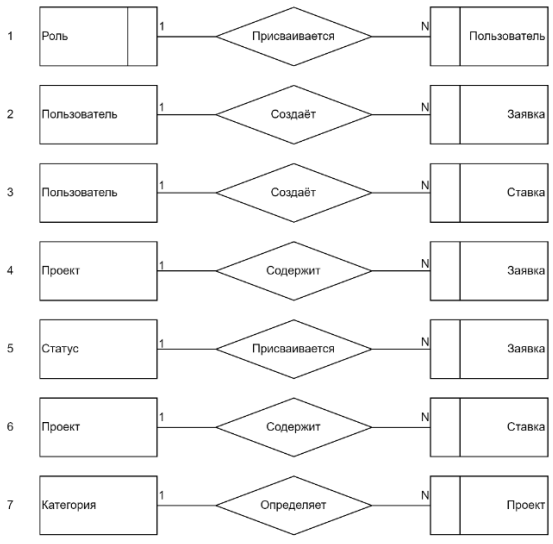

Рисунок 2.2 – Диаграммы ER-типа

База данных программы работает под управлением СУБД MySQL. Схема БД представлена на рисунке 2.3.

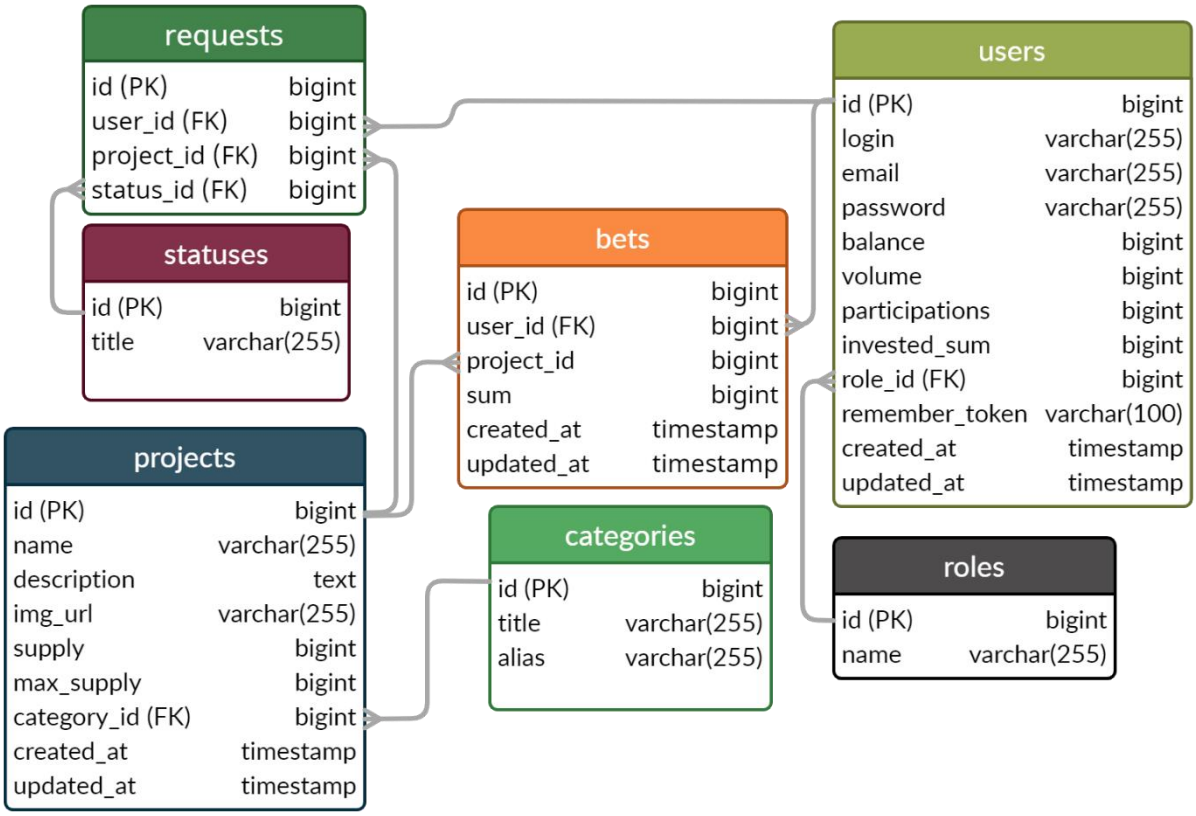

Рисунок 2.3 – Схема базы данных

В базе данных содержится совокупность таблиц. В каждой таблице реализована определенная сущность:

- Users - список пользователей;
- Roles - список названий ролей;
- Bets - список вложений пользователей системы в проекты;
- Projects - список проектов;
- Requests - список заявок;
- Categories - список категорий;
- Statuses - список статусов заявок.

Таблица 2.1 - Структура таблицы Users

Наименование поля	**Тип**	**По умолчанию**	**Ключ**	**Описание**
Id	Bigint		PK	Идентификатор записи
Login	Varchar(255)			Логин пользователя
Email	Varchar(255)			Электронная почта пользователя
Password	Varchar(255)			Пароль пользователя
Balance	Bigint	0		Баланс пользователя
Volume	Bigint	0		Объём транзакций на блокчейне
Participations	Bigint	0		Число проектов, в которые инвестировал пользователь
Invested_sum	Bigint	0		Сумму инвестированных активов
Role_id	Bigint		FK	Идентификатор роли

Таблица 2.2 - Структура таблицы Roles

Наименование поля	**Тип**	**Разрешить пустое**	**Ключ**	**Описание**
Id	Bigint		PK	Идентификатор записи
Name	Varchar(255)			Наименование роли

Таблица 2.3 - Структура таблицы Statuses

Наименование поля	**Тип**	**Разрешить пустое**	**Ключ**	**Описание**
Id	Bigint		PK	Идентификатор записи
title	Varchar(255)			Наименование статуса

Таблица 2.4 - Структура таблицы Requests

Наименование поля	**Тип**	**Разрешить пустое**	**Ключ**	**Описание**
Id	Bigint		PK	Идентификатор записи
User_id	Bigint		FK	Идентификатор пользователя
Project_id	Bigint		FK	Идентификатор проекта
Status_id	Bigint		FK	Идентификатор статуса

Таблица 2.5 - Структура таблицы Projects

Наименование поля	**Тип**	**По умолчанию**	**Ключ**	**Описание**
Id	Bigint		PK	Идентификатор записи
name	Varchar(255)			Наименование проекта
Description	Text			Описание проекта
Img_url	Varchar(255)			Ссылка на директорию изображения
Supply	Bigint	0		Количество собранных средств
Max_supply	Bigint			Максимально доступное для
Category_id	Bigint		FK	Идентификатор категории

Таблица 2.6 - Структура таблицы Categories

Наименование поля	**Тип**	**Разрешить пустое**	**Ключ**	**Описание**
Id	Bigint		PK	Идентификатор записи
title	Varchar(255)			Наименование категории
alias	Varchar(255)			Альтернативное наименование категории

Таблица 2.7 - Структура таблицы Bets

Наименование поля	**Тип**	**Разрешить пустое**	**Ключ**	**Описание**
Id	Bigint		PK	Идентификатор записи
User_id	Bigint		FK	Идентификатор пользователя
Project_id	Bigint		FK	Идентификатор проекта
Sum	Bigint			Вложенная сумма

Поле «Ключ» в таблице показывает является ли данное поле первичным или внешним ключом, или не является. Например, в таблице Requests все поля, за исключение первичного ключа, являются внешними ключами.

Таблица Users содержит основную информацию о пользователях системы. Также имеется внешний идентификатор ролей.

Таблица Statuses хранит информацию о наименованиях статусов заявок.

Таблица Roles хранит информацию о наименованиях ролей пользователей. Она отвечает за права, предоставляемые пользователю в системе.

Таблица Requests содержит в себе все заявки пользователей на участие в проектах. Именно одобренная заявка даёт пользователю право вкладываться в проект.

Таблица Projects содержит всю основную информацию о всех проектах системы.

Таблица Categories содержит информацию о наименованиях категорий проектов. По категориям осуществляется поиск проектов.

Таблица Bets содержит информацию о всех вкладах пользователей в проекты.

2.4. Проектирование и разработка пользовательского интерфейса

Интерфейс - совокупность технических, программных и методических (протоколов, правил, соглашений) средств сопряжения в вычислительной системе пользователей с устройствами и программами, а также устройств с другими устройствами и программами [16].

Выбор структуры диалога, на котором будет построено приложение – первый из этапов разработки интерфейса приложения. Данное приложение состоит диалога на основе экранных форм и меню.

После выбора структуры диалога приложения, необходимо разработать макеты интерфейса приложения.

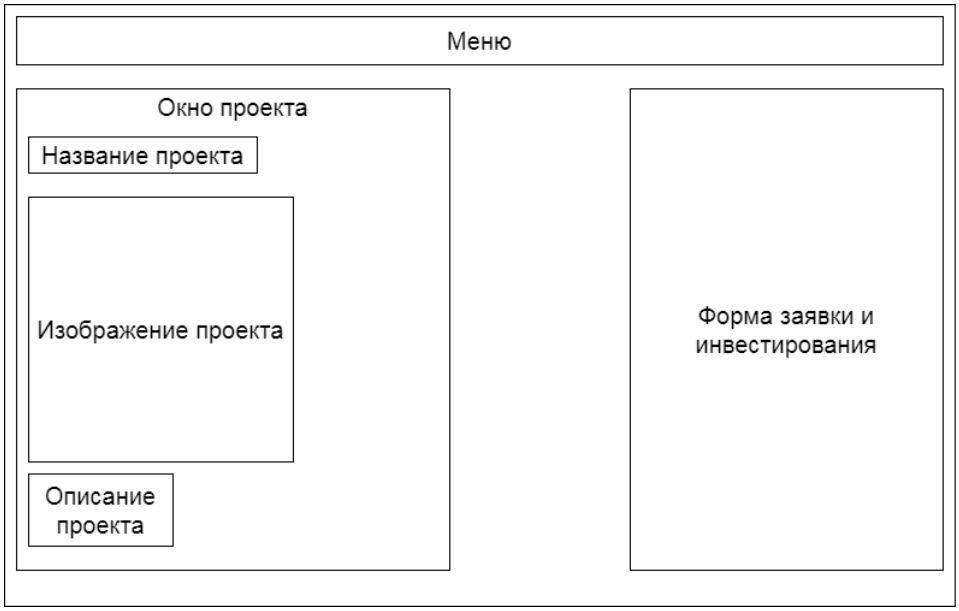

Рисунок 2.7 – Макет страницы просмотра проекта с формой заявки и инвестирования

На рисунке 2.7 представлен макет страницы просмотра проекта с формой заявки и инвестирования, а также расположение основного меню, с которым пользователь взаимодействует для перехода по основным экранам системы.

На рисунке 2.8 представлены макеты формы подачи заявки, статуса поданной заявки, формы выбора готового решения по инвестированию и формы инвестирования средств в проект.

Рисунок 2.8 – Макет формы подачи заявки, статуса поданной заявки, формы выбора готового решения по инвестированию и формы инвестирования средств в проект

В форме выбора готового решения по инвестированию представлена функция автоматизации – автоматическая поддержка пользователя. Данная функция работает по принципу расчёта рейтинга пользователя по аддитивному критерию

На рисунке 2.9 представлен макет меню приложения. Меню находится в шапке приложения. В меню входят основные ссылки и кнопки перехода по страницам проекта.

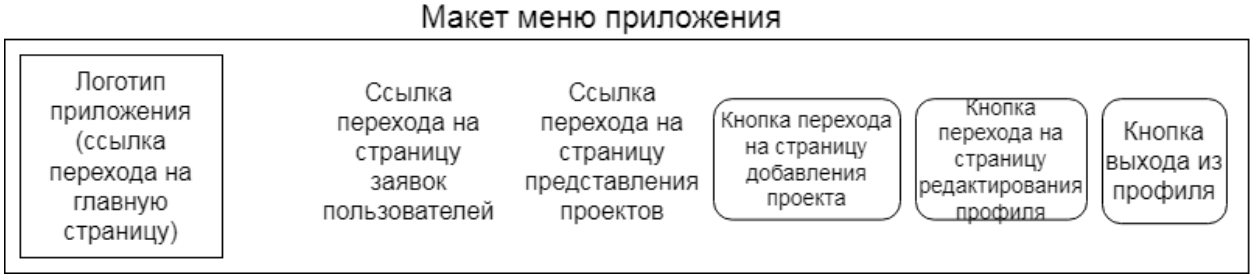

Рисунок 2.9 – Макет меню приложения

В таблице 2.8 представлены виды возможных ошибок, которые может допустить пользователь при работе с приложением.

Таблица 2.8 – Виды ошибок и ответов системы на них

Номер сообщения	Ошибка пользователя	Ответы системы
1	Неправильный ввод email’а или пароля пользователя	Ошибка оповещает о неправленом вводе email и пароля
2	Превышение возможной суммы инвестирования пользователем	Ошибка оповещает о превышенной сумме возможного инвестирования в проект

В данном разделе были созданы макеты меню приложения, формы подачи заявки, статуса поданной заявки, формы выбора готового решения по инвестированию и формы инвестирования средств в проект. По данным макетам разработан пользовательский интерфейс приложения.

2.4 Разработка алгоритмов

Для более наглядного представления работы информационной системы было принято решение построить блок-схемы алгоритмов работы самой системы и некоторых её модулей отдельно.

Блок-схема — это диаграмма, на которой обычно представлен процесс, система или компьютерный алгоритм и которая используется для

документирования, планирования, уточнения или визуализации многоэтапного рабочего процесса [25].

Ниже представлен общий алгоритм работы модулей приложения на рисунке 2.4. Описание работы каждого модуля присутствует в разделе 2.1.

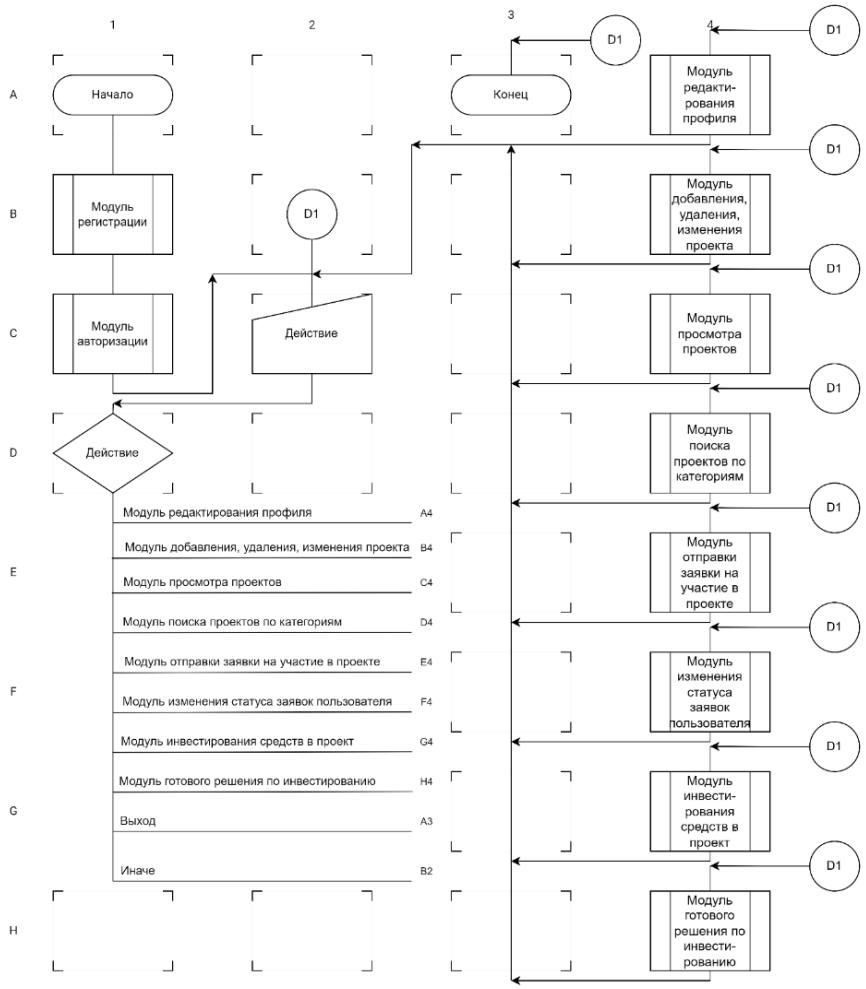

Рисунок 2.4 – Общий алгоритм работы модулей приложения

Также разработан алгоритм, отвечающий за работу функции автоматизации. На рисунке 2.5 и 2.6 представлена пошаговая работа модуля готового решения по инвестированию и расчёт аддитивного критерия.

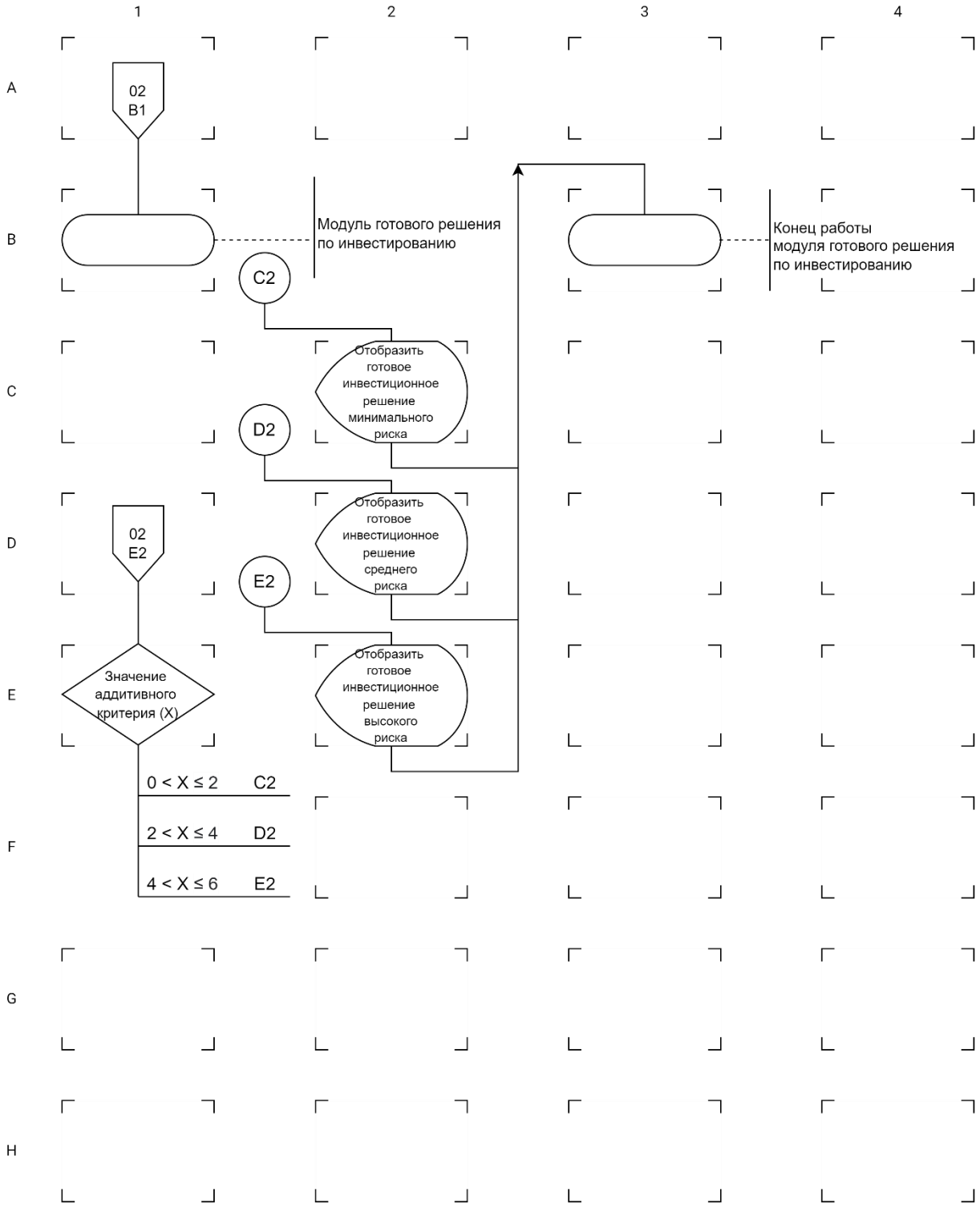

Рисунок 2.5 – Алгоритм работы модуля готового решения по инвестированию

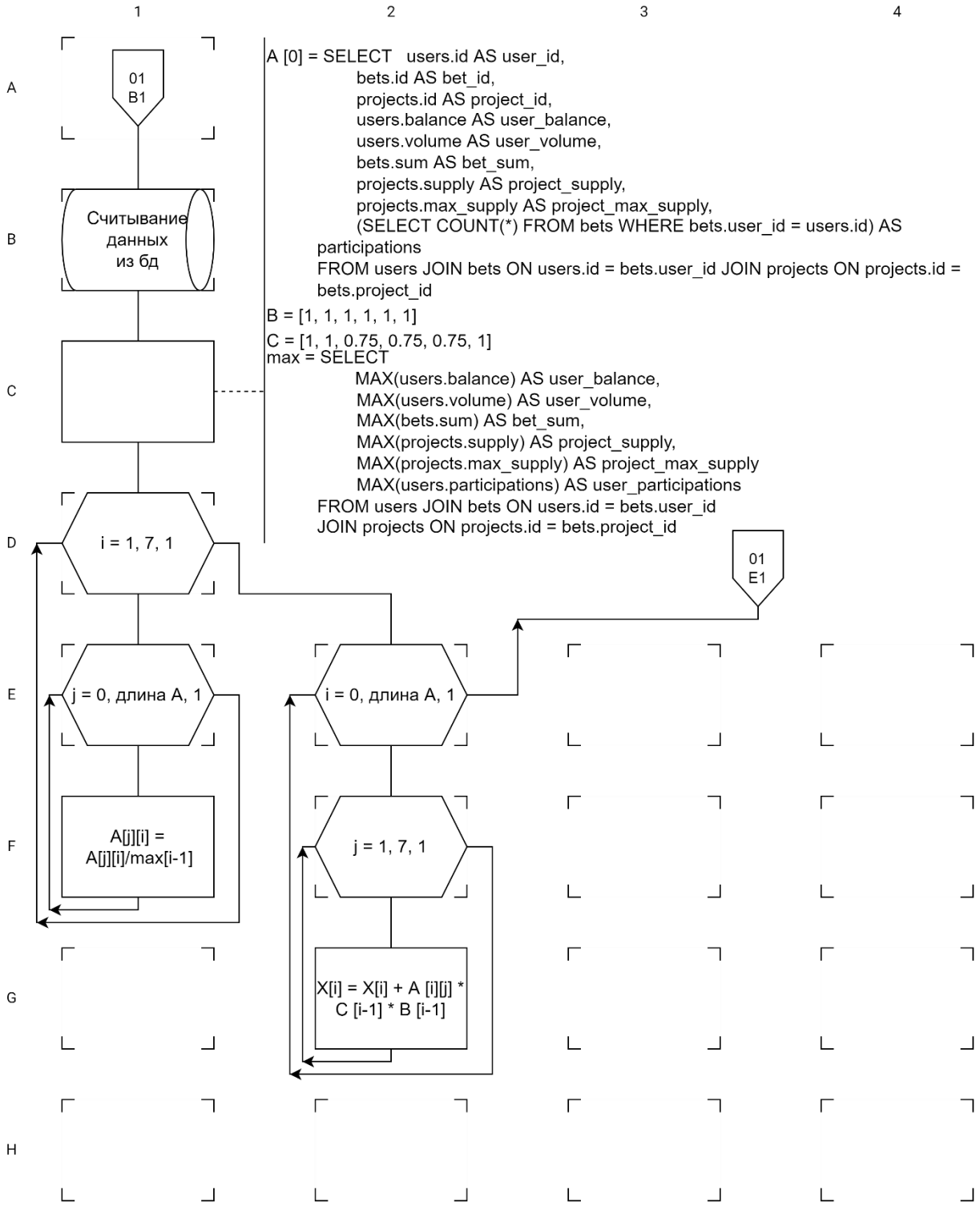

Рисунок 2.6 – Расчёт аддитивного критерия

В алгоритмах показана пошаговая работа модуля готового решения по инвестированию, а также расчёт аддитивного критерия. Система берёт данные пользователя из базы данных. Рассчитывается аддитивный критерий пользователя из имеющихся данных и, при выполнении одного из условий,

отображается соответствующее готовое решение по инвестированию, которым можно будет воспользоваться.

В результате данного раздела был разработан общий алгоритм системы и алгоритм функции автоматизации, каждый из которых в дальнейшем поможет предотвратить совершение ошибок при разработке приложения и избежать сбои при работе с модулем готового решения по инвестированию.

2.5 Разработка бизнес-логики

Все модули системы представляют собой множество элементов графического интерфейса, публичных методов классов-контроллеров, визуализирующих эти элементы и обрабатывающих запросы со стороны пользователя, а также статических методов классов-моделей, содержащие соответствующие таблицы из базы данных, которые используются в классах-котроллерах и в которых описана основная логика данного модуля.

При разработке информационной системы были использованы следующие классы [13].

Классы-контроллеры:

- UserController - содержит методы, отображающие список пользователей в соответствующем шаблоне, методы обрабатывающие запросы пользователя на регистрацию, авторизацию, удаления пользователей;
- PageController - отвечает за отображение всех логически связанных страниц приложения;
- Projectcontroller - содержит методы, обрабатывающие добавление, удаление и изменение лотов;
- RequestController - содержит методы, отвечающие за изменение статуса заявок.

Классы-модели:

- Category - воспроизводит таблицу Categories, используется для связи с таблицей Projects;

- Project - воспроизводит таблицу Projects, объекты класса Projects;
- Req - содержит статические методы добавления, назначения, изменения статуса заявки;
- Role - воспроизводит таблицу Roles;
- Status - воспроизводит таблицу Statuses;
- User - воспроизводит таблицу Users, объекты класса Users используются для создания новых записей, изменения их в таблице users. Также модель используется для связи с таблицами Roles, Requests.

Более подробно изучим работу некоторых из классов в рамках разработанной информационной системы.

В первую очередь рассмотрим методы класса UserController. Методы представленного класса отвечают за работу с данными пользователя [28].

```
public function signup(Request $request){
    $userData = $request->all();
    $validator = Validator::make($userData,[
        'email' => 'required|unique:users|email:rfc,filter',
        'login' => 'required|unique:users',
        'password' => 'required'
    ]);
    if($validator->fails()) {
        return redirect(route('signup-page'))
            ->withErrors($validator)
            ->withInput();
    }
    $user = new User();
    $user->login = $userData['login'];
    $user->email = $userData['email'];
    $user->password = bcrypt($userData['password']);
    $user->role_id = 2;
    $user->balance = 0;
    $user->save();
    return redirect(route('main-page'));
}
```

Рисунок 2.10 - Метод signup

Метод signup принимает введённые пользователем данные и валидирует их. Если валидация прошла успешно, то данные заносятся в базу данных, а пользователя перенаправляет на главную страницу [18].

Метод login схож с signup, только вместо добавления новой записи в БД метод проверяет уже существующих пользователей на соответствие с введёнными данными.

Также имеется метод выхода из аккаунта logout.

```php
public function logout() {
    Auth::logout();
    return redirect('/');
}
```

Рисунок 2.11 - Метод logout

```php
public function login(Request $request)
{
    $userInfo=$request->only('email','password');
    $validator= Validator::make($userInfo, [
        'email'=>'required|email:rfc,filter',
        'password'=>'required'
    ]);
    if ($validator->fails()) {
        return redirect(route('login-page'))
            ->withErrors($validator)
            ->withInput();
    }
    if (Auth::attempt($userInfo)) {
        return redirect('/');
    }
    return redirect(route('login-page'))
        ->withErrors(['auth_error' => 'Email или пароль введены некорректно'])
        ->withInput();
}
```

Рисунок 2.12 - Метод login

Метод profile позволяет пользователю в случае необходимости изменить данные своего аккаунта. Метод принимает данные и, если валидация проходит успешно, меняет старые данные на новые.

```php
public function profile(Request $request)
{
    $userData = $request->all();
        $validator = Validator::make($userData, [
            'email' => Rule::unique('users')
            ->where(function ($query) {
                return $query->where('email');
            }),
            'login' => 'required',
            'password' => 'required'
        ]);
    if ($validator->fails()) {
        return redirect(route('profile-page'))
            ->withErrors($validator)
            ->withInput();
    }
    $user = Auth::user();
    $user->email = $userData['email'];
    $user->login = $userData['login'];
    $user->password = bcrypt($userData['password']);
    $user->save();
    return redirect(route('profile-page'));
}
```

Рисунок 2.13 - Метод profile

Класс ProjectController описывает методы, отвечающие за работу с данными проекта информационной системы.

Метод project_add работает по аналогии с регистрацией пользователя - принимает данные, валидирует их, добавляет новый проект в базу данных, перенаправляет пользователя на страницу с проектами.

```php
public function project_add(Request $request){
    $ProjectData = $request->all();
    $validator = Validator::make($ProjectData,[
        'name' => 'required|unique:projects',
        'description' => 'required|unique:projects',
        'category_id' => 'required',
        'img_url' => 'required|unique:projects|mimes:jpeg,png',
    ]);
    if($validator->fails()) {
        return redirect(route('project-add-page'))
            ->withErrors($validator)
            ->withInput();
    }
    $project = new Project();
    $project->name = $ProjectData['name'];
    $project->description = $ProjectData['description'];
    $project->category_id = $ProjectData['category_id'];
    if ($request->hasFile('img_url')) {
        $project->img_url = $ProjectData['img_url']->store('/img/projects');
    }
    $project->save();
    return redirect(route('projects'));
}
```

Рисунок 2.14 - Метод project_add

Метод editProject отвечает за редактирование данных о проекте, аналогичен методу редактирования профиля.

```php
public function editProject(Request $request){
    $editData = $request->all();
    $validator = Validator::make($editData,[
        'img_url' => '|mimes:jpeg,png',
    ]);
    if($validator->fails()) {
        return redirect()->back()
            ->withErrors($validator)
            ->withInput();
    }
    $project = Project::findOrFail($editData['project_id']);
    $project->name = $editData['name'];
    $project->description = $editData['description'];
    $project->category_id = $editData['category_id'];
    if ($request->hasFile('img_url')) {
        $project->img_url = $editData['img_url']->store('/img/projects');
    }
    $project->save();
    return redirect(route('projects'));
}
```

Рисунок 2.15 - Метод editProject

Метод delProject отвечает за удаление проекта из базы данных.

```php
public function delProject ($id)
{
    $project = Project::findOrFail($id);
    if($project->supply>0){
        return redirect()->back();
    }
    DB::table('projects')->where('id', '=', $id)->delete();
    return redirect(route('projects'));
}
```

Рисунок 2.16 - Метод delProject

Метод search предоставляет возможность поиска нужного проекта в информационной системе. Метод принимает полученные данные, сравнивает с записями проектов в таблице и выдаёт подходящие запросу варианты.

```
public function search(Request $request){
    $search = $request->input('search');
    $project = Project::query()
        ->where('name', 'LIKE', "%{$search}%")
        ->orWhere('description', 'LIKE', "%{$search}%")
        ->get();
    $requests = Req::orderBy('id', 'desc')->get();
    return view('search', ['projects'=>$project,
                           'requests' => $requests,
                           'search' => $search]);
}
```

Рисунок 2.17 - Метод search

Метод transfer отвечает за занесение транзакций в базу данных. Метод получает данные и заносит информацию в базу данных.

```
    public function transfer(Request $request) {
        $data = json_decode($request->getContent(), true);
        $bet= new Bet();
        $bet->sum = $data['amount'];
        $bet->user_id = $data['id'];
        $bet->project_id = $data['project_id'];
        $bet->created_at = Carbon::now();
        $bet->updated_at = Carbon::now();
        $bet->save();

    }
}
```

Рисунок 2.18 - Метод transfer

Класс PageController отвечает за отображение страниц информационной системы [20]. Большая часть из методов однотипны - обычное отображение шаблона. Но есть те, которые дополнены некоторым функционалом:

- projects - сортирует все проекты по id;
- editProject - находит проект по id;
- requests - сортирует все запросы по id;
- search - сортирует результаты поиска по id;
- bets - сортирует ставки по id.

```php
class PageController extends Controller
{
    public function index(){
        return view('index');
    }
    public function signup(){
        return view('sign-up');
    }
    public function login(){
        return view('login');
    }
    public function profile(){
        return view('profile');
    }
    public function projects(){
        $projects = Project::orderBy('id', 'desc')->get();
        return view('projects', ['projects' => $projects]);
    }
    public function project_add(){
        $categories = Category::all();
        return view('project-add', ['categories' => $categories]);
    }

    public function editProject($id){
        $project = Project::findOrFail($id);
        return view('project-edit', ['project' => $project]);
    }

    public function requests(){
        $requests = Req::orderBy('id', 'desc')->get();
        return view('requests', ['requests' => $requests]);
    }
```

Рисунок 2.19 - Класс PageController

Также в классе PageController присутствует функция project_single, отвечающая за отображение страницы проекта с формой инвестирования в проект и инвестиционной рекомендацией. Функция изображения на рисунках 2.20 и 2.21 [21].

```php
public function project_single($id){
    $project = Project::findOrFail($id);
    $requests = Auth::user()->requests->where('project_id', $id)->all();
    $statuses = Status::orderBy('id', 'desc')->get();
    $data = DB::table("users")
        ->join("requests", function($join){
            $join->on("users.id", "=", "requests.user_id");
        })
        ->join("projects", function($join){
            $join->on("projects.id", "=", "requests.project_id");
        })
        ->select("users.balance", "users.volume", "users.participations",
            "users.invested_sum", "projects.supply", "projects.max_supply")
        ->where('users.id' ,'=', Auth::user()->id)
        ->get();
    $balance = DB::table("users")->max('users.balance');
    $volume = DB::table('users')->max('users.volume');
    $participations = DB::table('users')->max('users.participations');
    $invested_sum = DB::table('users')->max('users.invested_sum');
    $supply = DB::table('projects')->max('projects.supply');
    $max_supply = DB::table('projects')->max('projects.max_supply');
    $max = ['balance' =>$balance,
            'volume' => $volume,
            'participations' => $participations,
            'invested_sum' => $invested_sum,
            'supply' => $supply,
            'max_supply' =>$max_supply];
```

Рисунок 2.20 – Функция project_single

```php
    $b = [1, 1, 1, 1, 1, 1];
    $c = [1, 1, 0.75, 0.75, 0.75, 1];
    $matrix = [];
    $array = json_decode(json_encode($data), true);
    $k=0;
    foreach ($array as $item)
    {
        $matrix[$k] = [
            $item['balance']/$max['balance'],
            $item['volume']/$max['volume'],
            $item['participations']/$max['participations'],
            $item['invested_sum']/$max['invested_sum'],
            $item['supply']/$max['supply'],
            $item['max_supply']/$max['max_supply']
        ];
        $k++;
    }
    $rate = 0;
    for ($i=0; $i < count($item);$i++){
        $rate += $matrix[0][$i] * $c[$i] * $b[$i];
    }
    return view('project-single', ['project' => $project,
                                   'requests' => $requests,
                                   'statuses' => $statuses,
                                   'rate' => $rate]);
}
```

Рисунок 2.21 – Продолжение функции project_single

Класс RequestController отвечает за заявки пользователей - создание самой заявки и изменение её статуса администратором [24]. Здесь присутствует четыре метода, которые выполняют следующие функции:

- addReq - отвечает за создание заявки пользователем;
- reqWait – присваивает заявке статус «Запрошено»;
- reqSuccess - меняет статус заявки на «Одобрено»;
- reqDenied – присваивает заявке статус «Отклонено».

```
class RequestController extends Controller
{
    public function addReq(Request $request){
        $ReqData = $request->all();
        $req = new Req();
        $req->user_id = $ReqData['user_id'];
        $req->project_id = $ReqData['project_id'];
        $req->status_id = 1;
        $req->save();
        return redirect()->back();
    }
    public function reqWait($id){
        $reqw = Req::findOrFail($id);
        $reqw->status_id = 1;
        $reqw->save();
        return redirect()->back();
    }
    public function reqSuccess($id){
        $reqs = Req::findOrFail($id);
        $reqs->status_id = 2;
        $reqs->save();
        return redirect()->back();
    }
    public function reqDenied($id){
        $reqs = Req::findOrFail($id);
        $reqs->status_id = 3;
        $reqs->save();
        return redirect()->back();
    }
}
```

Рисунок 2.22 – Класс RequestController

Далее представлен код интеграции подключения криптовалютного кошелька Metamask в систему. Данное подключение реализовано с помощью активации сервера Moralis и функции login. Функция представлена на рисунке 2.23 [22].

```
/* Moralis init code */
const serverUrl = "https://b2l05rdlwnhd.usemoralis.com:2053/server";
const appId = "6cKTnr3cJLcmYEER9ywhAAti5Z9NmY6FU01Vcf0L";
Moralis.start({ serverUrl, appId });
/* Authentication code */
async function login() {
  await Moralis.authenticate();
}
```

Рисунок 2.23 – Функция login

Реализация функции инвестирования средств в проект с помощью отправки криптовалюты кошельком Metamask представлена на рисунке 2.24 [16]. Данной функции необходимы данные о кошельке, на который пользователь отправляет свои средства, а также информация о типе активов, которые отправляет пользователь.

```
async function transferErc20(){
    const address = document.getElementById('erc20-address').value;
    const amount = document.getElementById('erc20-amount').value;
    const contract = document.getElementById('erc20-contract').value;
    const decimals = document.getElementById('erc20-decimals').value;

    const options = {
        type: "erc20",
        amount: Moralis.Units.Token(amount, decimals),
        receiver: address,
        contractAddress: contract,
    };
    let result = await Moralis.transfer(options);
    sendAPI(address, amount, contract, decimals);
}
document.getElementById('transfer-erc20').onclick = transferErc20;
document.getElementById("btn-login").onclick = login;
document.getElementById("btn-logout").onclick = logOut;
```

Рисунок 2.24 – Функция transferErc20

Функция sendAPI, представленная на рисунке 2.25, отвечает за сохранение информации о транзакции пользователя в базу данных.

```
async function sendAPI(address, amount, contract, decimals) {
    const response = await fetch('http://kursachfin-main-main/api/test', {
        method: 'POST', // *GET, POST, PUT, DELETE, etc.
        headers: {
            'Content-Type': 'application/json'
            // 'Content-Type': 'application/x-www-form-urlencoded',
          },
        body: JSON.stringify({'address': address,
                              'amount': amount,
                              'contract': contract,
                              'decimals':decimals})
         // body data type must match "Content-Type" header
      });
    let ans = await response.json();
    console.log(ans);
}
```

Рисунок 2.25 – Функция sendAPI

3. РЕАЛИЗАЦИЯ

3.1. Настройка технологического обеспечения

В данном курсовом проекте разработана информационная система, позволяющая пользователям просматривать проекты и участвовать в них. Система использует базу данных MySQL. Перед первым запуском программы необходимо подключить файлы существующей базы данных, а также настроить подключение к ней.

Опишем последовательность действий.

В файле .env задаются настройки конфигурации для установки соединения с БД. По умолчанию данные настройки уже имеют заранее заданные при разработке проекта значения, которые в дальнейшем необходимо использовать при конфигурации нового сервера MySQL. В случае необходимости их можно изменить на собственные. За установку соединения отвечают следующие константы:

DB_HOST=127.0.0.1

DB_PORT=3306

DB_DATABASE=invest

DB_USERNAME=root

DB_PASSWORD=root

Подключение базы данных:

1. Установить OpenServer версии 5.3.7;
2. Открыть OpenServer и зайти в настройки;
3. Перейти во вкладку Сервер, и настроить IP-адрес сервера - 127.0.0.1, если он не настроен, и порт MySQL - 3306;
4. Перейти во вкладку Модули и настроить:
   - HTTP - Apache_2.4PHP_7.2-7.4;
   - PHP - PHP_7.4;
   - MySQL 8.0.
5. Перезапустить OpenServer;

6. Нажать на флажок, выбрать дополнительно, и в самом верху PhpMyAdmin;
7. Ввести USERNAME и PASSWORD - «root» и создать базу данных, которая называется invest.

3.2 Инструкция использования

Для получения полного доступа к функционалу пользователю необходимо нажать на кнопку «Sign up» в верхнем правом углу на главной странице приложения (рисунок 3.1).

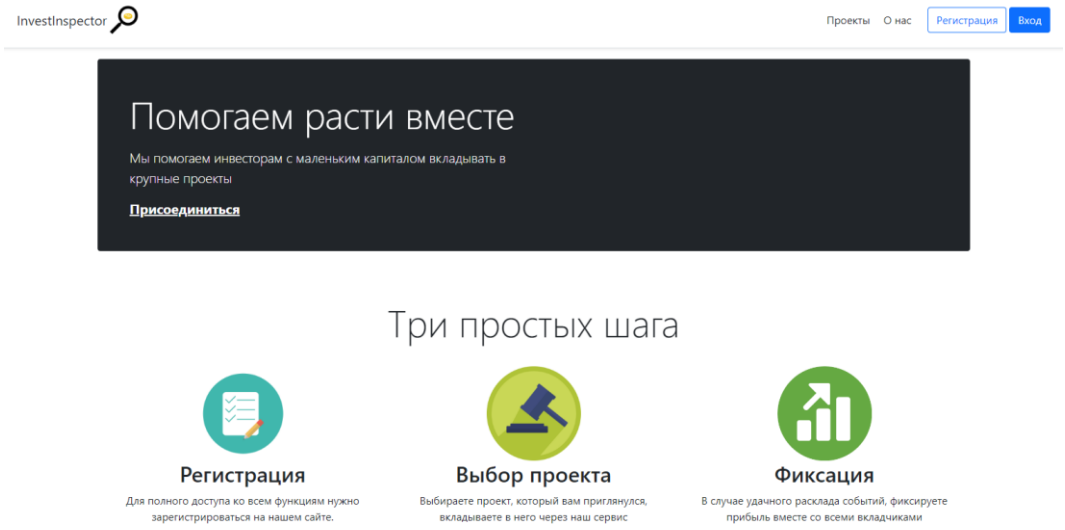

Рисунок 3.1 - Главная страница системы

После перехода на страницу создания аккаунта пользователь вводит почту, логин, пароль в форме регистрации и отправляет данные соответствующей кнопкой (рисунок 3.2).

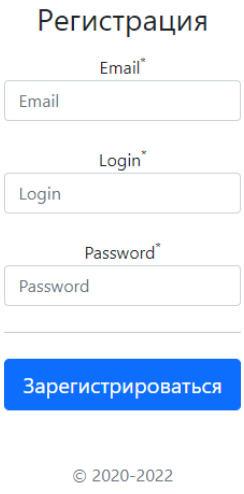

Рисунок 3.2 - Форма регистрации

После успешного создания аккаунта произойдёт перенаправление пользователя на главную страницу, откуда он может быть перенаправлен на страницу входа в аккаунт, где необходимо ввести почту и пароль, введённые при регистрации (рисунок 3.3).

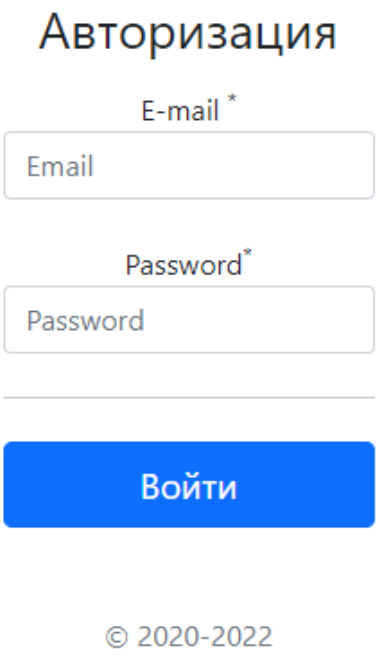

Рисунок 3.3 - Форма логина

Теперь пользователь в системе, и он может работать с ней в полной мере (рисунок 3.4).

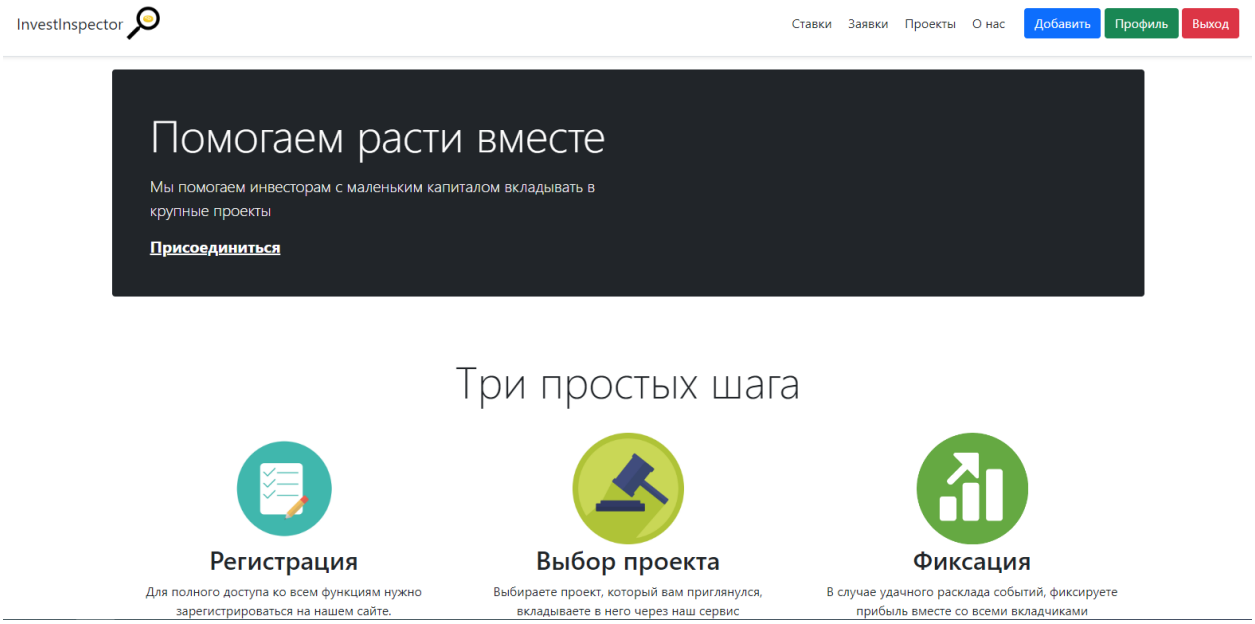

Рисунок 3.4 - Пользователь в системе

Первое, что доступно пользователю - редактирование профиля. Перейдя по ссылке «Профиль», он может изменить почту, логин и пароль, а также увидеть свой баланс (рисунок 3.5).

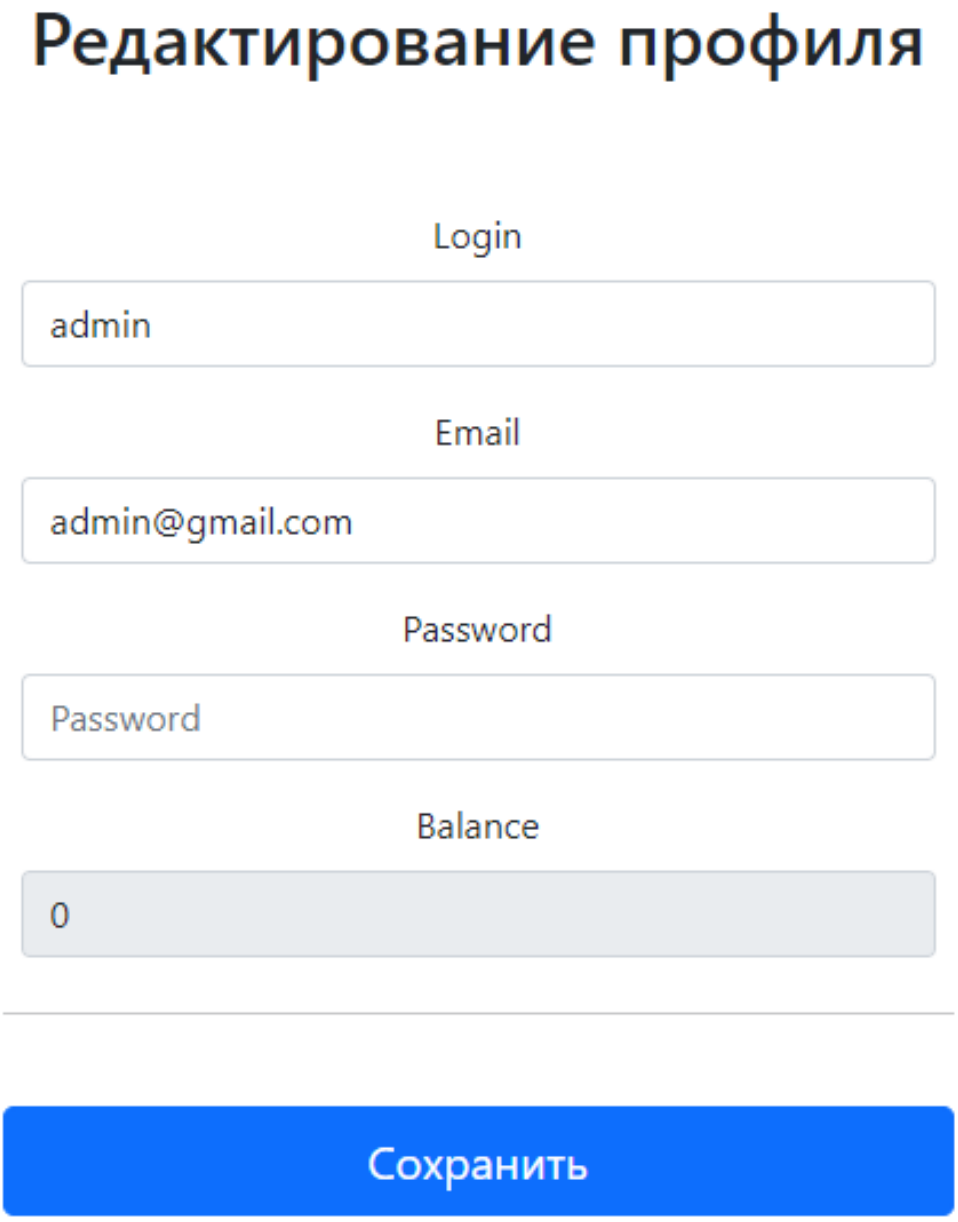

Рисунок 3.5 - Редактирование профиля

Вторая возможность пользователя - просмотр проектов и подача заявки на участие. Подача запроса на участие осуществляется через кнопку «Подать заявку» (рисунок 3.6).

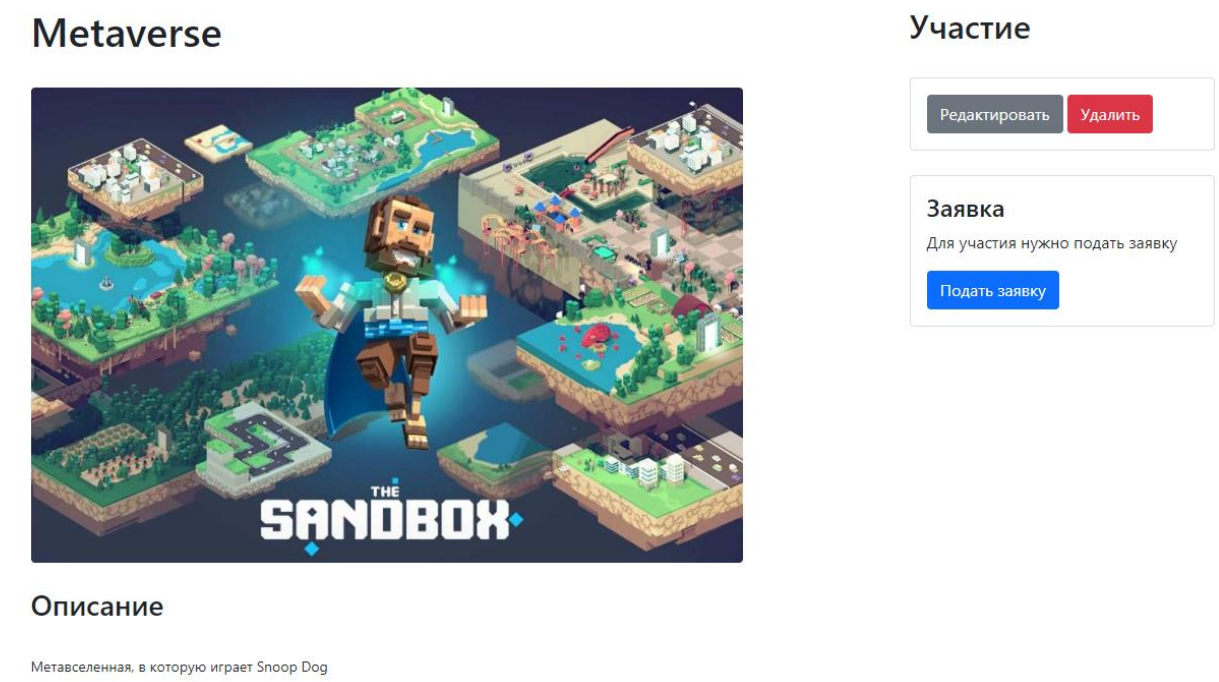

Рисунок 3.6 - Страница с информацией о проекте

После подачи заявки появляется её статус, который имеет значение по умолчанию - «Запрошено» (рисунок 3.7).

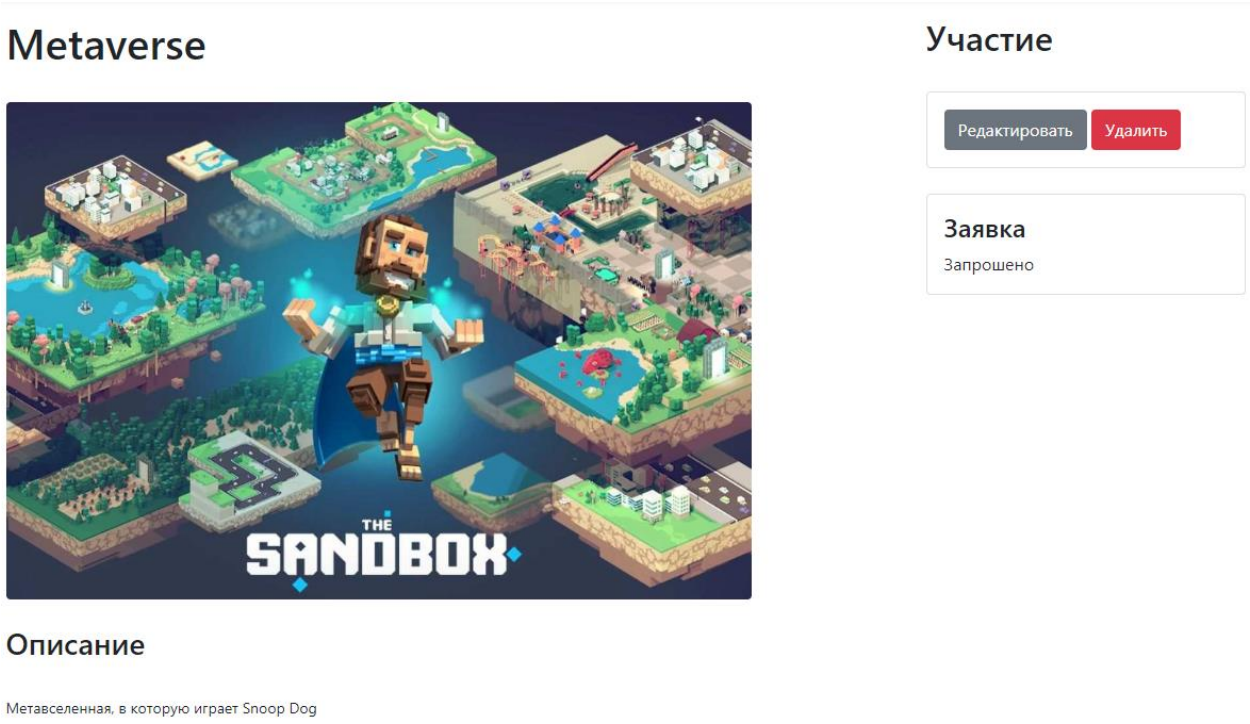

Рисунок 3.7 - Статус заявки «Запрошено»

Когда заявка пользователя принята, её статус принимает значение «Одобрено» (рисунок 3.8).

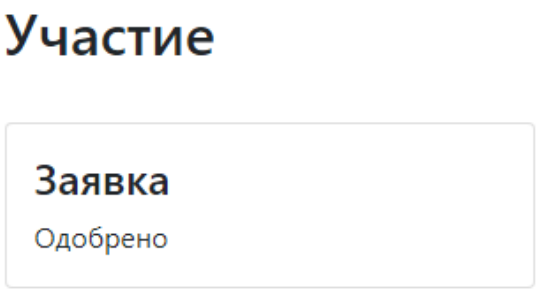

Рисунок 3.8 - Статус заявки «Одобрено»

При отклонении заявки пользователя статус принимает значение «Отклонено» (рисунок 3.9).

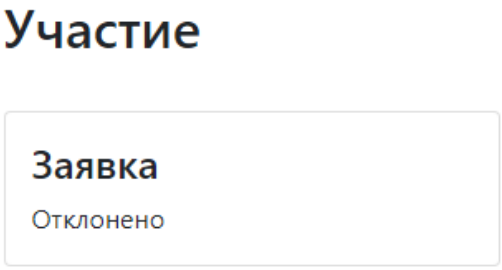

Рисунок 3.9 - Статус заявки «Отклонено»

Пользователь может инвестировать свои средства, когда статус заявки имеет значение «Одобрено» (рисунок 3.10).

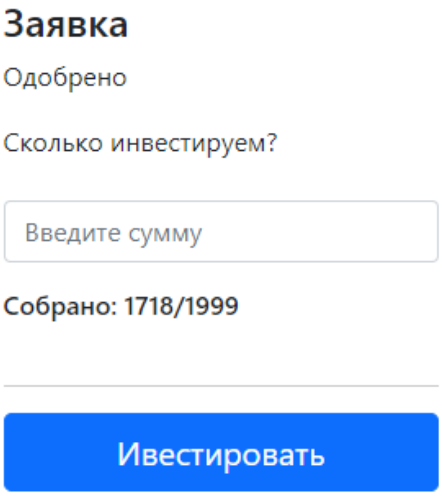

Рисунок 3.10 - Окно инвестирования

Далее будут представлены возможности администратора. После авторизации администратора на главной странице можно увидеть появившиеся ссылки «Добавить» и «Заявки» (рисунок 3.11).

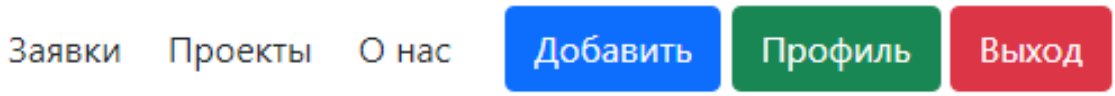

Рисунок 3.11 - Новые ссылки

Нажав на кнопку «Добавить», администратор будет перенаправлен на страницу добавления проекта. После ввода требуемых данных, необходимо нажать на кнопку «Добавить проект» (рисунок 3.12).

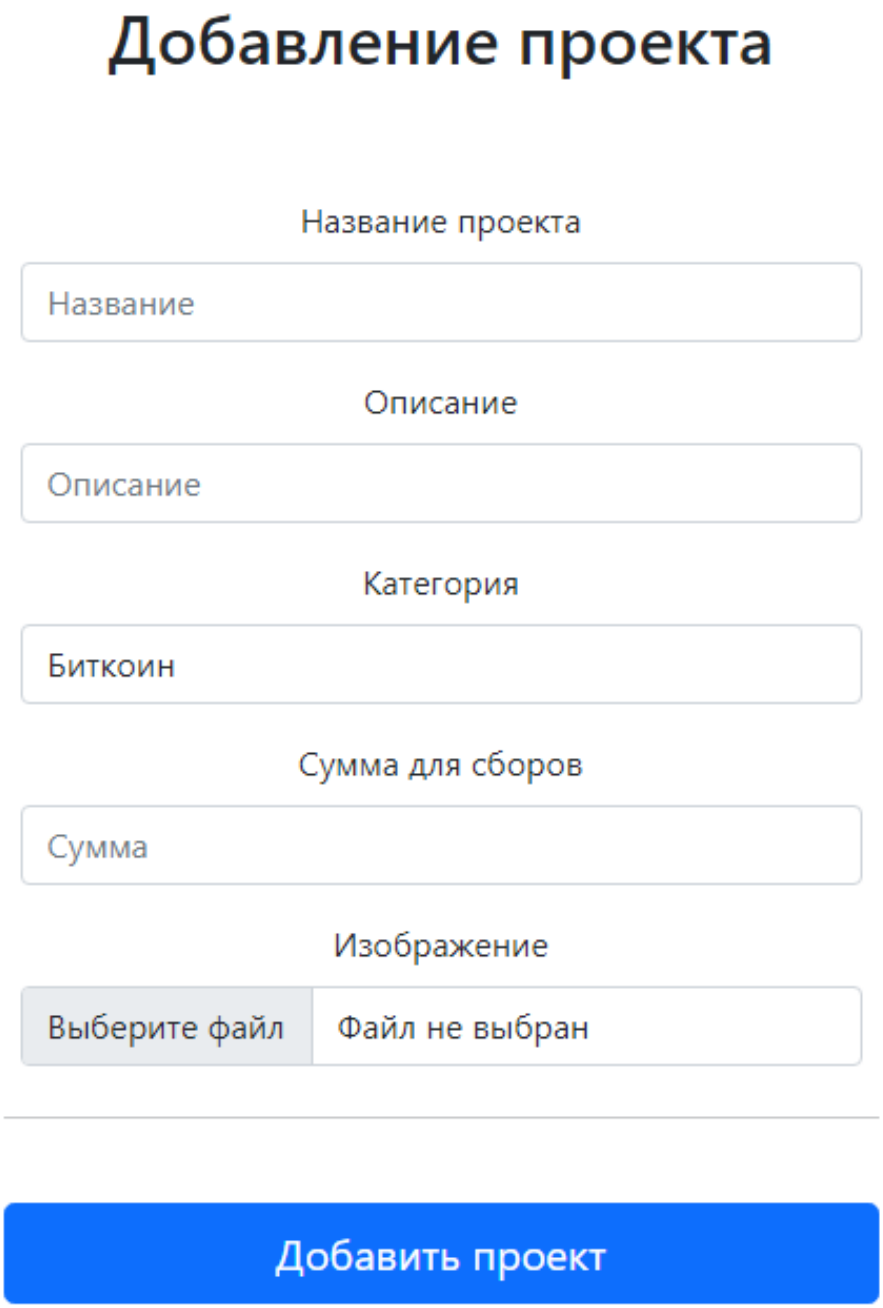

Рисунок 3.12 - Страница добавления проекта

После проделанных действий произойдёт перенаправление администратора на страницу отображения проектов, где он может видеть успешно добавленный проект (рисунок 3.13).

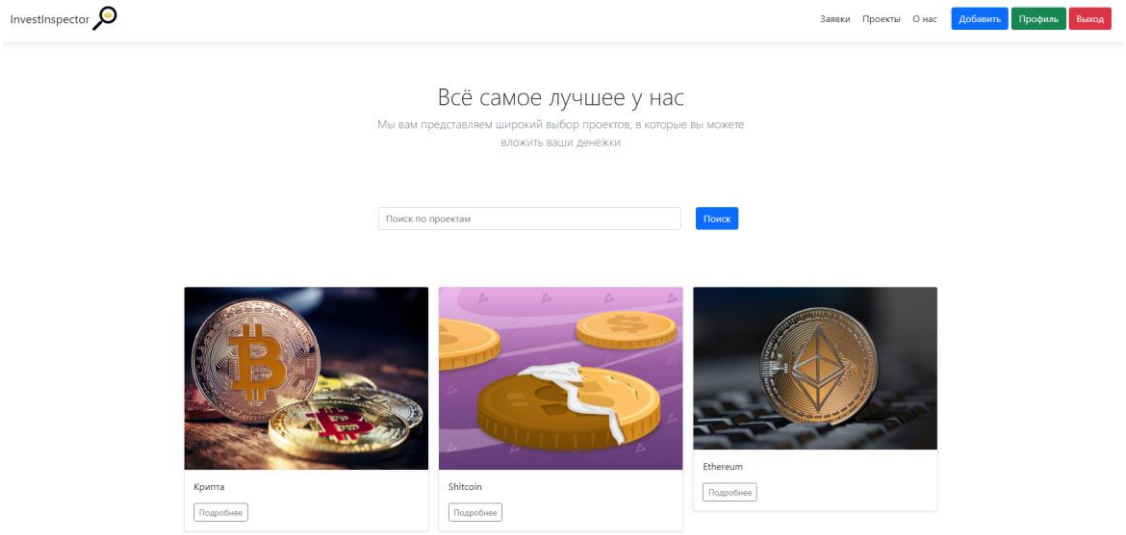

Рисунок 3.13 - Добавленный проект

Чтобы перейти к самому проекту, необходимо нажать на ссылку «Подробнее». Администратор может увидеть, что на странице появились две новые кнопки – «Редактировать» и «Удалить» (рисунок 3.14).

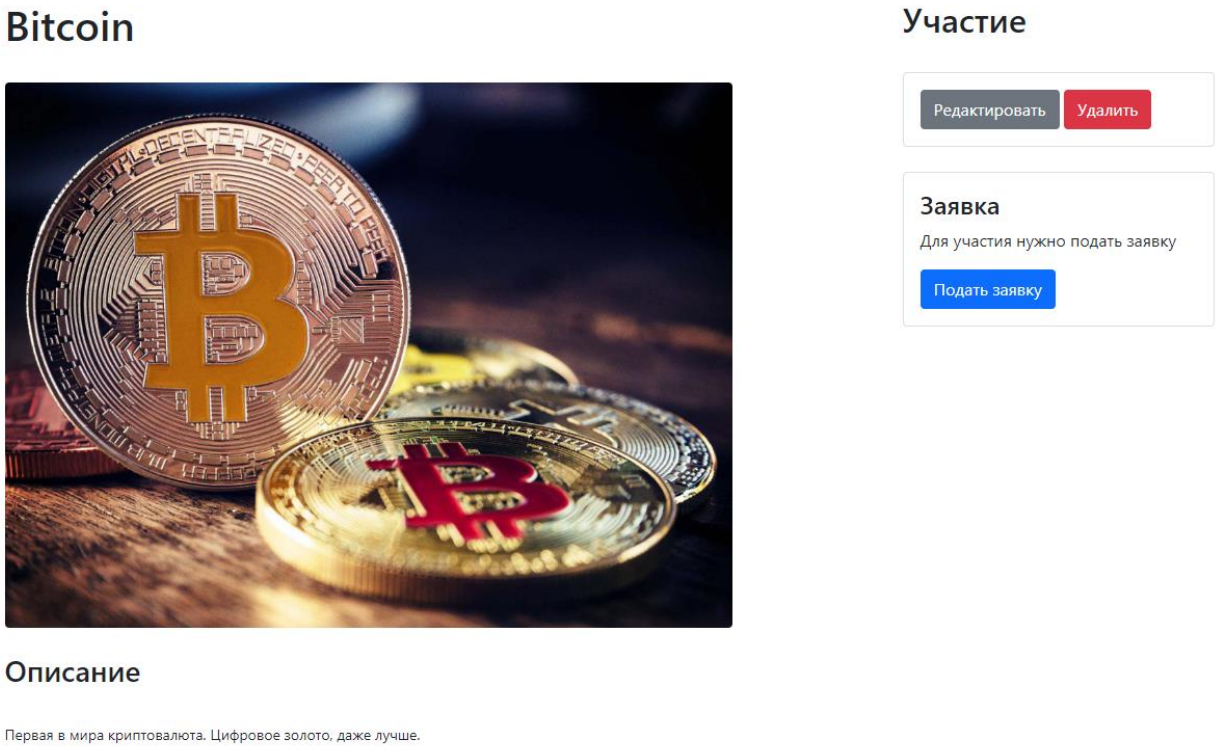

Рисунок 3.14 - Станица проекта «Крипта»

После нажатия на кнопку «Редактировать», происходит перенаправление администратора на страницу редактирования данных о проекте. Внеся нужные изменения, администратор нажимает кнопку «Сохранить» и совершается его перенаправление на страницу с проектами (рисунок 3.15).

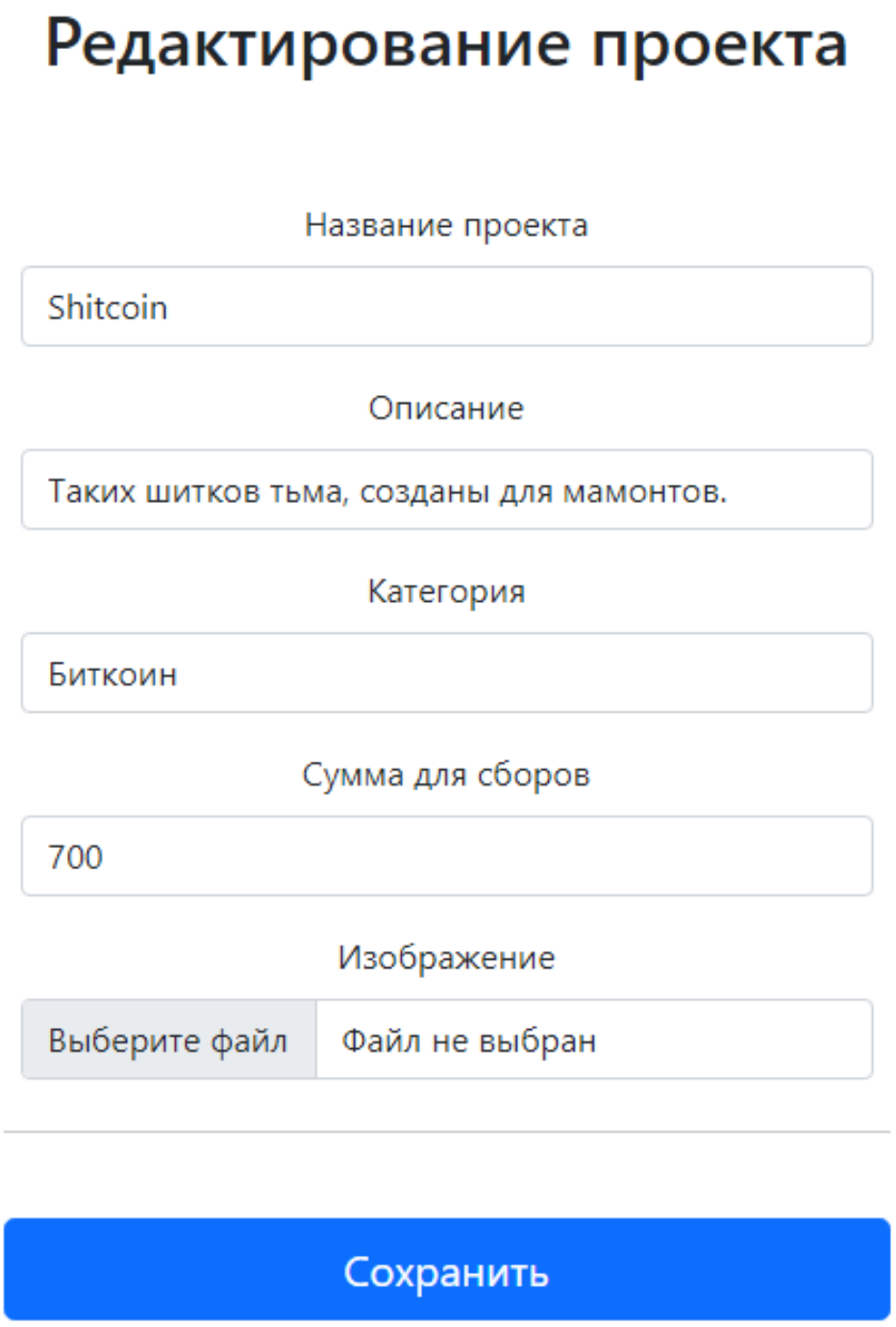

Рисунок 3.15 - Страница редактирования проекта

Администратор может заметить, что данные о проекте изменились (рисунок 3.16).

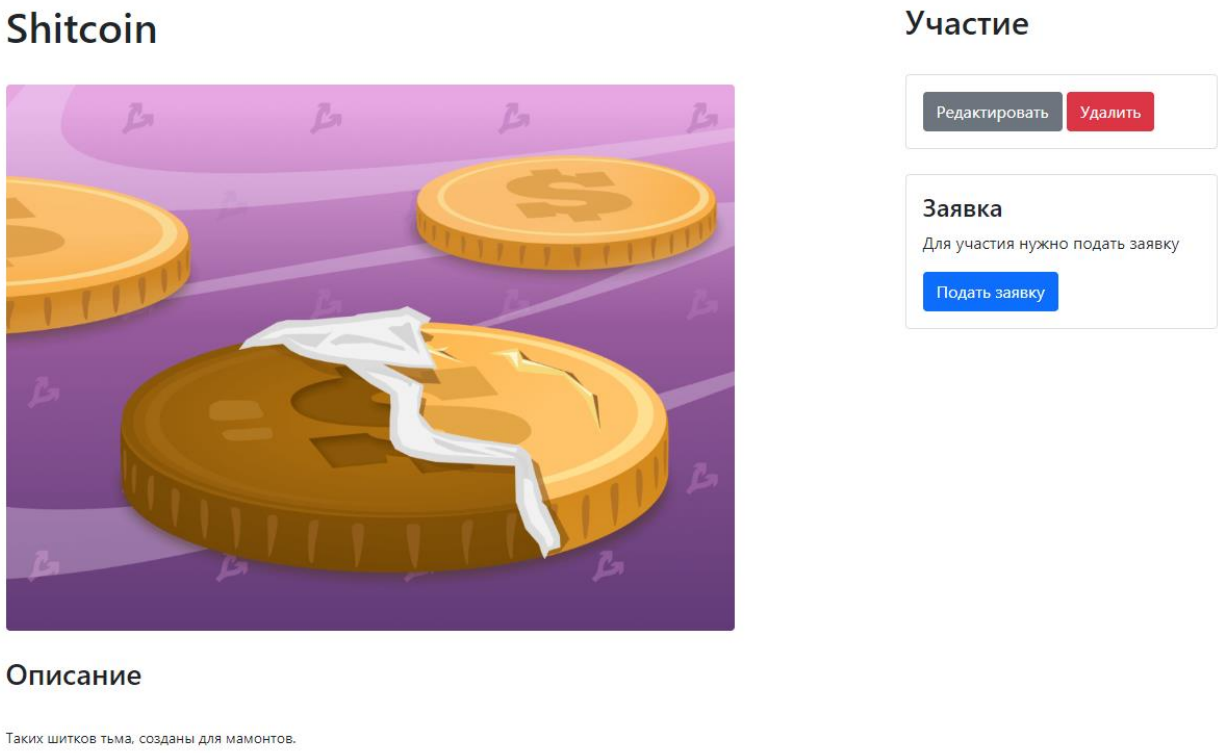

Рисунок 3.16 – Изменения вступили в силу

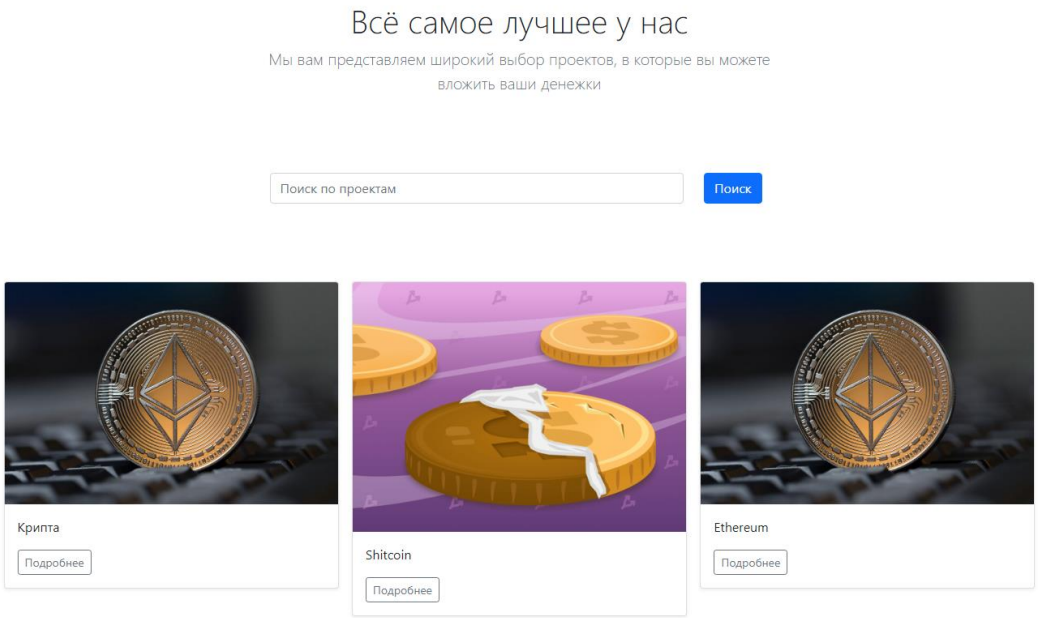

Рисунок 3.17 - Страница с проектами

Также, нажав на кнопку «Удалить» на странице проекта, запись о нём удаляется из базы данных - проекта больше нет в системе (рисунок 3.18).

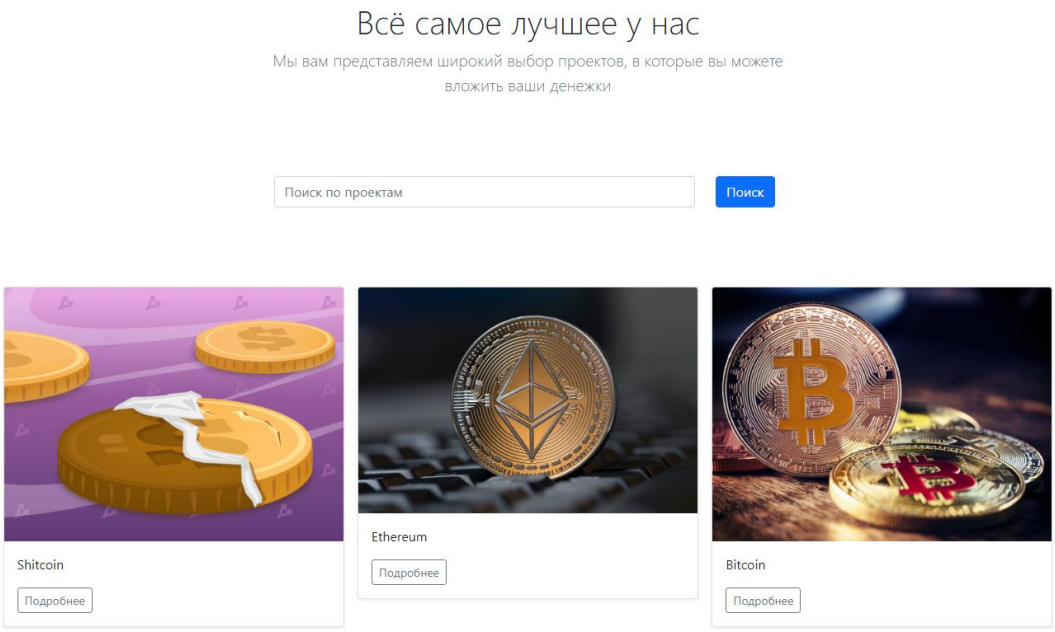

Рисунок 3.18 - Проекта больше нет в системе

Последняя, но не по важности возможность администратора - просмотр заявок и изменение их статусов. Для их просмотра нужно перейти по ссылке «Заявки». На данной странице отображаются все собранные данные по заявке, а также есть кнопки, с помощью которых можно менять её статус (рисунок 3.19).

Пользователь	ID Проекта	Статус	Действие
admin	Metaverse	Запрошено	Запрошено Одобрено Отклонено
user4	Metaverse	Одобрено	Запрошено Одобрено Отклонено
user3	Shitcoin	Одобрено	Запрошено Одобрено Отклонено
user2	Ethereum	Одобрено	Запрошено Одобрено Отклонено
user1	Ethereum	Одобрено	Запрошено Одобрено Отклонено
user1	Bitcoin	Одобрено	Запрошено Одобрено Отклонено

Рисунок 3.19 - Страница с заявками пользователей

Также в процессе интеграции приложения с блокчейном были добавлены некоторые изменения на страницу проекта. Добавлена ссылка на подключения крипто кошелька, а также представлено готовое инвестиционное решение на рисунке 3.20.

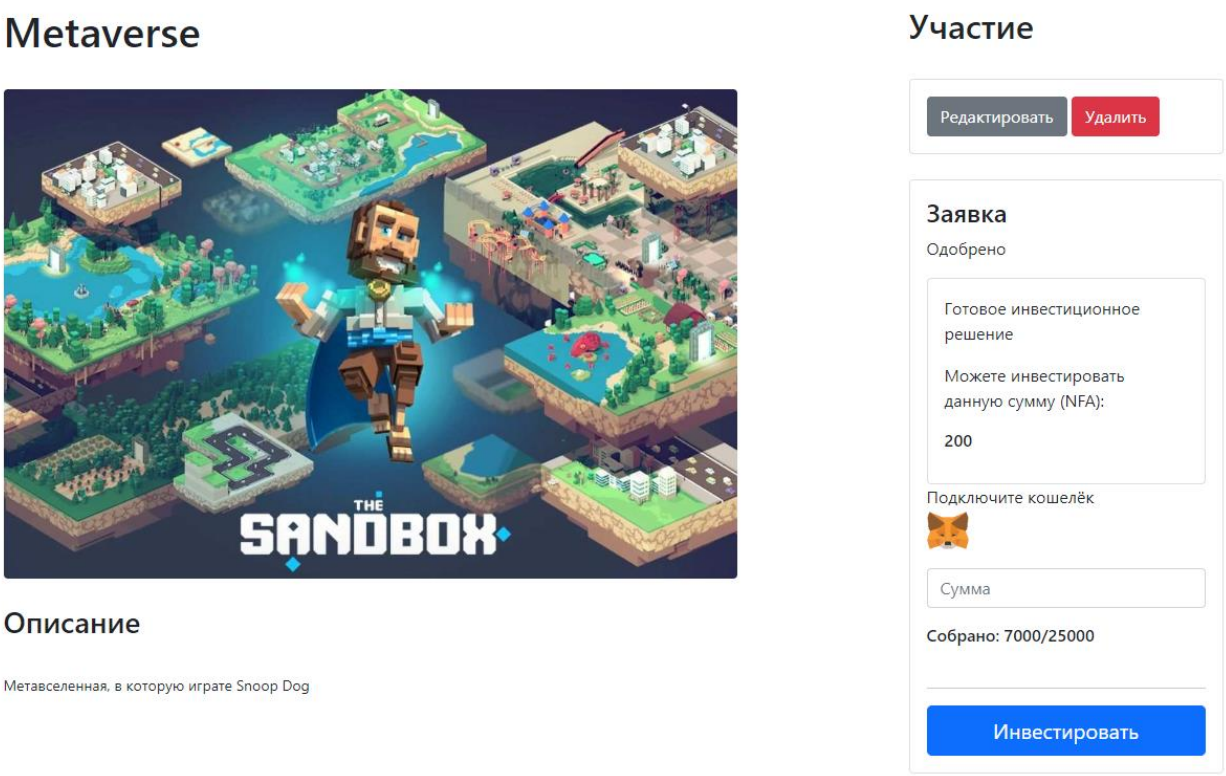

Рисунок 3.20 – Страница проекта с готовым инвестиционным решением и функцией подключения кошелька

4. ТЕСТИРОВАНИЕ

4.1. Проведение тестирования и отладки

Тестирование - основная часть разработки системы, которая позволяет выявлять ошибки в программном коде, непредвиденное поведение алгоритмов и минимизировать последствия этих ошибок [19].

Тестирование данной информационной системы проводилось в несколько этапов: проверка функционала доступного посетителям веб-приложения, проверка функционала пользователя, зарегистрированного в системе, проверка функционала администратора. Каждый из этих этапов был подвержен ручному тестированию с использованием графического интерфейса информационной системы.

Таблица 4.1 - Результаты тестирования общего функционала

Название функции	**Входные данные**	**Ожидаемый результат**	**Полученный результат**	**Вывод**
Проверка данных при заполнении формы регистрации	Отправка пустой формы	Под каждым ошибочным полем появляется подсказка	Под каждым ошибочным полем появляется подсказка	Функция работает верно
Отправка формы регистрации	Заполнены все поля	Пользователь будет зарегистрирован в системе	Пользователь зарегистрирован в системе	Функция работает верно
Запуск сессии пользователя	Заполненная форма входа	Пользователь перенаправлен на главную страницу, данные на главной странице изменены из-за успешного входа	Пользователь перенаправлен на главную страницу, данные на главной странице изменены из-за успешного входа	Функция работает верно

Таблица 4.2 - Результаты тестирования функционала пользователя

Название функции	**Входные данные**	**Ожидаемый результат**	**Полученный результат**	**Вывод**
Редактирование профиля	Корректные данные	Профиль отредактирован	Профиль отредактирован	Функция работает верно
Подача заявки	Кнопка «Подать заявку» была нажата	Кнопка «Подать заявку» пропала, появился статус «Запрошено»	Кнопка «Подать заявку» пропала, появился статус «Запрошено»	Функция работает верно
Инвестирование в проект	Введена корректная сумма, нажата кнопка «Инвестировать»	Сумма собранных средств изменилась, баланс пользователя изменился	Сумма собранных средств изменилась, баланс пользователя изменился	Функция работает верно

Таблица 4.3 – Результаты тестирования функционала администратора

Название функции	**Входные данные**	**Ожидаемый результат**	**Полученный результат**	**Вывод**
Добавление проекта	Корректные данные	Проект добавлен	Проект добавлен	Функция работает верно
Редактирование проекта	Корректные данные	Информация о проекте изменена	Информация о проекте изменена	Функция работает верно
Удаление проекта	Кнопка «Удалить» на странице проекта нажата	Проект удалён из системы	Проект удалён из системы	Функция работает верно
Изменение статуса заявки пользователя	Нажата кнопка «Одобрено» на странице заявок	Статус заявки обновлён	Статус заявки остался неизменным	Функция не работает

ЗАКЛЮЧЕНИЕ

В ходе курсового проектирования была достигнута поставленная цель, а именно, разработана информационная система краудинвестинговой платформы «Invest Inspector». Поставленная цель была достигнута путём выполнения задач курсового проекта.

В итоге курсового проектирования создана система, практическая значимость которой заключается в предоставлении пользователю выбора проекта для инвестирования и готового решения по инвестированию, сборе средств участников и дальнейшем инвестировании собранных денег.

Была проанализирована предметная область. На основе её анализа были разработаны и описаны следующие диаграммы: последовательности, вариантов использования и функциональная. UML-диаграмма последовательности дала понять, каким образом осуществляется порядок действий пользователя при работе с информационной системой. UML диаграмма вариантов использования отображает роли пользователей в системе и их возможности взаимодействия с ней. Функциональная UML диаграмма отобразила основные возможности ролей в системе, а также внешние взаимодействия системы с базой данных. В системе будут участвовать пользователи трёх видов: гость – посетитель приложения; сам пользователь – участник, зарегистрировавшийся в системе; администратор – имеет все права, отвечает за весь функционал в проекте. В проекте реализована функция автоматической поддержки пользователей, которая подразумевает под собой предоставление персональных рекомендаций по инвестированию.

При выборе средств разработки были определены языки программирования, система управления базы данных, IDEA для разработки веб-приложений и написания кода, веб-интерфейс работы с базой данных, средство установки локального сервера для тестирования приложения. Всё выше перечисленное было использовано для разработки функции

автоматизации. Выбор был осуществлён с помощью анализа достоинств и недостатков существующих решений.

По итогу проектирования базы данных были разработаны: логическая схема данных, инфологическая и даталогическая модель, а также диаграммы ER-типа. Все они показали будущую структуру базы данных. Всего было спроектировано на схеме данных восемь таблиц (comments, projects, categories, users, requests, bets, roles, statuses).

Разработан общий алгоритм системы и алгоритм функции автоматизации, каждый из которых помог предотвратить совершение ошибок при разработке приложения и избежать сбои при работе с модулем готового решения по инвестированию.

Созданы макеты меню приложения, формы подачи заявки, статуса поданной заявки, формы выбора готового решения по инвестированию и формы инвестирования средств в проект. По данным макетам разработан пользовательский интерфейс приложения.

Описаны процессы создания базы данных, её таблиц и связей. Также были реализованы SQL запросы на выборку из базы данных критериев для расчёта аддитивного критерия и поиска их максимальных значений.

Было описано подключение приложения к базе данных с помощью файла .env, а также, описан принцип написания кода приложения с помощью языков программирования HTML, CSS и PHP и его реализация. Присутствует объяснение работы функции автоматизации в коде – сбор данных по критериям и подсчёт рейтинга пользователей по ним.

Главное достоинство реализованной информационной системы – пользователю максимально облегчена задача в выборе своей стратегии инвестирования и представлена возможность инвестирования средств в проекты. В будущих версиях приложения планируется разработать смарт-контракт и провести его аудит, добавить возможность голосования за новые проекты и писать комментарии к проектам.

СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ

Используемая литература

1. Мюллер, Д. П. C# для чайников / Д. П. Мюллер. – Киев : Издательство «Диалектика», 2019. – 608 с.
2. Никсон, Р. Создаём динамические веб-сайты с помощью PHP, MySQL, JavaScript, CSS и HTML5. 4-ое издание / Р. Никсон. – Санкт-Петербург : Издательство «Питер», 2021. – 768 с.
3. Скляр, Д. Изучаем PHP 7 / Д. Скляр. – Санкт-Петербург : Издательство «Питер», 2017. – 464 с.
4. Скотт, А. Разработка на JavaScript. Построение кроссплатформенных приложений с помощью GraphQL, React, React Native и Electron / А. Скотт. – Санкт-Петербург : Издательство «Питер», 2021. – 320 с.
5. Стаффер, М. Laravel. Полное руководство. 2-е издание / М. Стаффер. – Санкт-Петербург : Издательство «Питер», 2020. – 512 с.
6. Флэнаган, Д. JavaScript. карманный справочник / Д. Флэнаган. – Санкт-Петербург : Издательство «Питер», 2018. – 320 с.
7. Шварц, Б. MySQL по максимуму. 3-е издание / Б. Шварц. – Санкт-Петербург : Издательство «Питер», 2018. – 864 с.

Интернет-источники

8. База данных. Oracle [Электронный ресурс]. – Режим доступа: http://oracle.com/database. – Дата доступа: 10.05.2022.
9. Внедрение SQL в интернет. PhpMyAdmin [Электронный ресурс]. – Режим доступа: https://www.phpmyadmin.net/docs/. – Дата доступа: 18.05.2022.
10. Документация OpenServer [Электронный ресурс]. – Режим доступа: http://www.ospanel.io/docs/. – Дата доступа: 21.05.2022.
11. Документация PhpStrom. JetBrains [Электронный ресурс]. – Режим доступа: https://www.jetbrains.com/ru-ru/phpstorm/documentation/. – Дата доступа: 20.05.2022.

12. Документация. Bootstrap [Электронный ресурс]. – Режим доступа: http://www.getbootstrap.com/docs/5.0/getting-started/introduction. – Дата доступа: 07.05.2022.
13. Документация. Laravel [Электронный ресурс]. – Режим доступа: http://www.laravel.com/docs/8.x/. – Дата доступа: 22.05.2022.
14. Документация. MySQL [Электронный ресурс]. – Режим доступа: http://dev.mysql.com/doc/. – Дата доступа: 11.05.2022.
15. Документация. PHP [Электронный ресурс]. – Режим доступа: http://www.php.net/docs.php. – Дата доступа: 15.05.2022.
16. Интерфейс — что это такое? Определение, значение, перевод. Вдох [Электронный ресурс]. – Режим доступа: https://vdoh.ru/chto-takoe/interface. – Дата доступа: 25.05.2022.
17. Лекция 7. Инфологическое моделирование. Казахстан университет "Алатау" [Электронный ресурс]. – Режим доступа: http://bourabai.ru/dbt/dbms/7.htm. – Дата доступа: 20.05.2022.
18. Платформа HTML Academy [Электронный ресурс]. – Режим доступа: https://htmlacademy.ru/study. – Дата доступа: 15.05.2022.
19. Руководство для начинающих:Git за полчаса. Proglib [Электронный ресурс]. – Режим доступа: https://proglib.io/p/git-for-half-an-hour/. – Дата доступа: 16.05.2022.
20. Самоучитель HTML4. HTMLBook [Электронный ресурс]. – Режим доступа: http://htmlbook.ru/samhtml. – Дата доступа: 10.05.2022.
21. Самоучитель HTML5. HTML5BOOK [Электронный ресурс]. – Режим доступа: https://html5book.ru/html-html5/. – Дата доступа: 17.05.2022.
22. Современный учебник JavaScript. JavaScript [Электронный ресурс]. – Режим доступа: https://learn.javascript.ru/. – Дата доступа: 10.05.2022.
23. Термин ICO. Лайкни [Электронный ресурс]. – Режим доступа: https://www.likeni.ru/glossary/ico/. – Дата доступа: 11.05.2022.

24. Уроки Laravel. itProger [Электронный ресурс]. – Режим доступа: https://itproger.com/course/laravel/. – Дата доступа: 19.05.2022.
25. Что такое алгоритм. КтоНаНовенького [Электронный ресурс]. – Режим доступа: https://ktonanovenkogo.ru/voprosy-i-otvety/algoritm-chto-ehto-takoe-vidy-algoritmov.html. – Дата доступа: 21.05.2022.
26. Что такое функциональная схема. Школа для электрика [Электронный ресурс]. – Режим доступа: http://electricalschool.info/main/electroshemy/849-chto-takoe-funkcionalnaja-skhema.html. – Дата доступа: 17.05.2022.
27. Язык графического описания UML. Википедия [Электронный ресурс]. – Режим доступа: https://ru.wikipedia.org/wiki/UML. – Дата доступа: 15.05.2022.

PageController

```php
<?php

namespace App\Http\Controllers;

use App\Models\Bet;
use App\Models\Category;
use App\Models\Project;
use App\Models\Req;
use App\Models\Status;
use Illuminate\Support\Facades\Auth;
use Illuminate\Support\Facades\DB;
use function GuzzleHttp\Psr7\_caseless_remove;

class PageController extends Controller
{
    public function index(){
        return view('index');
    }
    public function signup(){
        return view('sign-up');
    }
    public function login(){
        return view('login');
    }
    public function profile(){
        return view('profile');
    }
    public function projects(){
        $projects = Project::orderBy('id', 'desc')->get();
        return view('projects', ['projects' => $projects]);
    }
    public function project_add(){
        $categories = Category::all();
        return view('project-add', ['categories' => $categories]);
    }
    public function project_single($id){
        $project = Project::findOrFail($id);
        $requests = Auth::user()->requests->where('project_id', $id)->all();
        $statuses = Status::orderBy('id', 'desc')->get();
        $data = DB::table("users")
            ->join("requests", function($join){
                $join->on("users.id", "=", "requests.user_id");
            })
            ->join("projects", function($join){
                $join->on("projects.id", "=", "requests.project_id");
            })
```

Весь код, который Вы писали сами также дальше должен быть…