Кыргызский Государственный Технический университет им. И.Раззакова

Факультет информационных технологии

Кафедра Программное обеспечение Компьютерных систем

Пояснительная записка по дисциплине

«ППР ПО»

На тему: «Платформа платного контента OnlyPay»

Выполнил: студент группы ПИ (б) 3-20

Стройков Игорь

Проверила: ст. преп. Мукашова Ж.С.

Бишкек - 2022

Оглавление

[Введение . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 3](file:///C:\Users\User\Desktop\primer_otcheta_PPO1.docx#_Toc513457250)

[Актуальность работы. . . . . . . . . . . . . . . . . . . . . . . . . . . . . 3](file:///C:\Users\User\Desktop\primer_otcheta_PPO1.docx#_Toc513457251)

Обзор и анализ существующих систем . . . . . . . . . . . . . . . . . . . 4

[Глава 1 Анализ и разработка требований . . . . . . . . . . . . . . . . . . .](file:///C:\Users\User\Desktop\primer_otcheta_PPO1.docx#_Toc513457253) 5

1.1 [Цель работы. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .](file:///C:\Users\User\Desktop\primer_otcheta_PPO1.docx#_Toc513457252) 5

1.2 [Основания для разработки. . . . . . . . . . . . . . . . . . . . . . . .](file:///C:\Users\User\Desktop\primer_otcheta_PPO1.docx#_Toc513457252) 4

1.3 [Анализ требований . . . . . . . . . . . . . . . . . . . . . . . . . . . .](file:///C:\Users\User\Desktop\primer_otcheta_PPO1.docx#_Toc513457252) 5

[1.4 Функциональные требования. . . . . . . . . . . . . . . . . . . . . . .](file:///C:\Users\User\Desktop\primer_otcheta_PPO1.docx#_Toc513457262) 5

1.4.1 Функции ………………………………………………..…………..5

1.4.2 Исходные данные ……………………………………………...…..6

1.4.3 Организация входных и выходных данных……………………...6

1.4.4 Требования к надежности …………………………………..…… 6

1.4.5 Требования к составу и параметрам технических средств ……. 6

[1.5 Источники разработки. . . . . . . . . . . . . . . . . . . . . . . . . . .](file:///C:\Users\User\Desktop\primer_otcheta_PPO1.docx#_Toc513457268) 6

Глава 2 Конструкторские работы . . . . . . . . . . . . . . . . . . . . . . . 7

2.1 Диаграмма Ганте . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 7

2.2 Use Case . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 8

2.3 Диаграмма деятельности . . . . . . . . . . . . . . . . . . . . . . . . . 9

2.4 Диаграмма классов . . . . . . . . . . . . . . . . . . . . . . . . . . . 11

2.5 Диаграмма последовательности . . . . . . . . . . . . . . . . . . . . . 15

2.6 Диаграмма БД . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 17

Глава 3 Документо-ориентированные работы . . . . . . . . . . . . . . . . 19

3.1 Руководство пользователя . . . . . . . . . . . . . . . . . . . . . . . 19

3.2 Руководство программиста . . . . . . . . . . . . . . . . . . . . . . . 28

3.3 Демонстрация кода . . . . . . . . . . . . . . . . . . . . . . . . . . . 55

3.4 Расчёт стоимости программы . . . . . . . . . . . . . . . . . . . . . . 29

Заключение . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 30

Оглавление иллюстраций

[Рисунок 1 - Иллюстрация плана, график работ по проекту OnlyPay 7](#_Toc121214798)

[Рисунок 2 - Сценарий использования программного продукта OnlyPay 8](#_Toc121214799)

[Рисунок 3 - Диаграмма деятельности (автор) 9](file:///D:\0.%20Универ\ППР%20ПО\Отчёт.docx#_Toc121214800)

[Рисунок 4 - Диаграмма деятельности (подписчик) 10](file:///D:\0.%20Универ\ППР%20ПО\Отчёт.docx#_Toc121214801)

[Рисунок 5 - Диаграмма классов 12](file:///D:\0.%20Универ\ППР%20ПО\Отчёт.docx#_Toc121214802)

[Рисунок 6 - Диаграмма последовательности (часть 1) 16](file:///D:\0.%20Универ\ППР%20ПО\Отчёт.docx#_Toc121214803)

[Рисунок 7 - Диаграмма последовательности (часть 2) 17](file:///D:\0.%20Универ\ППР%20ПО\Отчёт.docx#_Toc121214804)

[Рисунок 8 - Диаграмма базы данных 18](#_Toc121214805)

[Рисунок 9 – Меню сверху (до авторизации) 20](#_Toc121214806)

[Рисунок 10 - Окно регистрации 20](#_Toc121214807)

[Рисунок 11 - Меню сверху (до авторизации) 20](#_Toc121214808)

[Рисунок 12 - Окно авторизации 21](#_Toc121214809)

[Рисунок 13 - Меню сверху (после авторизации) 21](#_Toc121214810)

[Рисунок 14 - Окно личного профиля 22](#_Toc121214811)

[Рисунок 15 - Меню пополнения баланса 22](#_Toc121214812)

[Рисунок 16 - Меню вывода денеж. средств 23](#_Toc121214813)

[Рисунок 17 - Окно статистики 23](#_Toc121214814)

[Рисунок 18 - Список авторов на Главной странице 24](#_Toc121214815)

[Рисунок 19 - Окно профиля автора 24](#_Toc121214816)

[Рисунок 20 - Окно оформления подписки 25](#_Toc121214817)

[Рисунок 21 - Окно постов 25](#_Toc121214818)

[Рисунок 22 - Навигационное меню 26](#_Toc121214819)

[Рисунок 23 - Окно написания комментария 26](#_Toc121214820)

[Рисунок 24 - Меню сверху 26](#_Toc121214821)

[Рисунок 25 - Окно добавления нового поста 27](#_Toc121214822)

[Рисунок 26 - Меню навигации 27](#_Toc121214823)

[Рисунок 27 - Окно редактирования поста 28](#_Toc121214824)

[Рисунок 28- Конструктивная модель затрат 36](#_Toc121214825)

[Рисунок 29 - Результат расчёта модели COCOMO 36](#_Toc121214826)

**Введение**

Создание шедевра может занимать слишком много времени, а зарабатывать на нём сразу иногда не получается. Бывает, что ценность некоторых произведений искусства признаётся только спустя десятилетия, а то и столетия. В условиях постоянной гонки за заработком сегодняшние творцы не могут ждать столетия, пока их работы оценят по достоинству. Кто-то пытается создавать произведения быстрее и делать их такими, какие понравятся публике. Но и с этим вариантом есть проблемы: издательства и площадки для творчества берут большие проценты с продаж и за сотрудничество, из-за чего заработок может сильно сократиться после вычетов расходов. Для решения вышестоящих проблем была разработана площадка, на которой авторы могут свободно творить и получать деньги и поддержку от спонсоров напрямую, а не через посредников.

Суть идеи - “плата за творчество напрямую” позволит стабильно зарабатывать, имея даже небольшую аудиторию.

**Актуальность работы**

Данная платформа позволяет зарабатывать авторам не за счёт рекламы, а за счёт постоянной подписки. Автор на месяц вперед знает, сколько он сможет заработать, а также отслеживать актуальность своего творчества. Данная площадка не является Краудфандингом. Краудфандинг направлен на сбор средств для одной цели, и как только достаточная сумма собрана, сбор закрывается. Данная же платформа позволяет постоянно перечислять деньги на счёт автора и регулярно получать контент.

**Обзор и анализ существующих систем**

**Платформа «Patreon», платформа «Boosty»**

* Высокая комиссия (8-15%)
* Ограничения цензурой
* Отсутствие местных электронных кошельков
* С одной стороны, достаточно функциональный интерфейс, но и в тоже время сложный для начинающих и не опытных пользователей
* Отсутствие краткосрочных и единоразовых подписок (авторы должны постоянно выкладывать контент, выполняя условия подписки долгосрочных подписок, что может сказаться на качестве или состоянии автора)
* Аналоги не продвигают авторов, новичкам сложно найти свою аудиторию
* Иностранные платежи: оплата контента и зарплата автору производится в иностранной валюте
* Вывод средств только раз в месяц в определенное число

**Глава 1. Анализ и разработка требований**

* 1. **Цель работы**

Предоставить возможность клиенту продавать свой контент, загружая его на платформу.

Использование данной платформы позволяет автору больше концентрироваться на качестве своего творчества и меньше беспокоиться о заработке. Также автор может отслеживать свою прибыль и анализировать актуальность своего творчества.

* 1. **Основания для разработки**

Основанием для разработки программы является практическое задание Кыргызского государственного технического университета имени И. Раззакова.

Наименование работы: «Only Pay»

Исполнители: студент группы ПИ(б) 3-20

Соисполнители: отсутствует.

Сроки исполнения: 1 сентября – 17 декабря 2022 года

* 1. **Анализ требований**

Программный продукт «Only Pay» предназначен для обеспечения удобства и эффективности реализации и заработка автора на своем контенте.

* 1. **Функциональные требования к разрабатываемой системе**

**1.4.1 Программа должна обеспечивать возможность выполнения следующих функций:**

*Функции пользователя(подписчик):*

* Регистрация пользователя;
* Авторизация пользователя;
* Пополнение баланса;
* Подписка на автора;
* Просмотр приобретенного контента;
* Связь с автором;
* Написание комментариев;

*Функции пользователя(автора):*

* Регистрация пользователя;
* Авторизация пользователя;
* Публикация контента;
* Отслеживание актуальности своего контента;
* Отслеживание прибыли от подписок;
* Снятие денежных средств;

*Функции сервера:*

* Подсчёт прибыли автора;
* Подсчет просмотров и подписок на контент;
* Хранение данных контента автора;
* Вывод денежных средств с баланса автора;
* Подсчет и вычет комиссии из прибыли автора;
* Принятие оплаты через банковскую карту или электронные кошельки;

**1.4.2 Исходные данные**

* Данные платежной системы автора/подписчика;
* Логин, пароль;
* Контент автора;
* Доступные тарифы по оплате подписки;
* БД комментариев;

**1.4.3 Организация входных и выходных данных**

* Входные данные поступают из веб-приложения пользователя (данные пользователя, логин, пароль, данные контента, платежные данные);
* Выходные данные – аутентификация, доступ, контент поступают с сервера в веб-приложение (браузер) пользователя;
* Данные хранятся в БД на серверах.

**1.4.4 Требования к надежности**

* Шифрование данных с длиной ключа 256 бит при отправке данных с браузера пользователя на сервер и дешифрование полученных с сервера данных веб-приложением в браузере пользователя;
* Авторизация и сессия организуются с помощью инструмента «JWT Token»;
* Шифрование паролей производится с помощью библиотеки «BCryptJS» ключом 2048-bit RSA с применением “Salt”;
* Предусмотреть валидацию вводимых данных пользователем при регистрации/авторизации/верификации и других полей.

**1.4.5 Требования к составу и параметрам технических средств**

* Требуемая версия Браузера Google Chrome - 105.0.5195.102;
* Возрастное ограничение: 18+

**1.5 Источники разработки**

* ГОСТ 34.602-89 – Техническое задание на создание автоматизированной системы.
* ГОСТ 19.201-78 - Техническое задание. Требования к содержанию и оформлению.

**2.0 Конструкторские работы**

**2.1 Диаграмма Ганте**



Рисунок 1 - Иллюстрация плана, график работ по проекту OnlyPay

( Диаграмма на рисунке №1 позволяет визуализировать управление проектом, структурирование и просмотр выполнения общих задач проекта. )

**2.2 Use Case – диаграмма варианта использования**



Рисунок 2 - Сценарий использования программного продукта OnlyPay

Диаграмма на рисунке №2 описывает функционал разрабатываемой платформы OnlyPay для двух групп пользователей: Группа «Автор» и группа «Подписчик».

Для обеих групп «Автор» и «Подписчик» доступны следующие общие функциональные возможности:

1. Регистрация – позволяет первоначально зарегистрировать новый профиль пользователя на платформе OnlyPay;
2. Авторизация – процесс проверки наличия зарегистрированного профиля пользователя на платформе OnlyPay;
3. Пополнение баланса – пополнение внутреннего счёта на платформе OnlyPay;

Для группы «Автор» доступны в отдельности следующие функциональные возможности:

1. Публикация поста – функция позволяет загрузить свой контент на платформу OnlyPay для дальнейшей «продажи» подписчикам;
2. Вывод денег с баланса – процесс вывода, заработанных с подписок, денежных средств с баланса профиля на эл. кошелёк пользователя.

Для группы «Подписчик» доступны в отдельности следующие функциональные возможности:

1. Выбор тарифа для подписки – предоставление выбора нескольких ценовых вариантов для оформления подписки;
2. Покупка подписки – процесс оформления платной подписки на автора, снятия денежной суммы с баланса подписчика, согласно выбранному тарифу;
3. Создание комментария – процесс написания собственной субъективной оценки под определенным постом.

**2.3 Диаграмма деятельности**

Рисунок 3 - Диаграмма деятельности (автор)

Бизнес-процессы диаграммы деятельности:

1. Регистрация – новый пользователь проходит первичную регистрацию;
2. Проверка наличия зарегистрированного профиля по указанным данным;
3. Авторизация – уже зарегистрированный пользователь проходит вход в систему;
4. Верификация – прохождение процедуры подтверждения своей личности путем загрузки своих документов;
5. Загрузка текста и медиа – процесс публикации поста путем заполнения всех форм в окне публикации;
6. Публикация поста – проверка системой на валидацию всех заполненных пользователем полей;
7. Просмотр статистики – получение информации о заработке пользователя как автора контента;
8. Сбор общей статистики – сбор всех данных всех подписок на автора, подсчёт общей прибыли, структурирование данных в ответ и отправка структурированной информации пользователю;
9. Вывод денег с баланса профиля – в случае наличия указанной для вывода суммы на балансе производится списание указанной суммы, вычет из этой суммы комиссии и результирующая сумма будет отправлена пользователю, комиссия отправится на баланс платформы OnlyPay;
10. Проверка наличия баланса – проверка наличия указанной суммы на балансе профиля пользователя, для вывода денежных средств.

Рисунок 5 - Диаграмма деятельности (подписчик)

Бизнес-процессы диаграммы деятельности:

1. Регистрация – новый пользователь проходит первичную регистрацию;
2. Проверка наличия зарегистрированного профиля по указанным данным;
3. Авторизация – уже зарегистрированный пользователь проходит вход в систему;
4. Пополнение баланса – выбор желаемой платежной системы для пополнения баланса;
5. Проверка наличия указанной суммы для пополнения баланса на счету пользователя;
6. Выбор профиля желаемого автора для подписки;
7. Просмотр профиля автора – в случае согласия подписаться на автора, необходимо оплатить тариф для подписки;
8. Проверка наличия необходимой суммы на балансе профиля OnlyPay;
9. Просмотр контента – в случае успешной подписки на автора, открывается для просмотра весь контент данного автора;
10. Написать комментарий – после выбора нужного поста автора, открывается возможность написать комментарий к данному посту.

**2.4 Диаграмма классов**

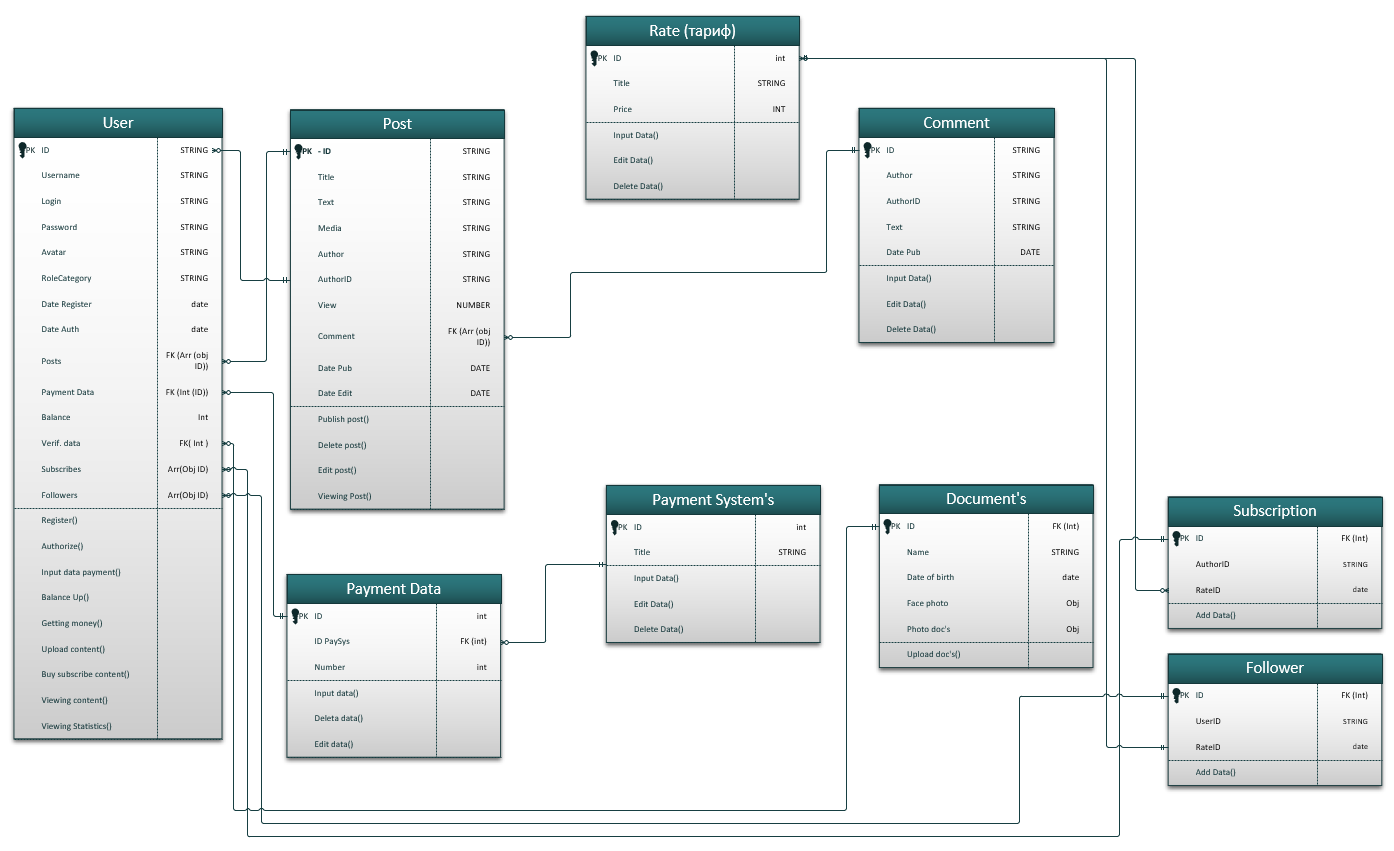


Рисунок 6 - Диаграмма классов

1. Класс User – отвечает за хранения структуры модели пользователя, которая хранит следующие поля:

* Username, Login, Password, Avatar, RoleCategory – тип строковый, хранят в себе строки;
* Date Register, Date Auth – тип Дата, хранят в себе дату регистрации и дату авторизации;
* Posts – тип данных Модели Post, содержит в себе внешние ключи объектов Модели Post;
* Payment data – тип данных Модели Payment data, содержит в себе внешние ключи объектов Модели Payment data;
* Balance – тип целочисленный, хранит в себе количество денежных средств;
* Verif. Data – тип данных Модели Document’s, хранит в себе вериф. данные о пользователе;
* Subscribes – тип данных Модели Subscription, хранит в себе данные ID подписки и ID ее тарифа;
* Followers – тип данных Модели Follower, хранит в себе данные ID подписки и ID ее тарифа;

Методы:

* Register – метод регистрации;
* Authorize - метод авторизации;
* Input Data Payment – метод указания платежных данных;
* Balance Up – метод пополнения баланса;
* Getting Money – метод вывода денежных средств с баланса;
* Upload Content – метод обновления своего контента;
* Buy Subscribe – метод покупки подписки;
* Viewing Content – метод просмотра контента;
* Viewing Statistics – метод просмотра статистики.

1. Класс Post – отвечает за хранение структуры модели поста, которая хранит следующие поля:

* Title, Text – тип данных Строковый, хранят в себе содержимое поста;
* Media – тип данных Строковый, хранит в себе название или ссылку на источник;
* Author, AuthorID – тип данных Строковый, хранят в себе строку имени пользователя и ID пользователя;
* View – тип данных Числовой, хранит в себе счётчик просмотра поста;
* Comment – тип данных Модели Comment, хранит в себе массив ID постов;
* Date Pub, Date Edit – тип данных Дата, хранят в себе данные о дате публикации и дате изменения поста;

Методы:

* Publish Post – метод публикации поста;
* Delete Post – метод удаления поста;
* Edit Post – метод редактирования поста;
* Viewing Post – метод просмотра поста.

1. Класс Rate - отвечает за хранение структуры модели поста, которая хранит следующие поля:

* Title – тип данных Строковый, хранит в себе название тарифа;
* Price – тип данных Числовой, хранит в себе стоимость тарифа.

Методы:

* Input Data – метод добавления данных;
* Edit Data – метод редактирования данных.

1. Класс Comment - отвечает за хранение структуры модели поста, которая хранит следующие поля:

* Author – тип данных Строковый, хранит имя пользователя;
* AuthorID – тип данных Строковый, хранит в себе ID пользователя;
* Text – тип данных Строковый, хранит в себе текст комментария;
* Date Pub – Тип данных Дата, хранит в себе дату публикации поста.

Методы:

* Input Data – метод добавления данных;
* Edit Data – метод редактирования данных.

1. Класс Payment Data - отвечает за хранение структуры модели поста, которая хранит следующие поля:

* IDPaySys – тип Строковый, хранит в себе ID платежной системы;
* Number – тип Числовой, хранит в себе реквизиты пользователя.

Методы:

* Input Data – метод добавления данных;
* Edit Data – метод редактирования данных.

1. Класс Payment System - отвечает за хранение структуры модели поста, которая хранит следующие поля:

* Title – тип данных Строковый, хранит в себе название платежной системы;

Методы:

* Input Data – метод добавления данных;
* Edit Data – метод редактирования данных.

1. Класс Document’s - отвечает за хранение структуры модели поста, которая хранит следующие поля:

* Name – тип Строковый, хранит в себе имя пользователя;
* Date of birth – тип Дата, хранит дату рождения пользователя;
* Face Photo – тип Строковый, хранит в себе название файла;
* Doc’s Photo – тип Строковый, хранит в себе название файла.

Методы:

* Input Data – метод добавления данных;
* Edit Data – метод редактирования данных.

1. Класс Subscription - отвечает за хранение структуры модели поста, которая хранит следующие поля:

* AuthorID – тип данных Строковый, хранит в себе ID автора, на которого оформлена подписка;
* RateID – тип данных Строковый, хранит в себе ID тарифа, с которым оформлена подписка.

Методы:

* Input Data – метод добавления данных;
* Edit Data – метод редактирования данных.

1. Класс Follower - отвечает за хранение структуры модели поста, которая хранит следующие поля:

* UserID – тип данных Строковый, хранит в себе ID подписчика, который оформил подписку;
* RateID – тип данных Строковый, хранит в себе ID тарифа, с которым оформлена подписка.

Методы:

* Input Data – метод добавления данных;
* Edit Data – метод редактирования данных.

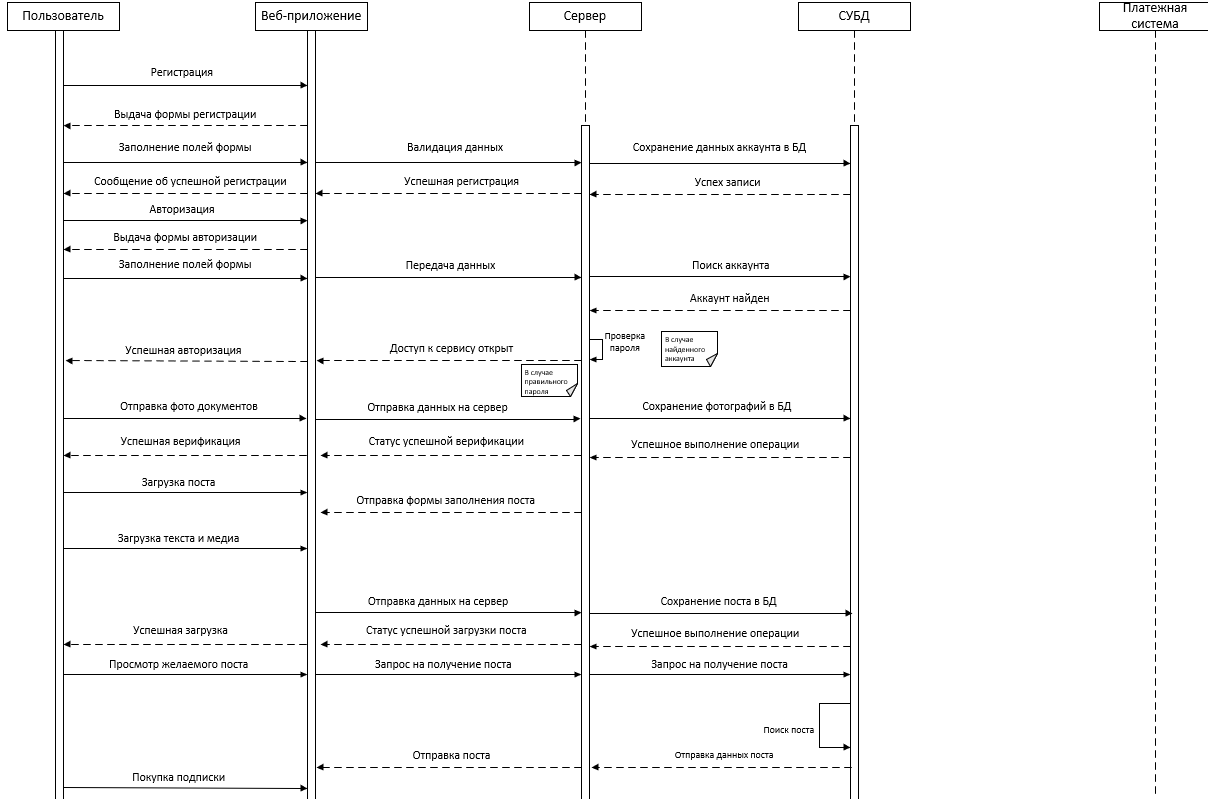
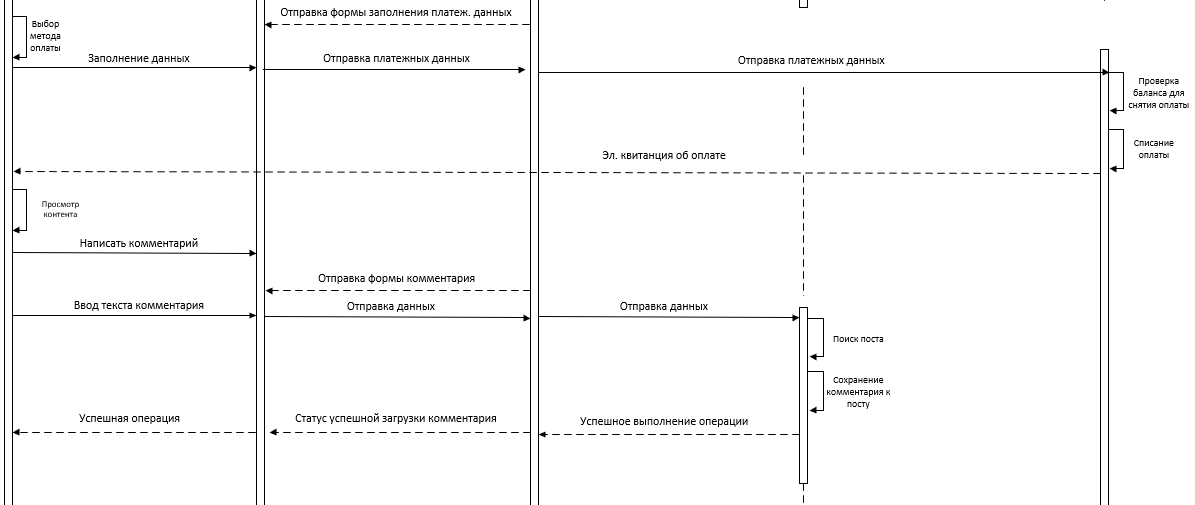
**2.5 Диаграмма последовательности**

Рисунок 7 - Диаграмма последовательности (часть 1)

Продолжение рисунка №5

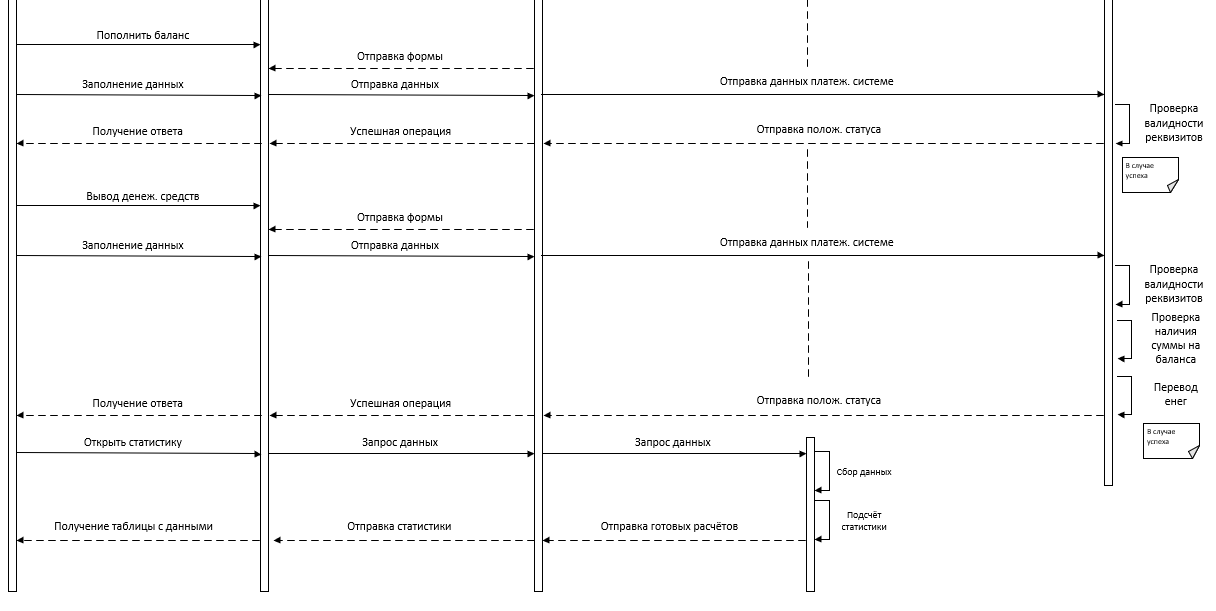


Рисунок 8 - Диаграмма последовательности (часть 2)

Этап 1: Регистрация 🡪 заполнение всех полей 🡪 отправка данных на сервер 🡪 валидация данных 🡪 сохранение в БД 🡪 отправка ответа на клиент.

Этап 2: Авторизация 🡪 заполнение всех полей 🡪 отправка данных на сервер 🡪 валидация данных 🡪 верификация введенных данных с данными в БД 🡪 отправка ответа на клиент.

Этап 3: Загрузка поста 🡪 заполнение полей контентом 🡪 отправка данных на сервер 🡪 сохранение поста в БД 🡪 отправка ответа на клиент.

Этап 4: Покупка подписки 🡪 выбор автора 🡪 выбор тарифа 🡪 выбор метода оплаты 🡪 оплата подписки 🡪 получение доступа к контенту.

Этап 5: Просмотр поста 🡪 выбор желаемого поста 🡪 ожидание открытия 🡪 просмотр поста

Этап 6: Добавление комментария 🡪 заполнение всех полей 🡪 загрузка комментария на сервер 🡪 получение обновленного списка комментариев с сервера 🡪 получение ответа на клиент.

Этап 7: Пополнение баланса 🡪 заполнение всех полей формы 🡪 отправка данных платёжной системе 🡪 обработка данных платёжной системой 🡪 получение ответа 🡪 в случае успеха, пополнение баланса на платформе.

Этап 8: Вывод баланса 🡪 заполнение полей формы 🡪 отправка запроса на сервер 🡪 отправка запроса с сервера в платёжную систему 🡪 получение статуса 🡪 в случае успеха, уменьшение баланса на платформе.

Этап 9: Просмотр статистики 🡪 запрос данных о подписках, подписчиках и тарифах 🡪 расчёт данных на сервере 🡪 отправка готовых расчётов на клиент 🡪 формирование данных в таблицу.

**2.6 Диаграмма базы данных**

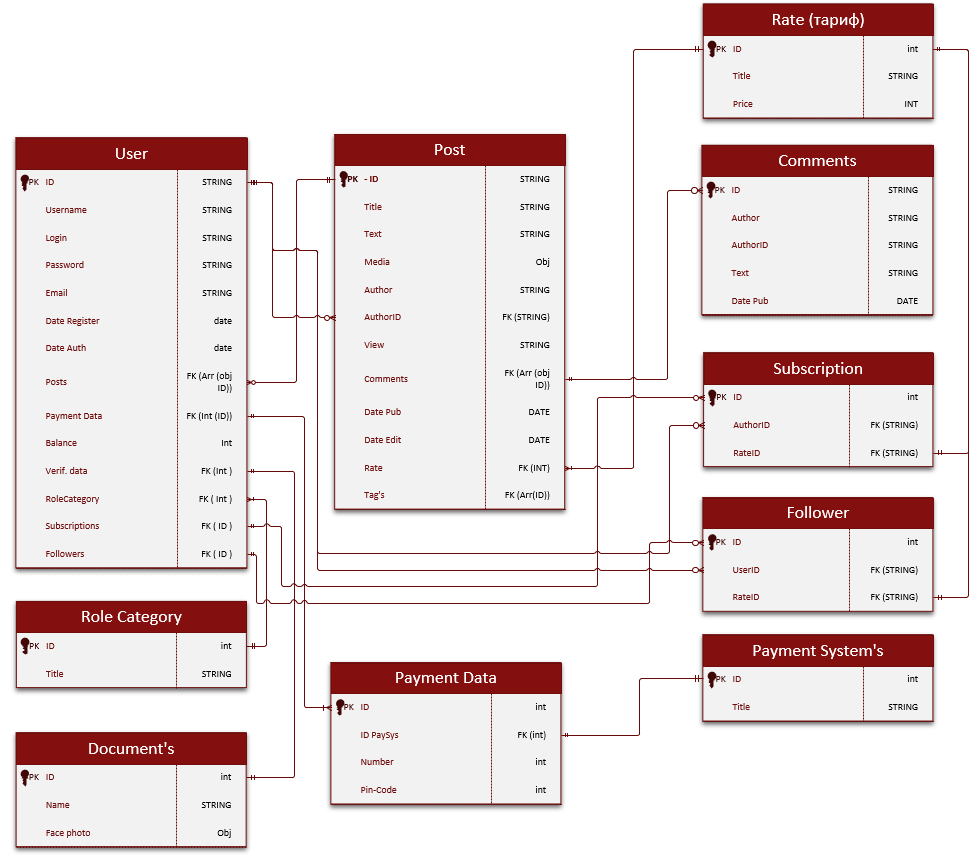


Рисунок 9 - Диаграмма базы данных

* Таблица User хранит в себе сущности класса User, в нее записываются все зарегистрированные профили пользователей OnlyPay.
* Таблица Post хранит в себе сущности класса Post, в нее записываются посты, которые добавляют пользователи в виде контента.
* Таблица Rate хранит в себе сущности класса Rate, в нее записываются все доступные для пользователя тарифы, по которым можно оформить подписку.
* Таблица Comment хранит в себе сущности класса Comment, в нее записываются все созданные комментарии пользователей.
* Таблица Subscription хранит в себе сущности класса Subscription, в нее записываются все оформленные подписки на платформе OnlyPay.
* Таблица Follower хранит в себе сущности класса Follower. В нее записываются все пользователи (подписчики), которые подписались хотя бы на одного автора и подписка, которая актуальна на текущий момент.
* Таблица Payment System хранит в себе сущности класса Payment System, в нее записываются все доступные платежные системы, с помощью которых можно пополнить баланс или вывести заработанные денежные средства.
* Таблица Payment Data хранит в себе сущности класса Payment Data, в нее записываются платежные реквизиты пользователя.
* Таблица RoleCategory хранит в себе сущности категорий, которые указал пользователь при регистрации на платформе OnlyPay.

**Глава 3 Документо-ориентированные работы**

**3.1 Руководство пользователя**

Для работы с платформой OnlyPay зарегистрируйтесь в системе.

1. **Регистрация**

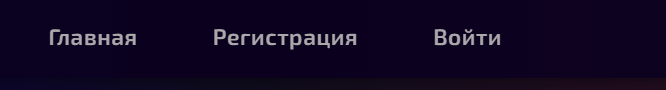


Рисунок 10 – Меню сверху (до авторизации)

* В верхнем меню нажмите на вторую по счёту кнопку «Регистрация».
* Заполните все необходимые поля и нажмите на кнопку «Зарегистрироваться».

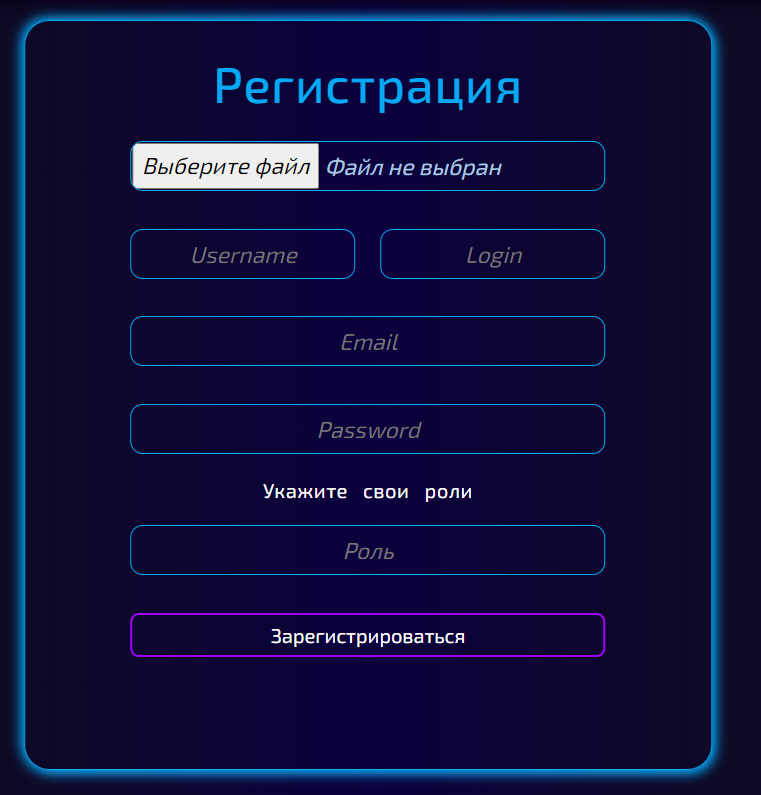


Рисунок 11 - Окно регистрации

1. **Авторизация**

* В верхнем меню нажмите на третью по счёту кнопку «Войти».

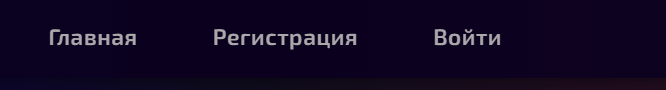


Рисунок 12 - Меню сверху (до авторизации)

* Введите логин и пароль, указанные при регистрации (необходимо соблюдать нижний и верхний регистры, а также пробелы).
* Нажмите на кнопку «Войти»

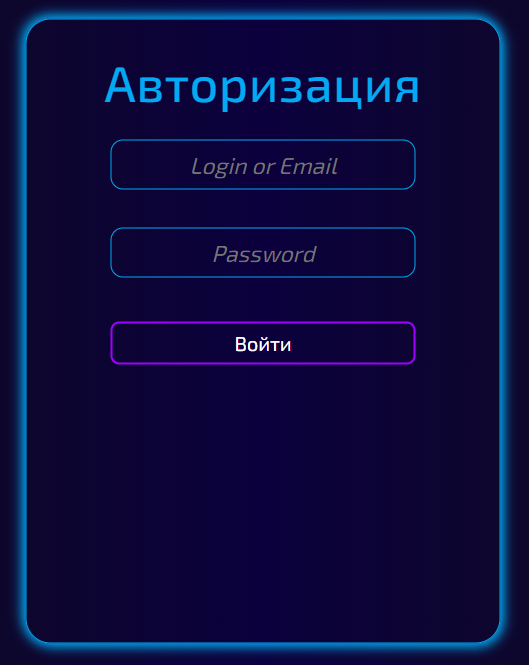


Рисунок 13 - Окно авторизации

1. **Просмотреть личный профиль**

* В верхнем меню нажмите на первую по счёту кнопку «Мой профиль».

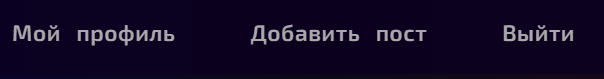


Рисунок 14 - Меню сверху (после авторизации)

* Откроется окно вашего профиля.
* В левой части профиля располагается иконка вашего профиля и меню управления вашим профилем.
* В правой части профиля располагается сноска с информацией о количестве подписчиков, подписок и постов, и все ваши посты.

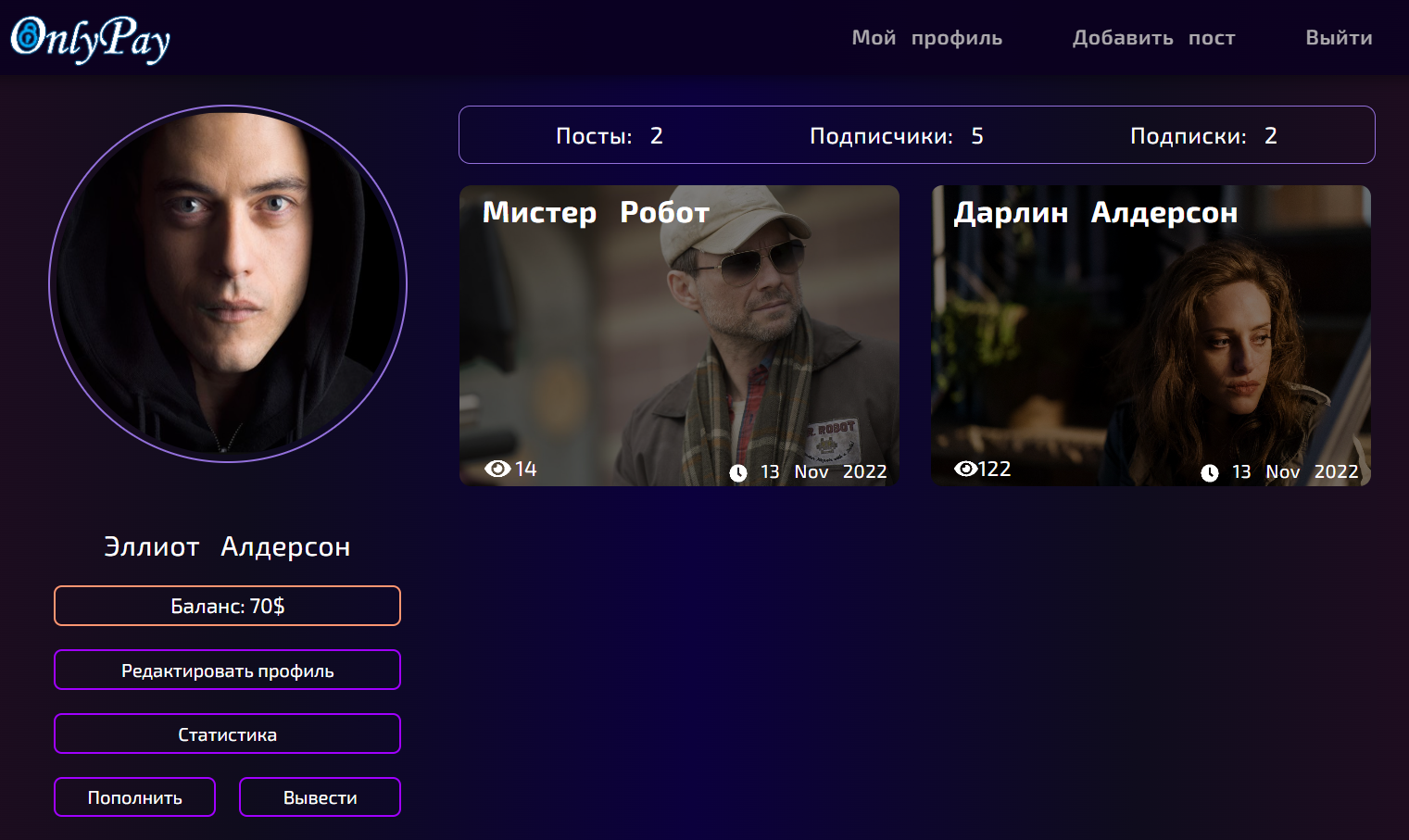


Рисунок 15 - Окно личного профиля

1. **Пополнить счёт**

* В левой части профиля, в меню управления профилем найти кнопку «Пополнить» и нажать на нее.
* В появившемся окне выбрать способ пополнения счёта, указать реквизиты и желаемую сумму для пополнения
* Нажать на кнопку «Пополнить»

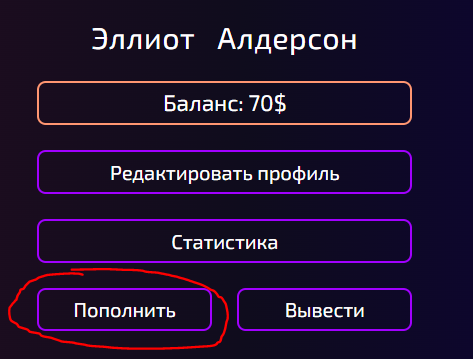


Рисунок 16 - Меню пополнения баланса

1. **Вывести прибыль**

* В левой части профиля, в меню управления профилем найти кнопку «Вывести» и нажать на нее.
* В появившемся окне выбрать способ вывода счёта, указать реквизиты и желаемую сумму для вывода средств.
* Нажать на кнопку «Вывести».

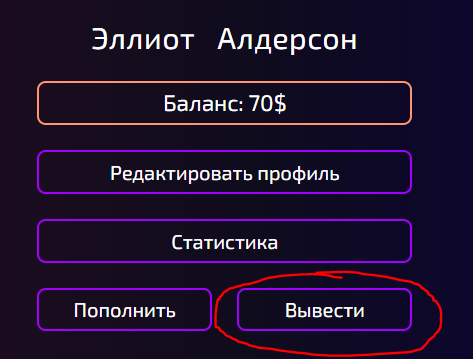


Рисунок 17 - Меню вывода денеж. средств

1. **Просмотреть статистику**

* В левой части профиля, в меню управления профилем найти кнопку «Статистика» и нажать на нее.
* В появившемся окне появится таблица со всеми подписками, оформленными на вас и их информацией.
* В случае отсутствия подписок на ваш профиль, будет выведено соответствующее сообщение.
* Ниже таблицы представлен расчёт вашей прибыли.

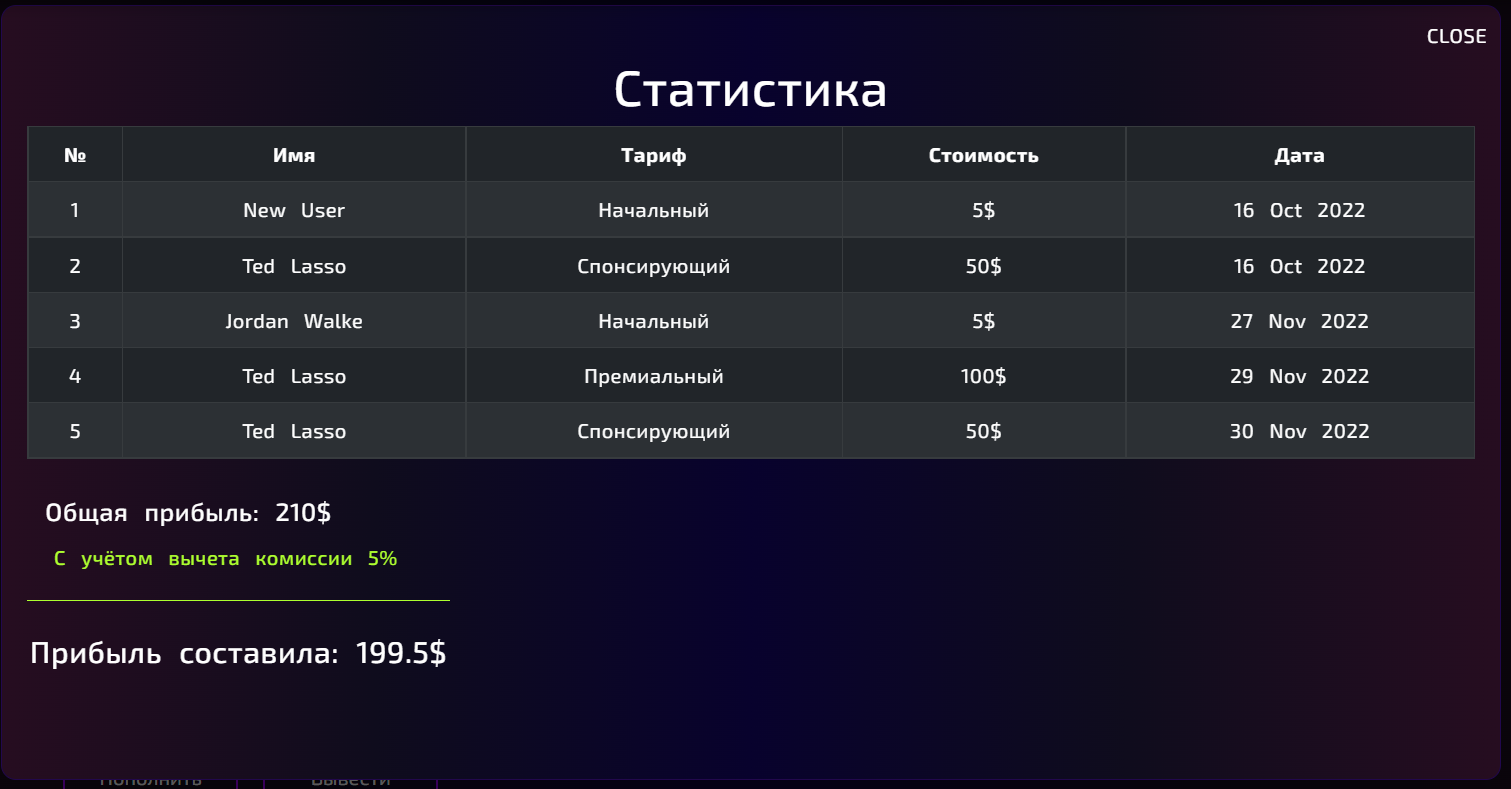


Рисунок 18 - Окно статистики

1. **Просмотреть профиль выбранного автора**

* В верхнем меню, в левой части нажмите на логотип OnlyPay, чтобы перейти на главную страницу.
* После приветствия на главной странице, ниже выведен список авторов, зарегистрированных на платформе OnlyPay.



Рисунок 19 - Список авторов на Главной странице

* Нажмите на иконку автора, чтобы открыть его профиль.

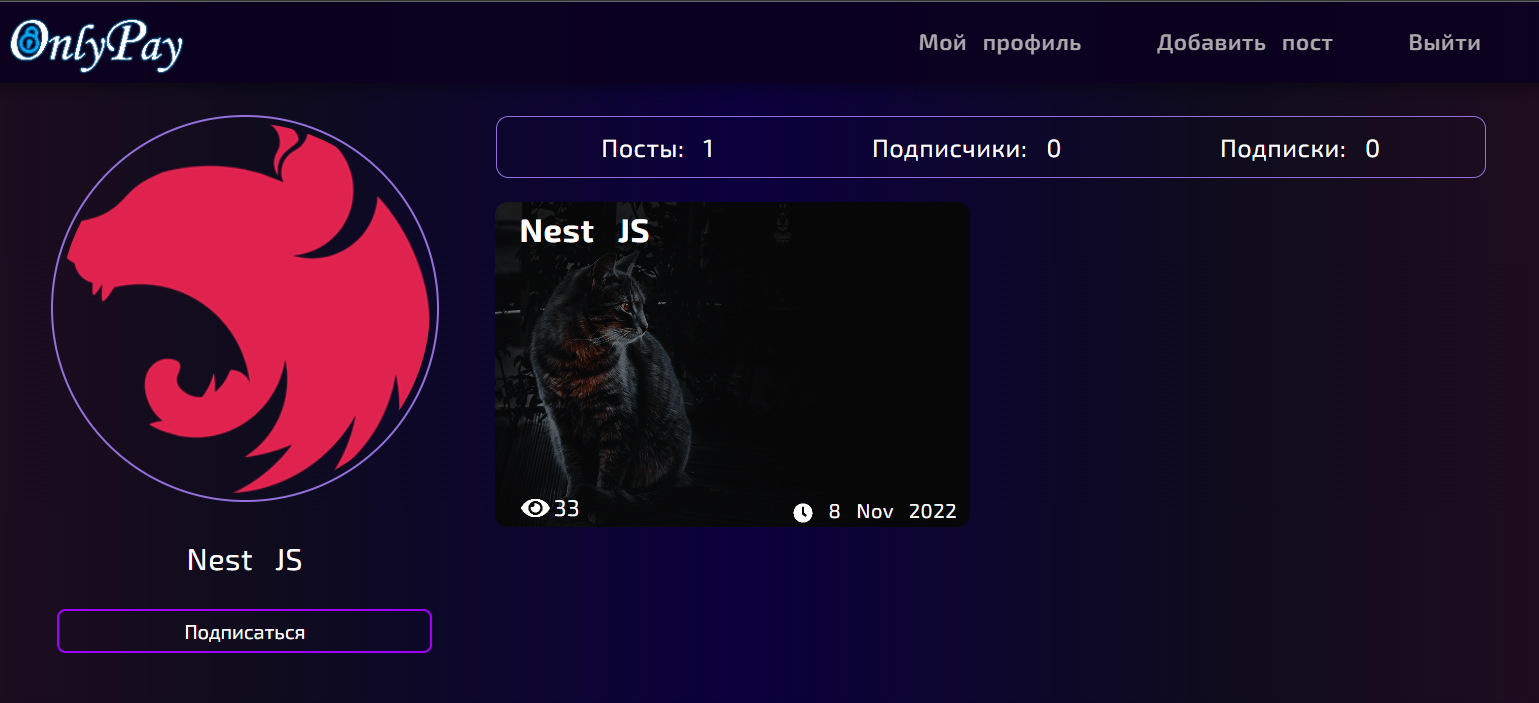


Рисунок 20 - Окно профиля автора

1. **Купить подписку на автора**

* После перехода на профиль автора, в левой части профиля, нажмите на кнопку «Подписаться».
* В появившемся окне выберите желаемый тариф.
* Выделите тариф, нажав на него и нажмите на кнопку «Подтвердить».
* Появится всплывающее уведомления о статусе операции (успешный или отказ).



Рисунок 21 - Окно оформления подписки

1. **Просмотреть чужой пост**

* В профиле автора или на главной странице наведите курсор на нужный пост, в области картинки поста появится надпись «Читать».
* Нажмите на надпись «Читать», после чего откроется пост.

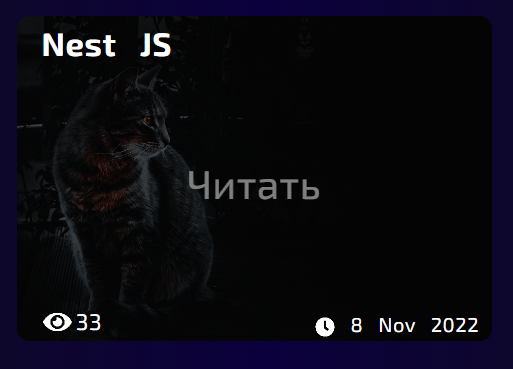


Рисунок 22 - Окно постов

1. **Просмотреть или добавить комментарий к чужому посту**

* На странице поста в левой части, в меню навигации, нажмите на самую нижнюю кнопку с иконкой сообщения.

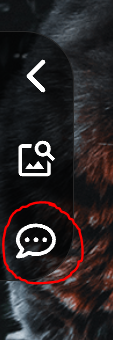


Рисунок 23 - Навигационное меню

* Откроется модальное окно с комментариями и нижним полем для написания нового комментария.
* Введите текст в нижнее поле и нажмите на кнопку справа «Отправить».

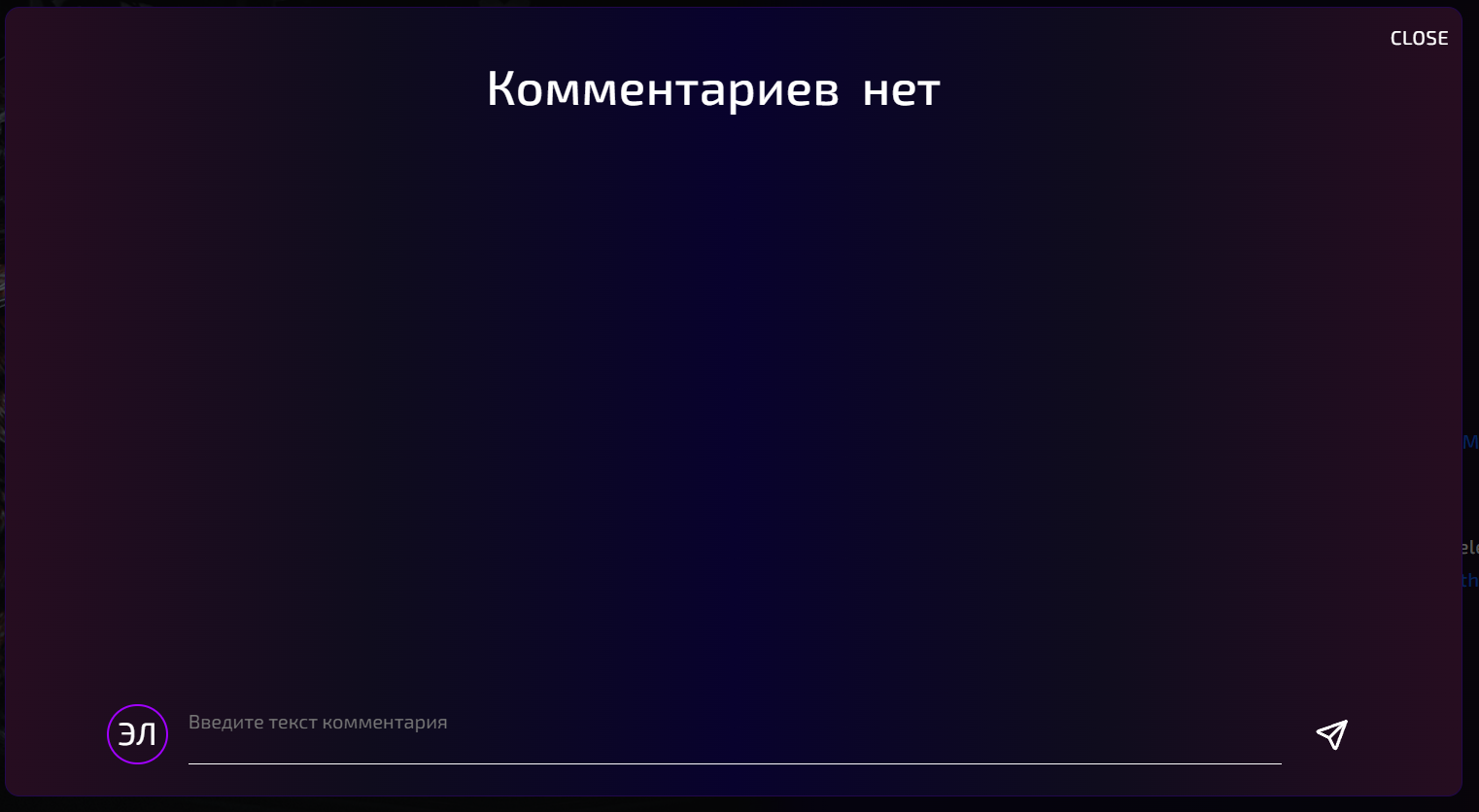


Рисунок 24 - Окно написания комментария

1. **Добавить пост**

* В верхнем меню нажмите на вторую по счёту кнопку «Добавить пост».

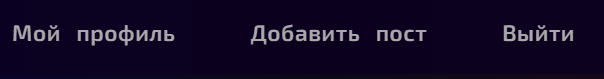


Рисунок 25 - Меню сверху

* Для загрузки картинки нажмите на кнопку «Выберите файл» и выберите картинку на вашем компьютере.
* Заполните все необходимые поля.
* Для очистки всех полей формы нажмите на кнопку «Очистить».
* Для публикации поста нажмите на кнопку «Добавить»

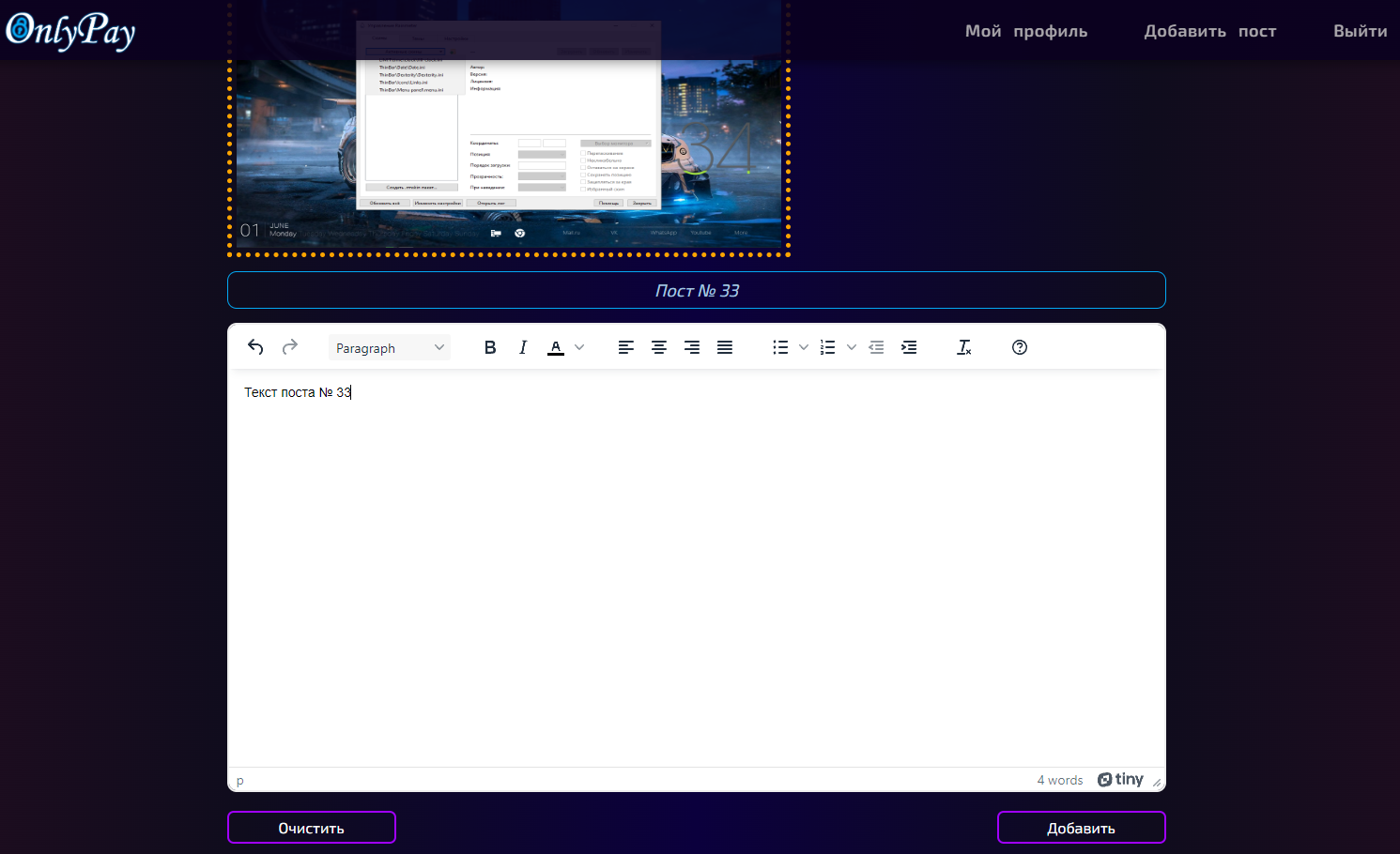


Рисунок 26 - Окно добавления нового поста

1. **Просмотреть свой пост**

* Перейдите в свой профиль.
* В правой части профиля в области списка постов выберите нужный пост, наведите на него курсов и нажмите надпись «Читать».

1. **Отредактировать свой пост**

* Откройте нужный пост
* На странице поста в левой части, в меню навигации, нажмите на центральную кнопку с иконкой карандаша.

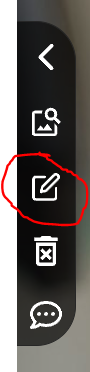


Рисунок 27 - Меню навигации

* В открывшейся странице внесите изменения.
* Для отмены операции нажмите на кнопку «Отмена»
* Для подтверждения операции редактирования нажмите на кнопку «Изменить».

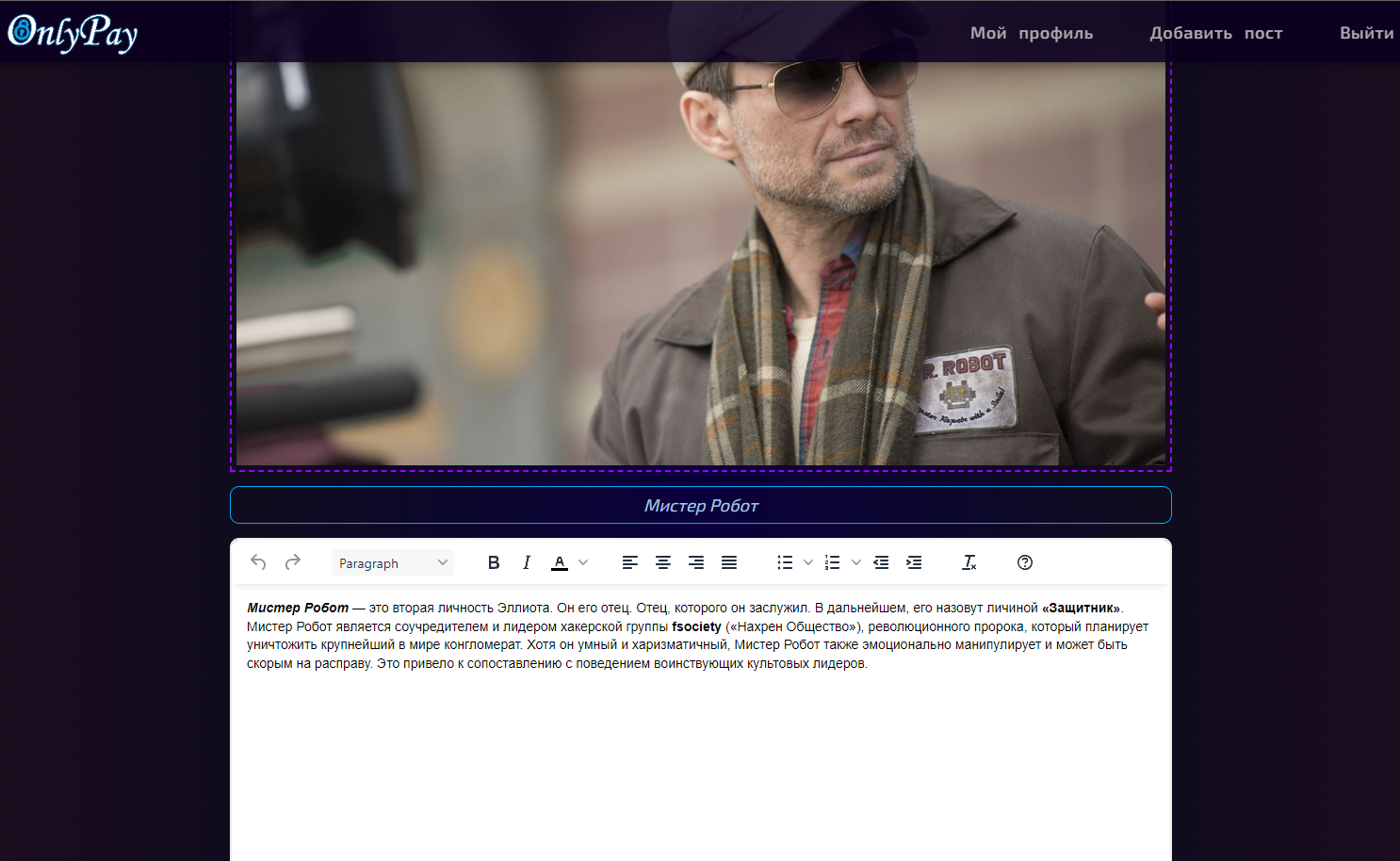


Рисунок 28 - Окно редактирования поста

**3.2 Руководство программиста**

**Front-End разработка**

1. Front-End часть построена на библиотеке React JS;
2. Язык разработки JavaScript;
3. Используется State Management Redux Toolkit;
4. Библиотека HTTP-запросов JavaScript Node.js Axios;
5. Роутинг на проекте реализован с помощью инструмента React-Router-Dom 6;
6. Отображение иконок – Ant Design Icons;
7. Текстовый редактор для публикации – Tiny MCE React
8. Отображение таблиц – React Bootstrap;
9. Язык разметки для форматирования текст – React-Markdown;
10. Для обработки и отображения дат – React-Moment;
11. Окна с отображением уведомлений – React-Toastify;
12. Стилизация компонентов – Styled Components.

**Back-End разработка**

1. Программная платформа Node JS;
2. Сервер развёрнут на Express JS;
3. Язык разработки JavaScript;
4. Автоматическая перезагрузка сервера Nodemon;
5. База данных Mongo DB;
6. ODM Mongoose;
7. Шифрование паролей – Bcrypt;
8. Cross Origin Resource Sharing – CORS;
9. Переменные среды - .ENV;
10. Загрузка файлов на Express – Express-FileUpload;
11. Формирование Веб-токена – JSONWebToken.

**Демонстрация кода**

Главная страница

const Main = () => {

const dispatch = useDispatch();

let username = useSelector((state) => state.authReducer.user || 'Гость' );

const arrPosts = useSelector(state => state.postReducer.posts);

const arrAuthors = useSelector(state => state.allAuthorsReducer.authors);

let status = useSelector(state => state.postReducer.status)

// let isAuth = useSelector(checkIsAuth);

React.useEffect(()=>{

dispatch(getAllPosts());

dispatch(getAllAuthors())

},[dispatch])

React.useEffect(()=>{

if (status)

{

toast(status);

dispatch(clearStatus());

}

},[status, dispatch])

return (

<MainTeg>

<LineWelcome>

{username} <div>, добро пожаловать на </div> <Welcome>OnlyPay</Welcome>

</LineWelcome>

<Shadow>Зарегистрироваться</Shadow>

<div>

{arrAuthors.length ?

<ContainerAuthors>

{arrAuthors.map((user)=>(

<UserItem key={user.\_id} {...user} />

))}

</ContainerAuthors>

:

<div>Будь первым!</div>

}

</div>

<div>

{arrPosts.length ?

<div>

Обычные посты

{arrPosts.map((post)=>(

<PostItem key={post.\_id} {...post} />

))}

</div> :

<div>Постов нет</div>

}

</div>

</MainTeg>

);

};

export default Main;

Контроллер Auth

import User from "../Models/User.js";

import bcrypt from "bcryptjs";

import jwt from "jsonwebtoken";

import path, {dirname} from "path";

import {fileURLToPath} from "url";

// ==================== Register ======================

export const register = async (req, res)=> {

try {

const {username,login,password, email, roleCategory, avatar} = req.body;

let isUser = await User.findOne({username});

if (isUser)

{

return res.json({message:`Пользователь ${username} уже зарегистрирован!`});

}

isUser = await User.findOne({login});

if (isUser)

{

return res.json({message:`Логин ${login} уже занят!`});

}

isUser = await User.findOne({email});

if (isUser)

{

return res.json({message:"Данный email уже зарегистрирован!"});

}

const salt = bcrypt.genSaltSync(10);

const hash = bcrypt.hashSync(password,salt);

let newUser;

let arrRoleCategory = roleCategory.split(",");

if(req.files)

{

let fileName = Date.now().toString() + req.files.avatar.name;

const \_\_dirname = dirname(fileURLToPath(import.meta.url));

await req.files.avatar.mv(path.join(\_\_dirname, "..", "uploads", fileName));

newUser = new User({

username,

login,

password:hash,

email,

avatar:fileName,

roleCategory:arrRoleCategory

});

}else

{

newUser = new User({

username,

login,

password:hash,

email,

roleCategory:arrRoleCategory

});

}

const token = jwt.sign(

{id:newUser.\_id},

process.env.JWT\_SECRET,

{expiresIn: "30d"}

);

await newUser.save();

res.json({username, avatar, token, message: "Регистрация прошла успешно!"});

}

catch (err)

{

res.json({message:"Ошибка регистрации!", error:`${err}`});

}

}

// ============= Login ==========================

export const login = async (req, res)=>{

try {

const {login, password} = req.body;

const isUser = await User.findOne({login});

if (!isUser)

{

return res.json({message:"Пользователь не найден!"})

}

const isPasswordCorrect = bcrypt.compareSync(password, isUser.password);

if (!isPasswordCorrect)

{

return res.json({message:"Пароль не верный!"});

}

const token = jwt.sign(

{id:isUser.\_id},

process.env.JWT\_SECRET,

{expiresIn: "30d"}

);

res.json({token,isUser, message:"Вы успешно авторизовались!"});

}

catch (err)

{

res.json({message:"Ошибка авторизации!"});

}

}

// ======================== getMe =========================

export const getMe = async (req, res)=>

{

try {

const isUser = await User.findById(req.userId)

if (!isUser)

{

return res.json({message:"Аккаунт не найден!"})

}

const token = jwt.sign(

{id: isUser.\_id},

process.env.JWT\_SECRET,

{expiresIn: "30d"}

)

return res.json({isUser, token})

}

catch (err){ return res.json({message:"Аккаунт не найден!"});}}

Метрики программного проекта

***Размерно-ориентированные метрики***

* Размерно-ориентированные метрики прямо измеряют программный продукт и процесс его разработки. Основываются размерно-ориентированные метрики на LOC-оценках (Lines Of Code). LOC-оценка — это количество строк в программном продукте.

*Достоинства размерно-ориентированных метрик:*

* 1) широко распространены;
* 2) просты и легко вычисляются.

*Недостатки размерно-ориентированных метрик:*

* 1) зависимы от языка программирования;
* 2) требуют исходных данных, которые трудно получить на начальной стадии проекта;
* 3) не приспособлены к непроцедурным языкам программирования.

***Функционально-ориентированные метрики***

* Функционально-ориентированные метрики косвенно измеряют программный продукт и процесс его разработки. Вместо подсчета LOC-оценки при этом рассматривается не размер, а функциональность или полезность продукта.
* Используется 5 информационных характеристик.

1. *Количество внешних вводов.* Подсчитываются все вводы пользователя, по которым поступают разные прикладные данные. Вводы должны быть отделены от запросов, которые подсчитываются отдельно.

2. *Количество внешних выводов.* Подсчитываются все выводы, по которым к пользователю поступают результаты, вычисленные программным приложением. В этом контексте выводы означают отчеты, экраны, распечатки, сообщения об ошибках. Индивидуальные единицы данных внутри отчета отдельно не подсчитываются.

3. *Количество внешних запросов.* Под запросом понимается диалоговый ввод, который приводит к немедленному программному ответу в форме диалогового вывода. При этом диалоговый ввод в приложении не сохраняется, а диалоговый вывод не требует выполнения вычислений. Подсчитываются все запросы — каждый учитывается отдельно.

4. *Количество внутренних логических файлов.* Подсчитываются все логические файлы (то есть логические группы данных, которые могут быть частью базы данных или отдельным файлом).

5. *Количество внешних интерфейсных файлов.* Подсчитываются все логические файлы из других приложений, на которые ссылается данное приложение.

После сбора всей необходимой информации приступают к расчету метрики — *количества функциональных указателей FP* (Function Points). Автором этой метрики является А. Албрехт (1979).

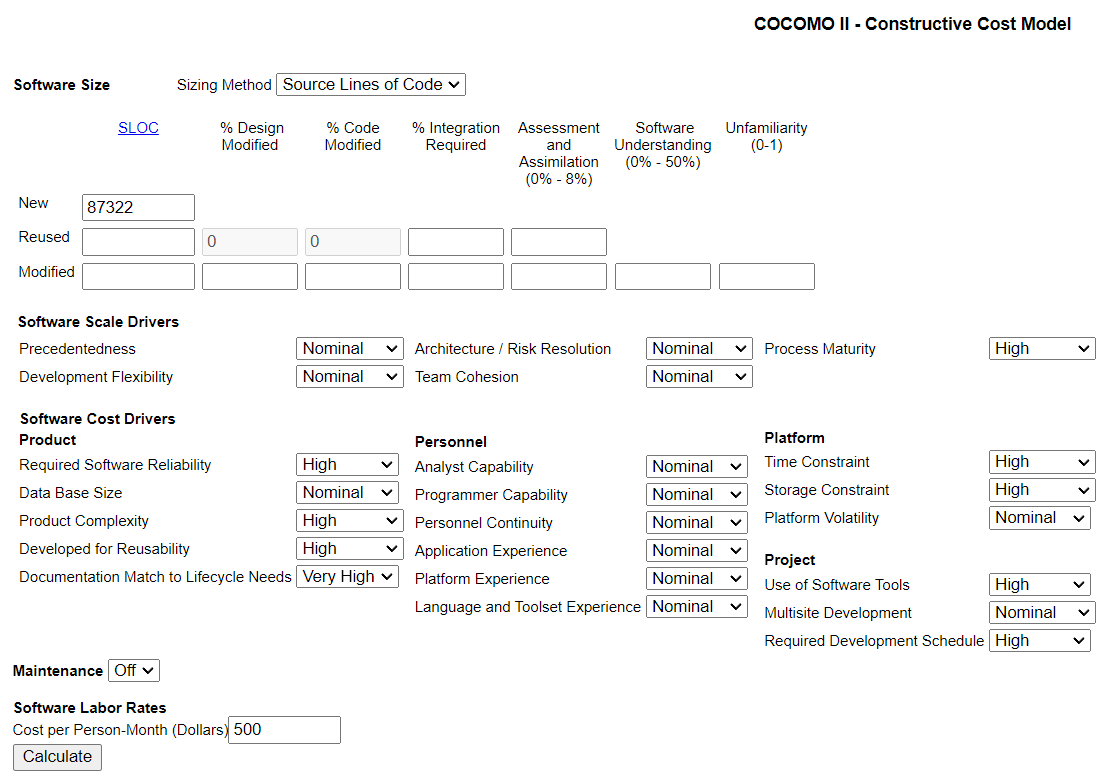
Онлайн расчёт проекта

Рисунок 29- Конструктивная модель затрат

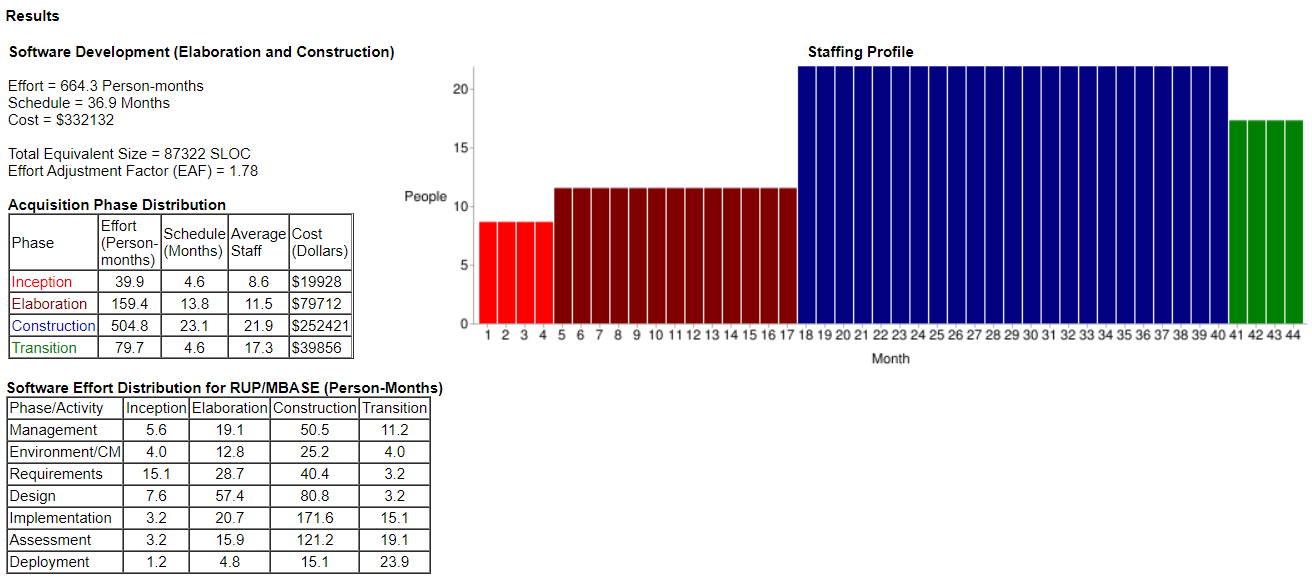


Рисунок 30 - Результат расчёта модели COCOMO

Для расчета стоимости программы был использован онлайн-ресурс COCOMO- Constructive Cost Model (конструктивная модель стоимости) – это способ оценки стоимости разработки программного обеспечения, разработанная Барри Боэмом.

Базовый уровень рассчитывает трудоемкость и стоимость разработки как функцию от размера программы. Размер выражается в оценочных тысячах строк кода KLoC - kilo lines of code.

Ссылка на сайт, на котором проводились расчеты:

<http://softwarecost.org/tools/COCOMO/>

В целом, COCOMO применим к трем классам проектов разработки ПО:

* Органический (Organic mode) – маленькие команды с хорошим опытом работы и не жесткими требованиями к разработке
* Полуразделённый вид (Intermediate / Semi-detached mode) – средние по размеру команды со смешанным опытом разработки и со смешанными требованиями (как жесткими, так и нет).
* Встроенный вид (Intered / Embedded mode) – разрабатываются с учетом множества жестких ограничений (по аппаратному, программному, операционному обеспечению и т.д.)

*Разрабатываемая мною система относится к органическому классу проектов разработки ПО, так как разработка проводится в команде из 1 человек, и в требованиях к разработке нет жестких ограничений.*

Итак, всего было написано 87322 строки кода

В качестве метода определения размера программного обеспечения был выбран: «Исходные строки кода».

Далее были определены уровни сложности продукта, языка и набора инструментов, ограничения времени, сплоченности команды, требуемой надежности программного обеспечения и многих других показателей. Критерии изображены на рисунке №27.

Результаты произведенных вычислений:

* Затраты на разработку программного обеспечения (разработка и конструирование)= 664,3 человеко-месяцев
* График = 36,9 месяцев
* **Стоимость = 333 000 долларов США**
* Общий эквивалентный размер = 87322 SLOC
* Коэффициент корректировки усилий (EAF) = 1,78

**Заключение**

В результате разработки была создана платформа для публикации платного контента, в которой были реализованы следующие функции:

* Регистрация пользователя;
* Авторизация пользователя;
* Пополнение баланса и вывод прибыли;
* Публикация платного контента;
* Покупка подписки и просмотр платного контента;
* Слежение за статистикой актуальности своего контента;
* Давать субъективную оценку содержимого контента в виде комментария под любым постом;
* Зарабатывать на своем творчестве в лице автора и получать эксклюзивные информационные (цифровые) материалы в лице подписчика.

В ходе разработки программного продукта были исследованы и закреплены следующие инструментальные средства:

* Было освоено два направления Front-End и Back-End в лице React JS и Node JS, Express JS;
* Передаче данных между клиентом и сервером:
* Подключение и хранение в документо-ориентированной базе данных Mongo DB;
* Освоение CRUD операций в стеке MERN;
* Организация регистрации и авторизации с использованием Веб-токенов;
* Обработка возможных ошибок как со стороны пользователя, так и со стороны сервера.

Так как платформа OnlyPay содержит в себе функции пополнения и вывода денежных средств, необходим внешний платежный сервис такие как:

Visa, MasterCard, Demir Bank, Элкарт, ЭлСом, РСК Банк, MBANK, Optima Bank.

В результате расчёта стоимости разработки программного продукта OnlyPay с помощью «Модели конструктивной стоимости COCOMO» была высчитана стоимость в размере 333 000$.