

Министерство образования и науки
Российской Федерации
Федеральное государственное
автономное образовательное учреждение
высшего образования
«Уральский федеральный университет
имени первого Президента России Б. Н.
Ельцина»

Институт радиоэлектроники и информационных технологий – РтФ Департамент информационных
технологий и автоматики
Школа бакалавриата

Итоговая работа
«ToDo List DApp (Decentralized Application)»

Выполнила:
Аббасов Руслан РИ-410913
Преподаватель:
Саиф М.А.

Введение

ToDo List DApp (Decentralized Application)

Цели и задачи проекта:

Цель проекта заключается в разработке децентрализованного приложения для управления задачами на основе блокчейна. В рамках этого проекта пользователи смогут создавать задачи, изменять их статус и удалять их при необходимости. Основу системы составит Ethereum, что обеспечит высокую степень надежности и прозрачности благодаря смарт-контрактам. Наша платформа предложит интуитивно понятный веб-интерфейс для легкого учета и управления задачами.

Краткое описание внесенных изменений:

Первоначально проект был разработан на основе scaffold-eth, который предлагает основную архитектуру для разработки смарт-контрактов и фронтенда. В ходе работы мы внесли ряд улучшений и модификаций.

1. Контракт:

- Мы разработали смарт-контракт ToDoList, который предоставляет функционал для создания задач, изменения их статуса (выполнена/не выполнена) и удаления всех задач.
- Реализованы события, которые уведомляют пользователей о создании задачи, изменении ее статуса и удалении всех задач.
- Контракт включает структуру Task, в которой хранятся идентификатор задачи, ее описание и статус выполнения.
- Смарт-контракт был успешно развернут в сети Ethereum с помощью Hardhat и соответствующего скрипта для деплоя.

2. Фронтенд:

- Создан интерфейс на основе React с интеграцией библиотеки ethers.js для взаимодействия с смарт-контрактом GWEI.
- Реализованы возможности для добавления новых задач, изменения их статуса и удаления всех задач, с автоматическим обновлением пользовательского интерфейса после каждой операции.
- Интерфейс предлагает пользователю возможность подключаться через Metamask и управлять своими задачами на блокчейне.

3. Тестирование:

- Для проверки функциональности смарт-контракта были разработаны юнит-тесты с использованием Hardhat и Chai. Эти тесты охватывают важные операции, такие как создание задач, изменение их статуса и удаление задач.
- Каждая функция смарт-контракта была протестирована на предмет правильного выполнения, а также на генерацию ожидаемых событий.

Смарт-контракт:

Описание изменений в YourContract.sol:

- Контракт был обновлен для реализации ключевых функций управления задачами, включая создание новых задач, изменение статуса выполнения и возможность удаления всех задач.
- Введена структура Task, которая включает в себя:
 - id: уникальный идентификатор задачи,
 - content: текстовое описание задачи,

- completed: статус выполнения задачи.
- Контракт теперь поддерживает следующие функции:
 - createTask: для создания новой задачи,
 - getTask: позволяет получить информацию о задаче по ее уникальному идентификатору,
 - toggleCompleted: используется для изменения статуса задачи на выполненную или невыполненную,
 - clearTasks: служит для удаления всех задач из системы.

Необходимость:

- Данные изменения предоставляют полное управление задачами посредством смарт-контракта и обеспечивают возможность интеграции с фронтенд-частью приложения. Это позволяет пользователям эффективно взаимодействовать с системой, управляя своими задачами без необходимости обращения к центральным серверам, что увеличивает безопасность и прозрачность процесса.

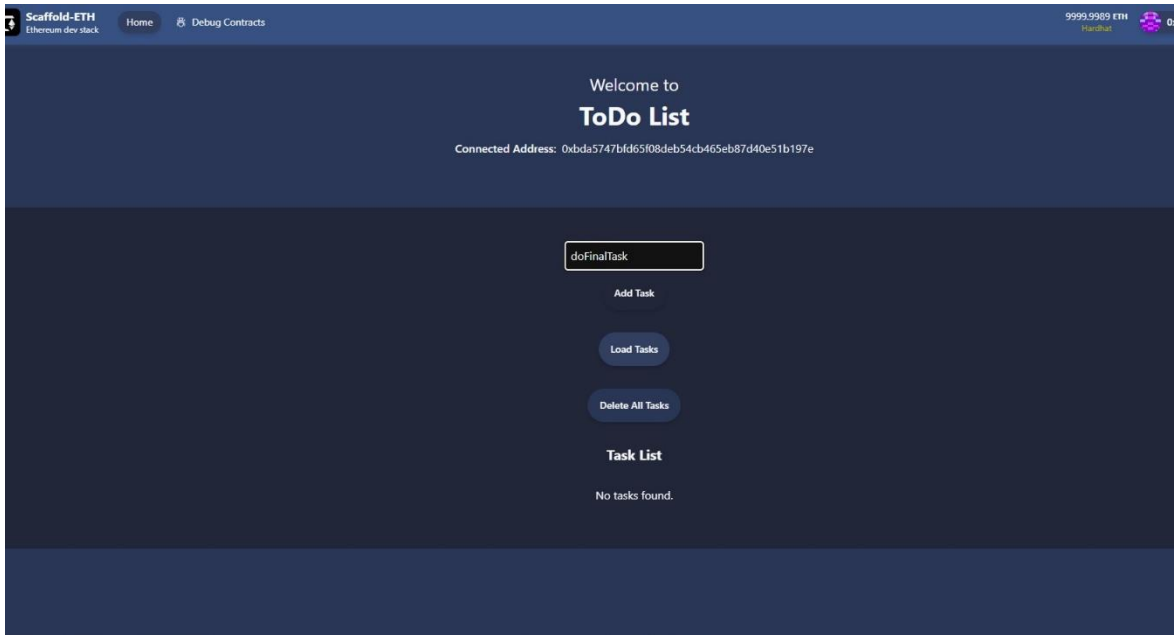
Фронтенд:

Описание изменений:

- Фронтенд приложения был разработан с использованием React и библиотеки ethers.js для взаимодействия с GWEi- смарт-контрактом, что обеспечило современный и отзывчивый пользовательский интерфейс.
- Пользователям доступны следующие функции:
 - Подключение своего кошелька через Metamask, что позволяет безопасно управлять задачами непосредственно из их криптовалютного кошелька,
 - Возможность добавления новых задач, что упрощает процесс управления задачами на децентрализованной платформе,
 - Функция переключения статуса задач между выполнено и не выполнено, дающая пользователям гибкость в управлении своим списком дел,
 - Опция удаления всех задач, что позволяет быстро очищать список задач по мере необходимости.
- Важным дополнением является реализация функции обновления интерфейса после каждой транзакции, что обеспечивает актуальную информацию для пользователей и улучшает опыт взаимодействия с приложением. Это позволяет пользователям видеть изменения в реальном времени, делая интерфейс более интерактивным и удобным для использования.

Основные взаимодействия интерфейса с контрактом:

- В интерфейсе фронтенда пользователи могут вводить текст задачи в специальное поле ввода. После ввода необходимого текста, нажатие кнопки "Add Task" вызывает функцию createTask контракта, инициируя процесс создания новой задачи.
- Данная функциональность обеспечивает интуитивно понятный и простой пользовательский опыт, позволяя пользователям без лишних хлопот добавлять задачи в их список дел. Благодаря интеграции с смарт-контрактом, введенные задачи сохраняются децентрализованно и надежно, что устраняет необходимость в централизованном хранении данных.
- После успешного создания задачи, фронтенд автоматически обновляет интерфейс, чтобы отобразить новую задачу в списке текущих задач, что в свою очередь повышает пользовательскую удовлетворённость и делает взаимодействие более плавным и приятным.






Account 4
hardhat




Запрос транзакции

Запрос от ?

 HTTP localhost:30

Взаимодействие с ?

 0x5FbDB...80ac

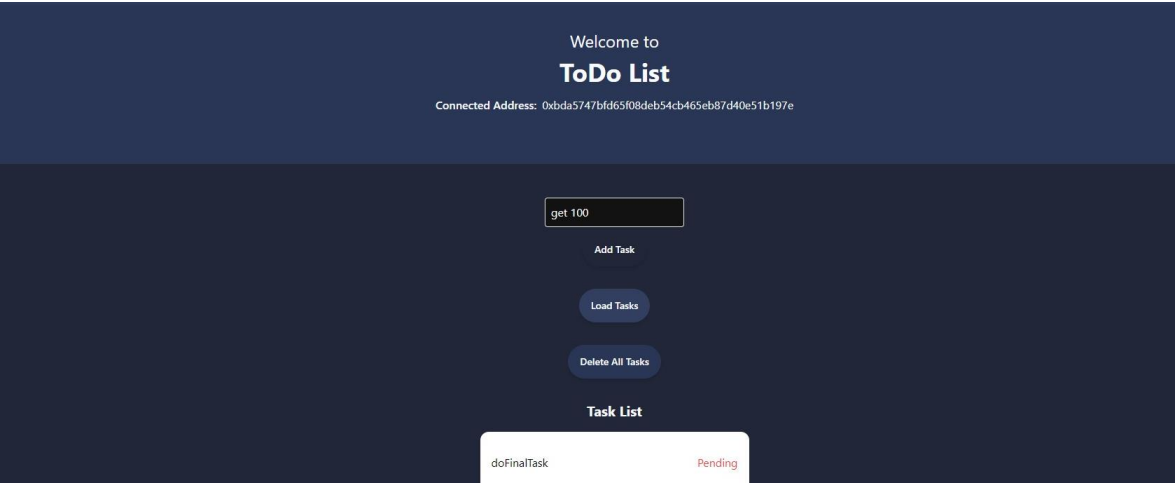
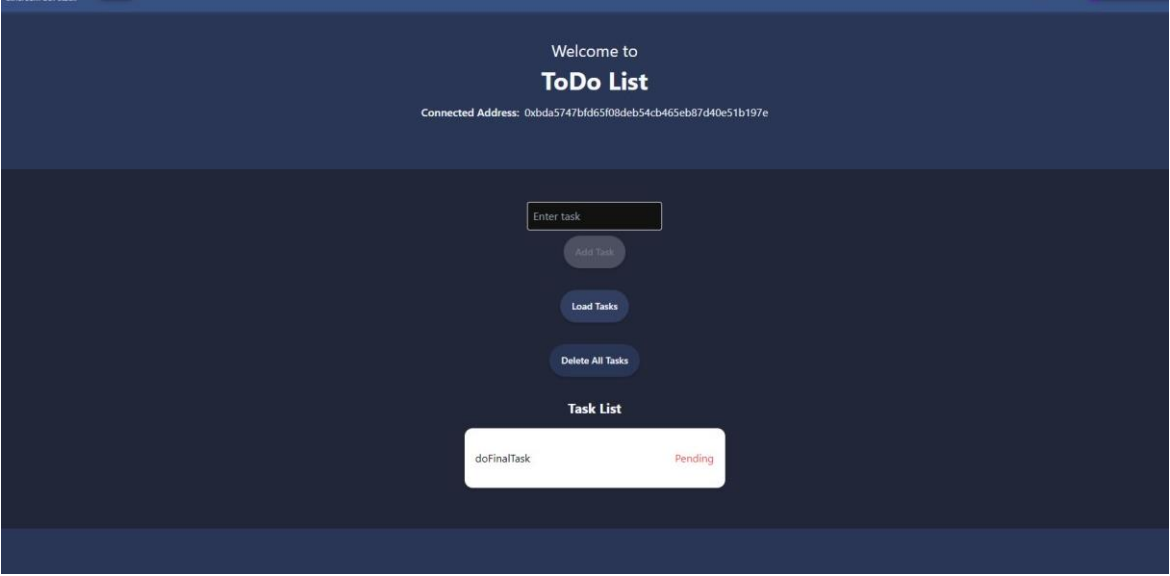
Network fee ?

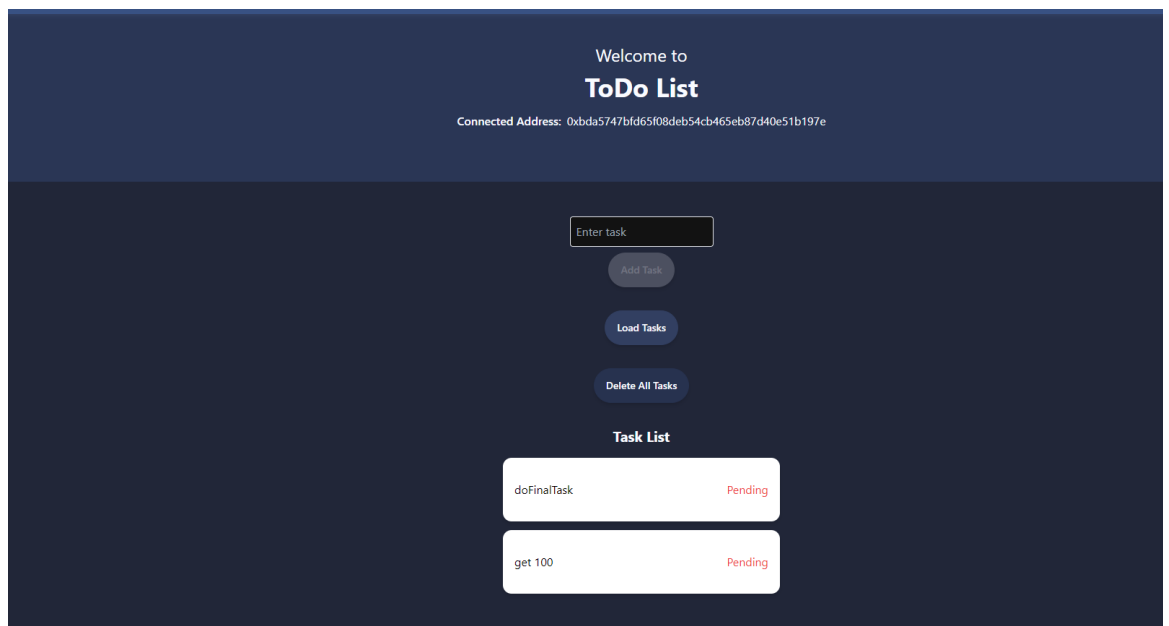
0.0001 gwei 0,00 \$

Скорость

Отмена

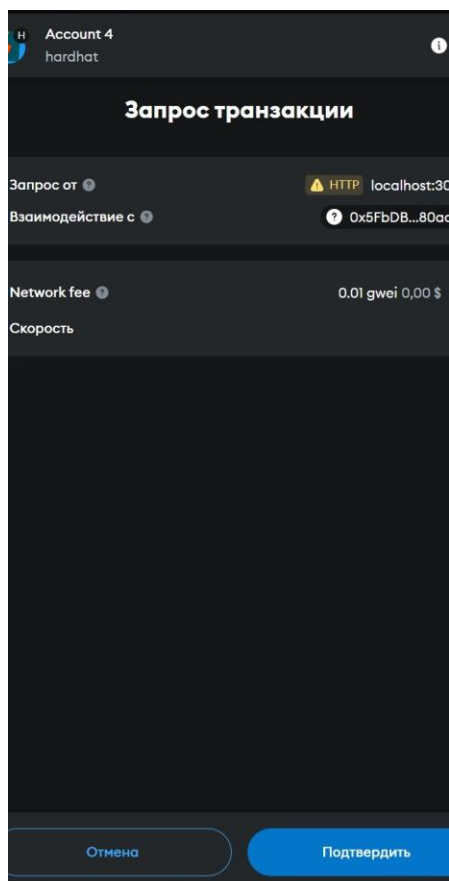
Подтвердить

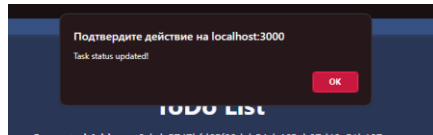




- Для переключения статуса задачи пользователь кликает на ее статус (Pending/Completed), что вызывает функцию toggleCompleted.

Запрос на переключение статуса

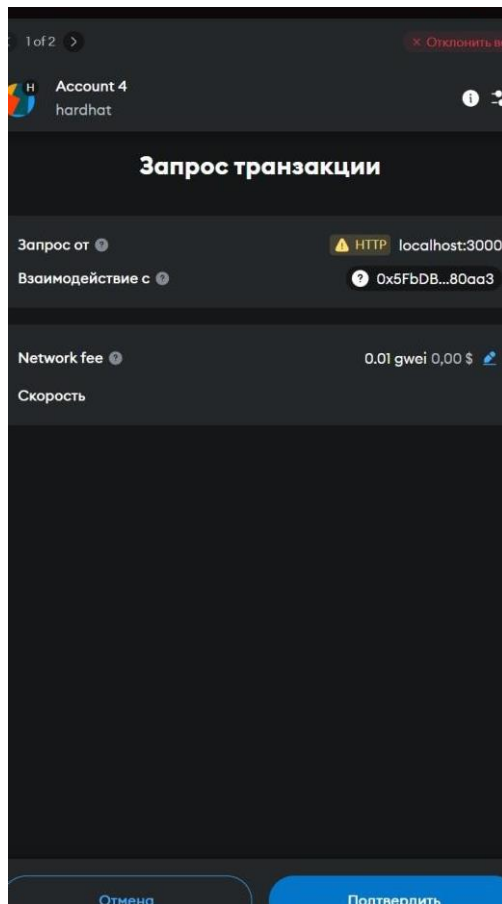


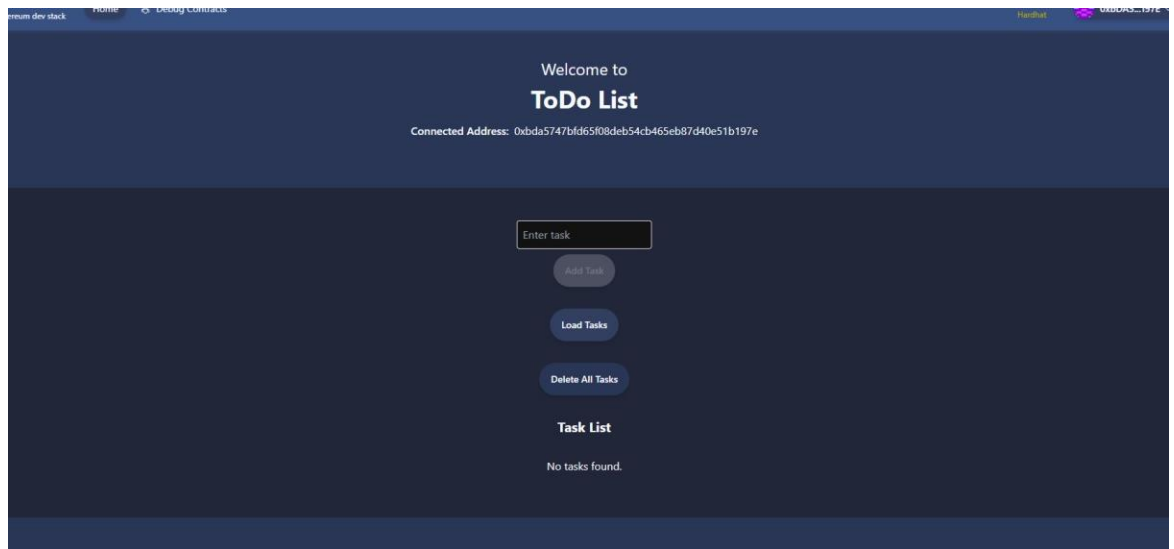


- Для удаления всех задач используется кнопка **Delete All Tasks**, которая вызывает функцию `clearTasks` контракта. Запрос на

удаление всех задач

-





- Интерфейс обновляется после каждого действия, чтобы отразить изменения в блокчейне

Скрипт размещения (Deploy Script):

Код обновленного скрипта размещения:

```
import { HardhatRuntimeEnvironment } from "hardhat/types";
import { DeployFunction } from "hardhat-deploy/types";
import { Contract } from "ethers";

const deployToDoList: DeployFunction = async function (hre:
HardhatRuntimeEnvironment) {
  const { deployer } = await hre.getNamedAccounts();
  const { deploy } = hre.deployments;

  await deploy("ToDoList", {
    from: deployer,
    args: [],
    log: true,
    autoMine: true,
  });

  const toDoList = await hre.ethers.getContract<Contract>("ToDoList", deployer);
  console.log("✅ ToDoList контракт развернут:", toDoList.address);
};

export default deployToDoList;
```

Описание ключевых изменений:

- Разработанный скрипт автоматизирует процесс развертывания контракта ToDoList как в локальной сети, так и в публичной сети GWEI. Это значительно упрощает задачу для разработчиков и позволяет сосредоточиться на основной логике приложения, минуя рутинные этапы.
- В скрипте реализован механизм деплоя с настройками autoMine для локальных сетей. Это означает, что транзакции, связанные с развертыванием контракта, будут автоматически подтверждены, что ускоряет процесс и устраняет необходимость вручную подтверждать каждую транзакцию. Данная функция особенно полезна при тестировании и отладке, так как сокращает время, затрачиваемое на развертывание.
- Контракт деплоится с указанием deployer как адреса для выполнения развертывания. Это гарантирует, что все транзакции, связанные с созданием и управлением контрактом, будут выполняться от заранее определённого адреса, обеспечивая высокий уровень контроля и безопасности в процессе развертывания.

Тестирование:

Список тестов с кратким описанием их назначения:

1. **Deployment Test:**
 - Цель: Проверяет, что контракт развернут с пустым списком задач. Этот тест гарантирует, что после развертывания контракта нет никаких предустановленных задач, что соответствует ожидаемому состоянию нового интерфейса.
2. **Create Task Test:**
 - Цель: Проверяет возможность создания новой задачи с правильным идентификатором (ID), контентом и статусом. Это гарантирует, что функция создания задачи работает корректно и каждая добавленная задача получает уникальный ID, а также правильно сохраняет введенный текст и статус.
3. **Toggle Task Completion Test:**
 - Цель: Проверяет возможность переключения статуса задачи с "не выполнена" на "выполнена" и обратно. Этот тест необходим для проверки правильности изменения статуса, что является ключевым функциональным требованием приложения.
4. **Clear Tasks Test:**

Цель: Проверяет удаление всех задач и отсутствие задач после выполнения функции clearTasks. Этот тест подтверждает, что данная функция работает должным образом, и после её вызова все задачи действительно удаляются.
5. **Event Emission Tests:**
 - Цель: Проверяет, что соответствующие события генерируются при создании задачи, изменении статуса и удалении задач. Этот тест важен для отслеживания изменений в состоянии контракта и подтверждения того, что приложение реагирует на изменения, уведомляя пользователей о выполненных действиях.

Результаты выполнения тестов (Скриншоты):

ToDoList					
Deployment					
✓ Should deploy with an empty task list					
Task Management					
✓ Should create a new task					
✓ Should toggle task completion					
✓ Should clear all tasks					
✓ Should emit TaskCreated event when a task is created					
✓ Should emit TaskCompleted event when a task is completed					
✓ Should emit AllTasksCleared event when all tasks are cleared					
7 passing (297ms)					
Solidity and Network Configuration					
Solidity: 0.8.20 · Optim: true · Runs: 200 · viaIR: false · Block: 30,000,000 gas					
Methods					
Contracts / Methods	Min	Max	Avg	# calls	gas (avg)
ToDoList					
clearTasks	52,181	54,421	53,674	3	-
createTask	76,792	94,000	84,201	7	-
toggleCompleted	23,135	45,035	39,560	4	-
Deployments					
					% of limit
ToDoList	-	-	461,181	1.5 %	-
Key					
0 Execution gas for this method does not include intrinsic gas overhead					

Выводы:

Краткий анализ работы:

- Разработанное децентрализованное приложение (DApp) для управления задачами на основе GWEI, использующее смарт-контракт, продемонстрировало успешную и корректную работу. Все ключевые функции, такие как создание задач, переключение статуса и удаление задач, функционируют без сбоев, что свидетельствует о высоком качестве реализации.
- Интерфейс приложения предоставляет пользователю возможность видеть изменения в реальном времени, что значительно улучшает пользовательский опыт. Эта плавная интеграция между смарт-контрактом и фронтенд-частью приложения позволяет пользователям быстро и удобно взаимодействовать с системой.
- Проведенные тесты подтвердили, что смарт-контракт работает корректно, генерируя правильные события в ответ на изменения в состоянии задач. Все функции успешно выполняются без ошибок, что минимизирует риск возникновения проблем в эксплуатации приложения. Применение комплексных тестов также способствует уверенности в надежности и безопасности приложения.

Возможные улучшения и следующий шаг:

- Добавление функциональности для редактирования задач: - В текущей версии приложения пользователи могут только создавать, завершать и удалять задачи. Введение возможности редактирования текста задачи позволит пользователям более гибко управлять своими задачами и сократит количество ошибок, связанных с необходимостью повторного создания задач при изменении их содержания.
- Оптимизация фронтенда для более быстрого отображения данных: - Для повышения

пользовательского опыта следует провести работу по оптимизации фронтенд-части приложения. Это может включать использование более эффективных подходов к загрузке и отображению данных, таких как применение кеширования, Lazy Loading и оптимизация работы с состоянием приложения. Быстрое отображение информации будет способствовать улучшению отзывчивости интерфейса и удовлетворению пользователей.

- Интеграция с другими смарт-контрактами и использование в разных сетях: - Следующий шаг — это исследование возможностей интеграции с другими смарт-контрактами, что может значительно расширить функционал приложения. Например, можно добавить возможность взаимодействия с контрактами для управления токенами или другими DApps. Также стоит рассмотреть возможность развертывания приложения в тестовых и основных сетях, таких как Rinkeby или Mainnet, что увеличит доступность и даст возможность большему числу пользователей воспользоваться разработкой.

Приложение

<https://github.com/RuslanAbbasov/FinalTask.git>