ФЕДЕРАЛЬНОЕ АГЕНТСТВО СВЯЗИ

Федеральное государственное образовательное бюджетное учреждение высшего профессионального образования

«Санкт-Петербургский государственный университет телекоммуникаций

им. проф. М. А. Бонч-Бруевича»

ОТЧЁТ

О ПРОДЕЛАННОЙ РАБОТЕ НА ФАКУЛЬТАТИВЕ

«Блокчейн: от теории к практике»:

Создание интернет-магазина со смарт-контрактом Escrow на базе платформы Ethereum

Научный руководитель: ассистент кафедры ПИВТ Помогалова А. В.

Выполнили: Громов А.А., Бакатов В.Н.,

Огорельцев П.А., Мурашкин Н.А.,

Храмов В.В.

Санкт-Петербург

2020 г

ВВЕДЕНИЕ

Интернет-магазин со смарт-контрактом Escrow представляет из себя сайт для приобретения смартфонов производителя Apple с усовершенствованной, клиенто-ориентированной системой платежа. Благодаря смарт-контракту Escrow, который выступает в качестве гаранта при переводе средств с адреса покупателя на адрес продавца, исключается любая возможность обмана и несвоевременной доставки приобретенного продукта.

ОБЗОР СИСТЕМЫ

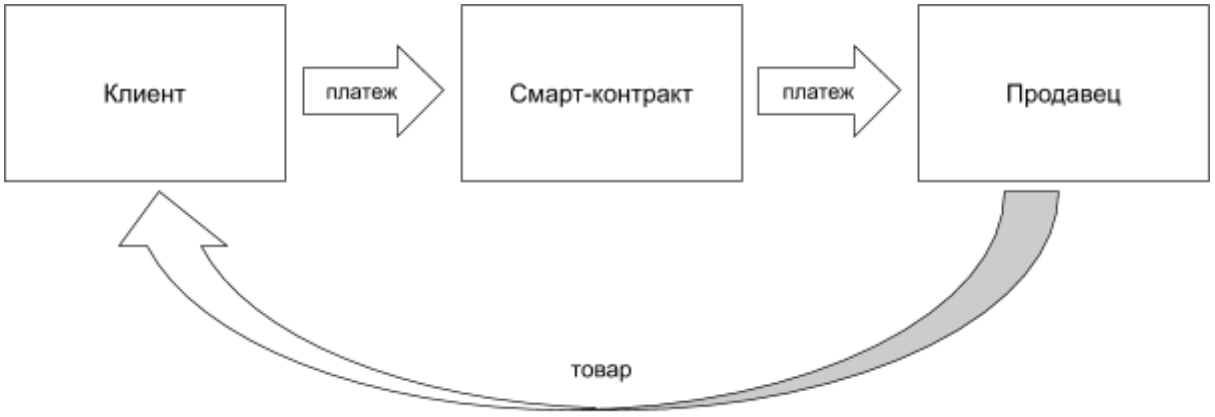

рис.1 Алгоритм работы системы при заказе

ОБЩИЕ СВЕДЕНИЯ И ОСНОВНЫЕ СВОЙСТВА

Принцип работы данного магазина основан на определенном порядке действий:

1. Покупатель делает заказ на сайте
2. Им производится оплата выбранного товара (транзакция осуществляется на адрес медиатора, за счет смарт-контракта Escrow, написанного на контрактно-ориентированном языке программирования Solidity)
3. Продавец получает уведомление о заказе покупателя, собирает заказ и отправляет его вместе с доверенным лицом(курьером)
4. Курьер доставляет товар и производит отметку об успешном выполнении заказа
5. Средства переводятся на счет продавца

В случае, если товар не доставлен в течение определенного срока доставки (2 месяца), покупатель может заявить об этом в своем Личном кабинете, после чего продавец вернет сумму в полном объеме и аннулирует заказ, либо же отдаст половину суммы, при этом ускорив получение товара покупателем.

Для случаев если покупателем был неправильно сделан заказ, и он желает отменить доставку, то он может перейти в Личный кабинет и нажать на кнопку «Отменить заказ». Продавец вернет всю сумму оплаты товара и отменит заказ.

РЕАЛИЗАЦИЯ ПРОЕКТА

В связи с малым количеством подобных решений для малого, среднего и крупного бизнеса, задача требовала определенных усилий для ее успешной реализации.

В ходе работы у нас возникли некоторые трудности в виде специфичности языка Solidity (о котором уже говорилось ранее), малого количества времени и большого размаха для творчества, однако справившись со всеми перечисленными обстоятельствами, нами были выполнены все части реализации данного проекта.

рис.2 Схема работы технической стороны проекта

Архитектура сайта написана на языке разметки HTML с использованием CSS, за аппаратную и интерактивную часть сайта отвечает язык программирования JavaScript.

Смарт-контракт сделан на Solidity версии 0.5.3, в самом контракте прописаны всевозможные варианты транзакций, учитываемые даже при невозможности доставить заказ - отмене, или же задержке заказа.

Серверная часть нашего проекта была написана на языке программирования Python, с помощью фреймворка Django, позволяющего развернуть сервер даже на микрокомпьютере Raspberry Pi, с относительно небольшой вычислительной мощностью процессора.

ПРИМЕНЕНИЕ ТЕХНОЛОГИЙ

Front-end

Front-end - это клиентская сторона пользовательского интерфейса к программно-аппаратной части сервера, то есть наш сайт, он написан с помощью HTML, CSS и JavaScript, где HTML отвечает за разметку сайта, CSS — за его стили, JavaScript является связующим элементом между смарт-контрактом и серверной частью.

Для связи смарт-контракта была использована библиотека web3.js — API для JavaScript, которая позволяет взаимодействовать с MetaMask. Это криптовалютный кошелек, который работает как расширение для браузеров. Через MetaMask покупатель производит основные транзакции на счет смарт-контракта. Также эта библиотека позволяет работать с блоками сети Ethereum, то есть напрямую обращаться к смарт-контракту, используя HTTP, IPC или WebSocket. В нашем случае для связи клиента с сервером использовался протокол HTTP.

API (Программный интерфейс приложения) — это описание способов, то есть набор классов, структур, функций или процедур, которыми одна программа может взаимодействовать с другой.

HTTP — протокол прикладного уровня передачи данных, изначально — в виде гипертекстовых документов в формате HTML, в настоящее время используется для передачи произвольных данных. Основой HTTP является технология «клиент-сервер», то есть предполагается существование:

- Потребителей (клиентов), которые инициируют соединение и посылают запрос;
- Поставщиков (серверов), которые ожидают соединения для получения запроса, производят необходимые действия и возвращают обратно сообщение с результатом.

Back-end

Back-end, то есть серверная часть нашего проекта, написан на языке Python используя фреймворк Django.

Django - open-source фреймворк для веб-приложений на языке Python, использующий шаблон проектирования MVC. Проект поддерживается организацией Django Software Foundation.

Фреймворк — программное обеспечение(платформа), облегчающее разработку и объединение разных компонентов большого программного проекта.

Model-View-Controller(MVC,«Модель-Представление-Контроллер») — схема разделения данных приложения, пользовательского интерфейса и управляющей логики на три отдельных компонента: модель, представление и контроллер — таким образом, что модификация каждого компонента может осуществляться независимо.

- Модель (Model) предоставляет данные и реагирует на команды контроллера, изменяя свое состояние.
- Представление (View) отвечает за отображение данных модели пользователю, реагируя на изменения модели.
- Контроллер (Controller) интерпретирует действия пользователя, оповещая модель о необходимости изменений.

Архитектура Django похожа на «Модель-Представление-Контроллер» (MVC). Контроллер классической модели MVC примерно соответствует уровню, который в Django называется Представление (англ. View), а презентационная логика Представления реализуется в Django уровнем Шаблонов (англ. Template). Из-за этого уровневую архитектуру Django часто называют «Модель-Шаблон-Представление» (MTV).

Сайт на Django строится из одного или нескольких приложений, которые рекомендуется делать отчуждаемыми и подключаемыми. Это одно из существенных архитектурных отличий этого фреймворка от некоторых других (например, Ruby on Rails). Один из основных принципов фреймворка — DRY (англ. Don't repeat yourself)

Для работы с базой данных Django использует собственный ORM, в котором модель данных описывается классами Python, и по ней генерируется схема базы данных.

ORM - технология программирования, которая связывает базы данных с концепциями объектно-ориентированных языков программирования, создавая «виртуальную объектную базу данных».

Благодаря этой технологии можно обращаться к базе данных на языке Python, не используя язык запросов SQL. Или например в процессе развития вашего проекта сменить базу данных на более производительную, так как ORM Django умеет общаться почти со всеми

популярными базами данных(PostgreSQL, MariaDB, MySQL, Oracle and SQLite, MongoDB).

Так как этот проект не рассчитан на большие нагрузки, мы использовали SQLite базу данных. Она выделяется тем, что для ее работы не требуется серверная часть. Из минусов можно выделить низкую скорость работы(по сравнению с другими SQL БД), а также отсутствие возможности масштабировать базу данных.

Smart-contract

В данной работе использовалась платформа Ethereum, как основа для смарт-контракта, которая работает на базе блокчейна. Сам контракт выступает посредником в оплате, который позволяет избавиться от третьей стороны и обезопасить транзакцию.

Smart-contract — компьютерный алгоритм, предназначенный для ,формирования, контроля и предоставления информации о владении чем-либо. Чаще всего речь идет о применении технологии блокчейна. Смарт-контракты Ethereum разрабатываются на одном из языков, спроектированных для трансляции в байт-код виртуальной машины Ethereum — Solidity, Vyper и Serpent, LLL, Mutan. Наша команда использовала язык Solidity для разработки смарт-контракта.

Блокчейн — выстроенная по определённым правила непрерывная последовательная цепочка блоков, содержащих информацию. Связь между блоками обеспечивается не только нумерацией, но и тем, что каждый блок содержит свою собственную хеш-сумму и хеш-сумму предыдущего блока.

Ethereum — криптовалюта и платформа для создания децентрализованных онлайн-сервисов на базе блокчейна (децентрализованных приложений), работающих на базе умных контрактов.

Solidity — объектно-ориентированный, предметно-ориентированный язык программирования самовыполняющихся контрактов для платформы Ethereum.

ДЕТАЛИЗАЦИЯ РЕАЛИЗАЦИИ

Для удобства взаимодействия покупателя с магазином наш сайт имеет несколько модулей:

- Главная страница

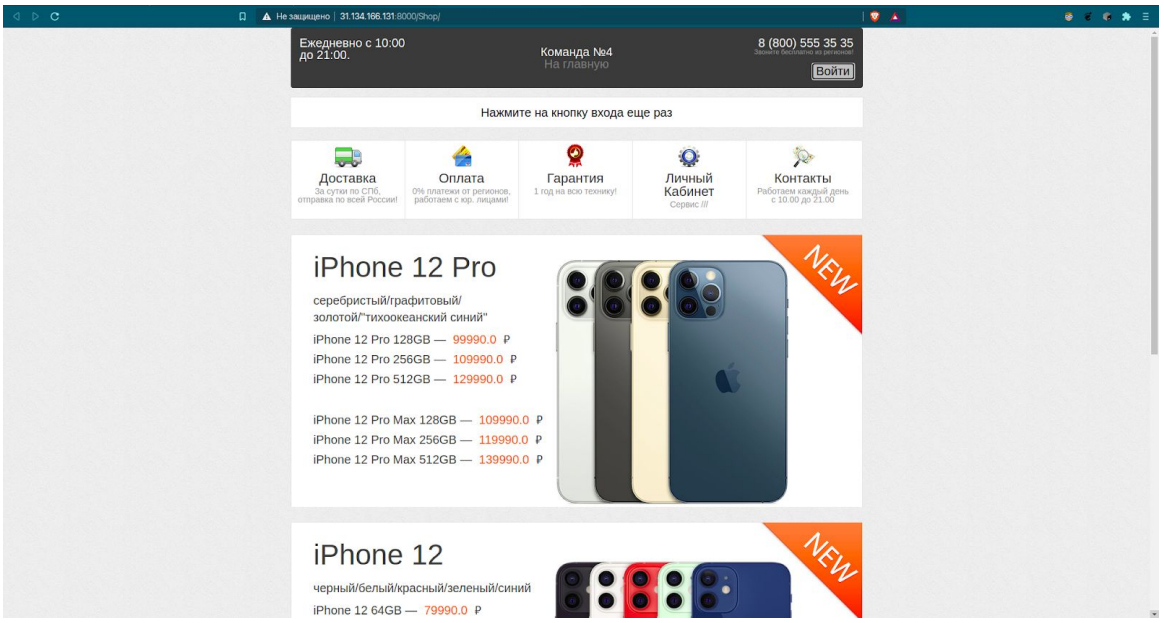

рис.3 Главная страница сайта

Данная страница является стартовой при переходе на сайт. В header’е данного сайта (то есть в его верхней части) расположена кнопка «Войти», при нажатии на которую всплывает дополнительное окно входа в личный кабинет через кошелек MetaMask.

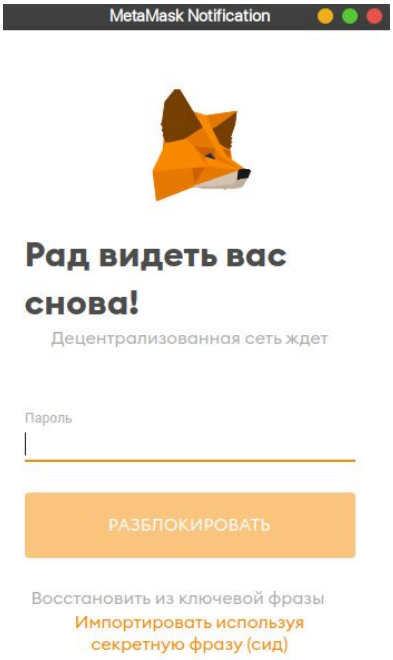

рис.4 Окно авторизации

После авторизации на главной странице сайта появится надпись: «Вы вошли под кошельком с адресом: 0x***». Теперь покупатель может приобрести товар нажав на его цену, а затем оплатить выбранный им продукт благодаря кошельку MetaMask.

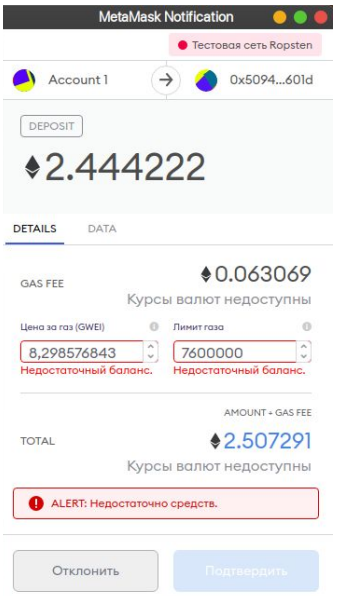

рис.5 Окно оплаты

- Доставка

  На данной странице размещена информация о сроках и способах доставки

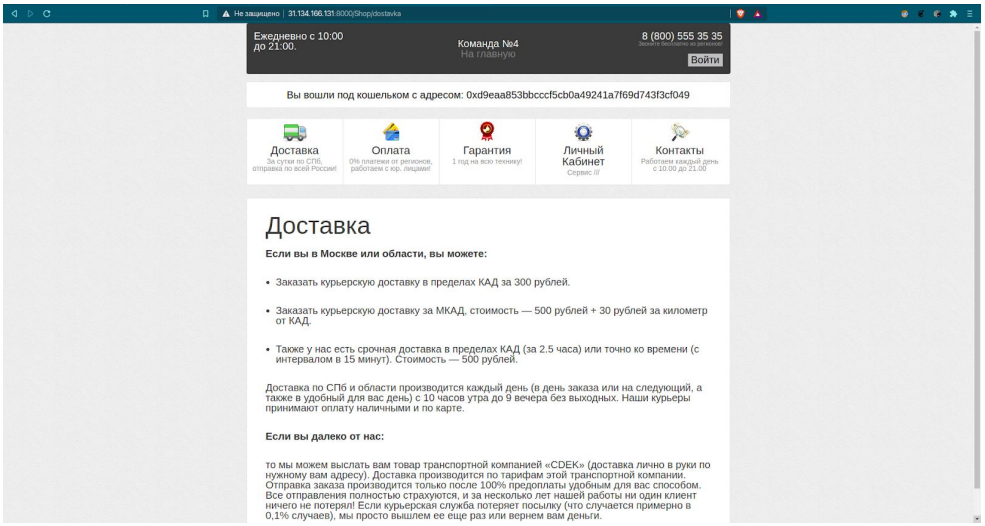

рис.6 Страница доставки

- Оплата

  На этой странице пользователь может увидеть информацию о способах оплаты и возможностях возместить или же вернуть средства

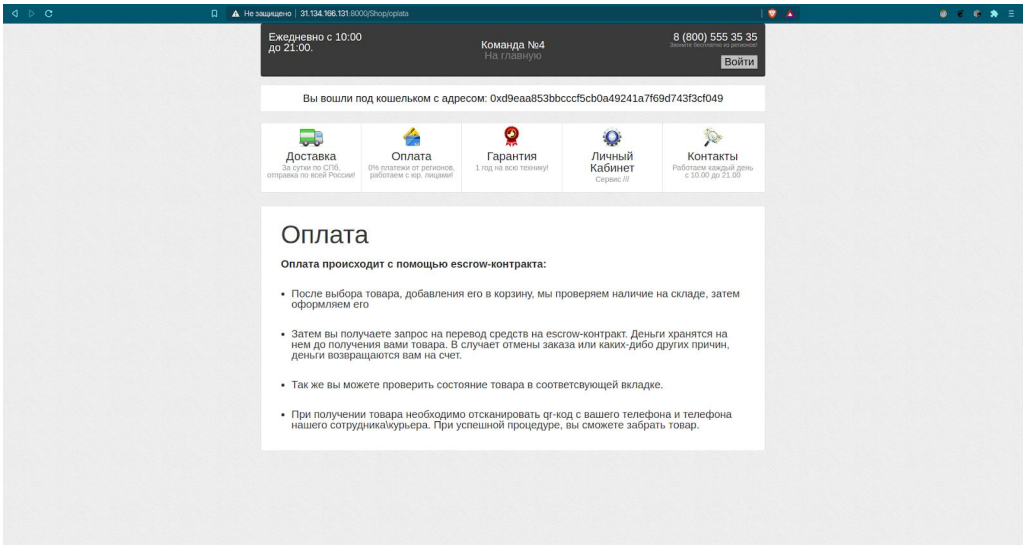

рис.7 Страница с информацией об оплате

- Гарантия

  На этой странице указаны условия и сроки гарантийного обслуживания приобретаемых товаров

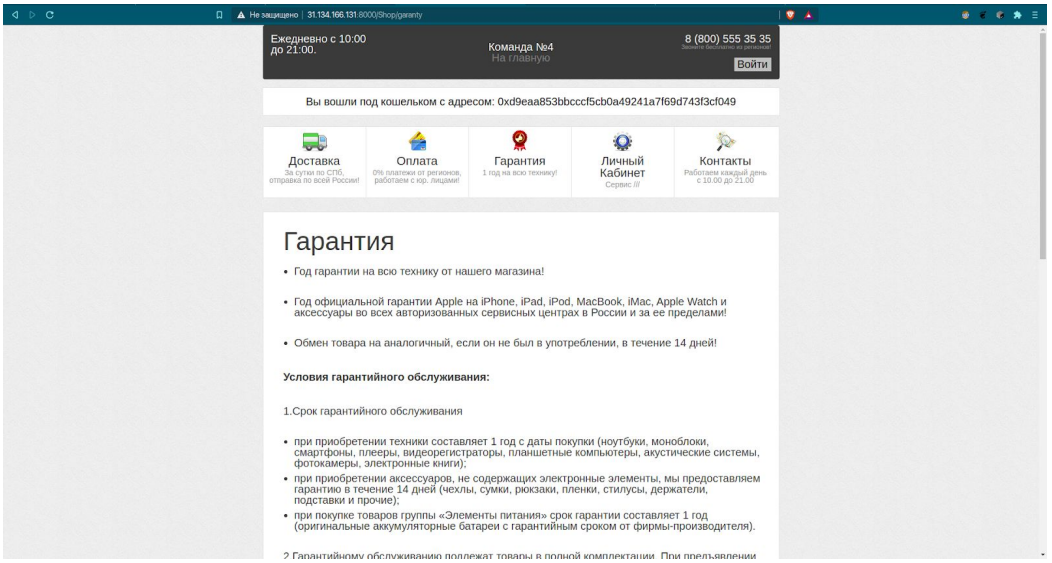

рис.8 Страница с информацией о гарантии

- Личный кабинет

  На данной странице содержится информация о заказах конкретного покупателя, также после заказа пользователь может с помощью двух кнопок ВОЗМЕСТИТЬ или ОТМЕНИТЬ ЗАКАЗ, произвести описанные действия.

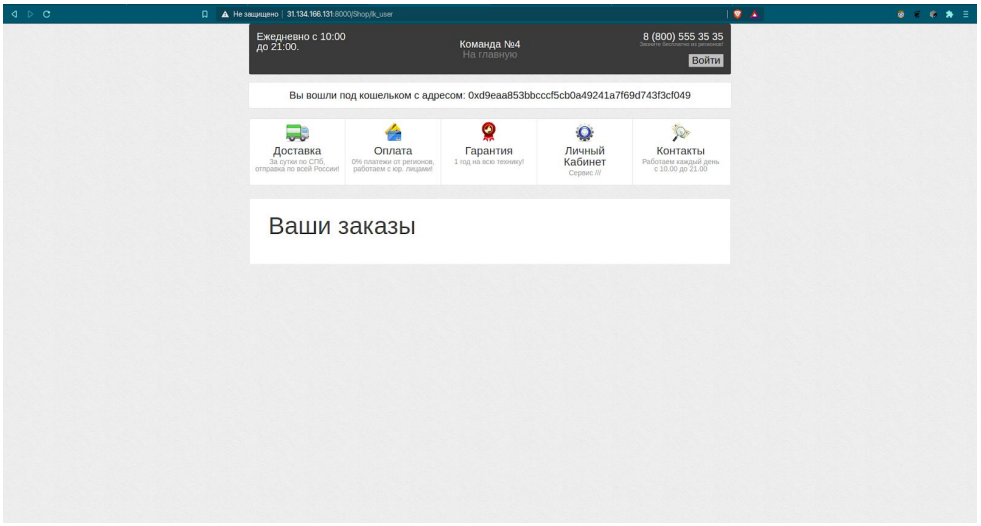

рис.9 Личный кабинет пользователя

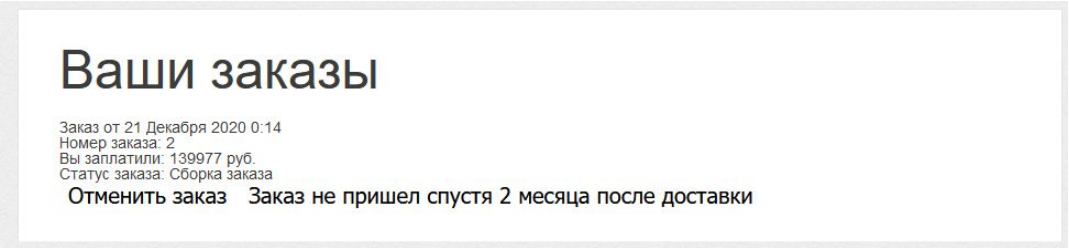

рис.10 История заказов

- Контакты

Контактная информация для связи с поддержкой интернет-магазина

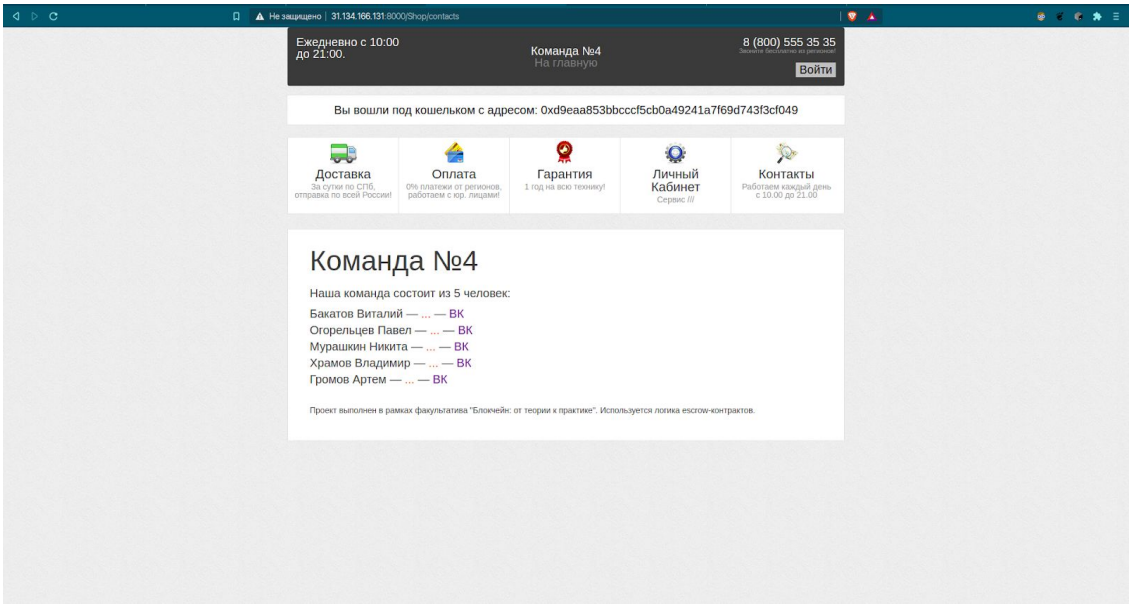

рис.11 Страница контактной информации

ЗАКЛЮЧЕНИЕ

Подводя итоги нашей работы, можно сказать, что мы справились с техническим заданием, создав клиентоориентированный интернет-магазин со смарт-контрактом, который позволяет не бояться за свои средства ни покупателю, ни продавцу.

Таким образом, любые попытки обойти систему оплаты и транзакций будут неуспешны, потому что Escrow-контракт обеспечивает надлежащую безопасность. Надежность сервиса подтверждена множественными тестами транзакций, отклика сайта и записей пользователей в базу данных.

СПИСОК ИСТОЧНИКОВ

1. https://web3js.readthedocs.io/ -документация библиотеки Web3JS
2. Chris Dannen. Introducing Ethereum and Solidity: Foundations of Cryptocurrency and Blockchain Programming for Beginners. — 2017. — С. 256.
3. https://docs.djangoproject.com/ - документация фреймворка Django
4. https://docs.soliditylang.org/ - документация контрактно-ориентированного языка программирования Solidity
5. https://stackoverflow.com/ - вопросы и проблемы с кодом

ПРИЛОЖЕНИЯ

1. https://github.com/AA122AA/Escrow_site - репозиторий с кодом программ
2. http://31.134.166.131:8000/Shop/ - рабочий сайт