МИНОБРНАУКИ РОССИИ

Федеральное государственное автономное   
образовательное учреждение высшего образования

«ЮЖНЫЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Институт компьютерных технологий и информационной безопасности

|  |  |
| --- | --- |
| Руководитель:  Доцент кафедры БИТ к.т.н., Е.А. Ищукова  \_\_\_\_\_\_\_\_\_\_\_\_  *(подпись)*  «\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_ 20\_\_ г. | К защите допустить:  Доцент кафедры ИБТКС  к.т.н., А.П. Плёнкин  \_\_\_\_\_\_\_\_\_\_\_\_\_  *(подпись)*  «\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_ 20\_\_ г. |

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

**К ТВОРЧЕСКОМУ ПРОЕКТУ**

ПО ДИСЦИПЛИНЕ «ВвИД»

|  |  |  |  |
| --- | --- | --- | --- |
| на тему: | Система для составления рейтинга студента на базе blockchain-технологий | | |
| Команда: FARGO | | |
| Выполнили: | | | Капитан команды Хорошилов Кирилл Сергеевич, КТбо2-5 | |
|  | | | *(фамилия, имя, отчество, группа)* | |
| Саядян Айк Арменович, КТбо2-5 | | |
| *(фамилия, имя, отчество, группа)* | | |
| Уажанов Тимур Бакытович, КТбо2-5 | | |
| *(фамилия, имя, отчество, группа)* | | |
| Соколов Павел Викторович, КТбо2-5 | | |
| *(фамилия, имя, отчество, группа)* | | |

«\_\_\_» мая 2020 г.

Таганрог 2020 г.

Список исполнителей

Хорошилов Кирилл: капитан команды, разработчик smart-контракта, организует работу команды, следит за сроками выполнения проекта.

Саядян Айк: проектировщик веб-приложения, отвечает за визуальный интерфейс.

Уажанов Тимур: проектировщик веб-приложения, тестировщик

Соколов Павел: разработчик, отвечает за отчетность команды.

Техническое задание на проект

Создать программу, позволяющую, на основе, предоставленных ей данных, составлять систему рейтинга, зарегистрированных в ней студентов (успеваемость, посещаемость, предоставление льгот и т.д.) на основе Blockchain-технологий

Требования к проекту:

- разработать ролевую модель: модель взаимодействия пользователей распределенной базы данных, роли участников и их функциональные возможности;

- определить наиболее подходящий механизм консенсуса для blockchain цепочки в соответствии с разработанной ролевой моделью;

- реализовать приложение с использованием blockchain-технологии с соблюдением требований, заданных в ТЗ.

Реферат

Пояснительная записка содержит 42 страниц, 8 рисунков, 5 таблицы, 12 источников и 3 приложения.

В ходе выполнения работы реализована система типа BRS на базе blockchain-технологии. Для достижение поставленной задачи были выявлены следующие подзадачи:

- изучение аналогов программы;

- реализация smart-контракта;

- создание сайта;

- создание Dapp;

В данной работе пошагово описан процесс реализации своей системы на основе blockchain-технологии и принцип ее работы. Проект является командной работой, которой каждый участник сделал существенный вклад.

Содержание

[Список исполнителей 2](#_Toc40466382)

[Техническое задание на проект 3](#_Toc40466383)

[Реферат 4](#_Toc40466384)

[Введение 7](#_Toc40466385)

[1 Анализ технического задания 8](#_Toc40466386)

[1.1 Этапы выполнения 8](#_Toc40466387)

[1.2 Распределение обязанностей 8](#_Toc40466388)

[1.3 План реализации 9](#_Toc40466389)

[2 Анализ подходов и решений 11](#_Toc40466390)

[2.1 Blockhain 11](#_Toc40466391)

[2.2 Как работает Blockhain 11](#_Toc40466392)

[2.3 Аналоги систем рейтинга 13](#_Toc40466393)

[2.4 Преимущества Blockchain 14](#_Toc40466394)

[2.5 Функционал проекта 14](#_Toc40466395)

[3 Разработка бэкэнда приложения 16](#_Toc40466396)

[3.1 Язык программирования смарт-контрактов Solidity 16](#_Toc40466397)

[3.2 Парсинг сайта БРС 19](#_Toc40466398)

[4 Разработка веб-приложения 21](#_Toc40466399)

[4.1 Разработка интерфейса приложения 21](#_Toc40466400)

[5 Объединение результатов работы 23](#_Toc40466401)

[5.1 NodeJS 23](#_Toc40466402)

[5.2 Express 23](#_Toc40466403)

[5.3 EJS 24](#_Toc40466404)

[Результат проделанной работы 25](#_Toc40466405)

[Заключение 28](#_Toc40466406)

[Список использованных источников 29](#_Toc40466407)

Приложение А……………………………………………………………………31

Приложение Б……………………………………………………………………36

Приложение В……………………………………………………………………39

Введение

Рейтинг студента будет представлять собой программу на основе Blockchain-технологий, с помощью которой пользователь сможет самостоятельно отслеживать собственную успеваемость в университете среди множества других студентов в определённом рейтинге, который будет выдавать определенные бонусы для тех, кто занимает лидирующие позиции.

Рассмотрим понятие Blockchain. Blockchain – децентрализованная база данных, представляющая собой непрерывную последовательность блоков, содержащих информацию. Все последующие блоки цепочки связаны с предыдущими криптографической подписью и имеют строгий хронологический порядок. Каждая новая транзакция подписывается и, после внесения её в цепочку блоков, распространяется по всем узлам. Сама сеть состоит из узлов или «нод» (node), которые хранят полную и актуальную версию Blockchain. Основными преимуществами, Blockchain-а является высокая скорость проведения операций, низкая стоимость транзакций. Кроме того, распределённая база, Blockchain-а минимизирует возможность подделки транзакций, поскольку практически невозможно получить доступ ко всем узлам сети одновременно.

Аналогичные системы рейтинга студентов существуют, но ни в одной из них не реализовывалась система на основе Blockchain-а. В связи с этим можно утверждать, что основное преимущество нашего проекта – надежность, так как ни один пользователь не сможет подделать свои данные в рейтинге. Неофициальный лозунг биткоина: «Верим в криптографию», прямо говорит о том, кто виноват в проблемах экономики: посредники, банкиры, «доверенные» третьи стороны, которым на самом деле нельзя доверять. Эти люди просто создают проблемы другим, уменьшая прибыль и усложняя транзакции. Крипто валюты заменяют фоновые операции, создаваемые посредниками, и транзакции при помощи программного обеспечения — распределенной и защищенной базы данных, называемой блокчейном. При этом процесс смены владельцев токена биткоина контролируют множество компьютеров. Также блокчейн технология надежна тем, что майнеры тратят свои средства на поддержку сети — покупают оборудование и платят за электроэнергию. Чтобы изменить блок в блокчейне и провести одну и ту же транзакцию дважды, им придётся потратить в два раза больше своих денег, поэтому обманывать становится невыгодно. Кроме того, с каждым новым блоком возрастает стоимость изменения предыдущих. Новые блоки хранят хеш стоящего перед ними блока.

1. Анализ технического задания
   1. Этапы выполнения

В соответствии с техническим заданием, ставится задача разработать систему с использованием технологии Blockchain, позволяющую узнавать место в рейтинге студентов. В ходе выполнения этой задачи, необходимо:

изучить материал по теме проекта (Blockchain, пиринговая сеть, алгоритмы шифрования, языки программирования – Python, Solidity);

- провести анализ сходных программ, найти их преимущества и недостатки;

- создать свою систему на языке «Solidity»;

- создать несколько ролей пользователей с разными правами в системе (и выделением специальной роли для подтверждения загруженных достижений в системе);

- организовать хранение данных пользователей через распределенное файловое хранилище;

- разработать веб-приложение для просмотра своего места в рейтинге.

Во время первого этапа нужно будет: разобраться в технологиях, связанных с этой темой, распределить роли в команде, провести анализ похожих «рейтингов студентов», определится с функционалом и стеком используемых технологий.

К моменту окончания второго отчетного периода необходимо выполнить основную часть работы и реализовать: Smart-контракт на Solidity и клиентское веб-приложение.

Третий отчетный период подразумевает тестирование и улучшение системы, а также подготовку к защите проекта.

* 1. Распределение обязанностей

Задачи распределены между всеми членами команды, в зависимости от их роли в команде. Каждую поставленную подзадачу необходимо выполнить к определенному сроку. В таблице 1 представлено распределение обязанностей в виде матрицы ответственности. где введены следующие обозначения: И исполнитель, О ответственный, Н наблюдатель, К – консультант.

Таблица 1 – Распределение обязанностей

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Срок выполнения | Подзадачи | Команда проекта | | | | Куратор проекта |
| Руководитель проекта | Саядян А.А. | Уажанов Т.Б. | Соколов П.В. | Ищукова  Е.А. |
| Хорошилов К.С. |
| 1 января | Изучение материала | И | И | И | И | К |
| 1 февраля – 1 марта | Реализация контракта | И/О | Н | Н | Н | К |
| 1 марта | Реализация авторизации | И | И | Н | Н | К |
| 1 марта – 1 апреля | Создание приложения | И | И/О | И | И | К |
| 1 апреля | Объединение приложения и кода | И/О | И | И | И | К |
| 1 апреля – 1 мая | Тестирование | И | И | И | И | Н |
| Устранение ошибок | И/О | И | И | И | Н |
| 1 мая | Подготовка к защите | И/О | И | И | И | К |
| 15 мая | Защита проекта | И/О | И | И | И | Н |

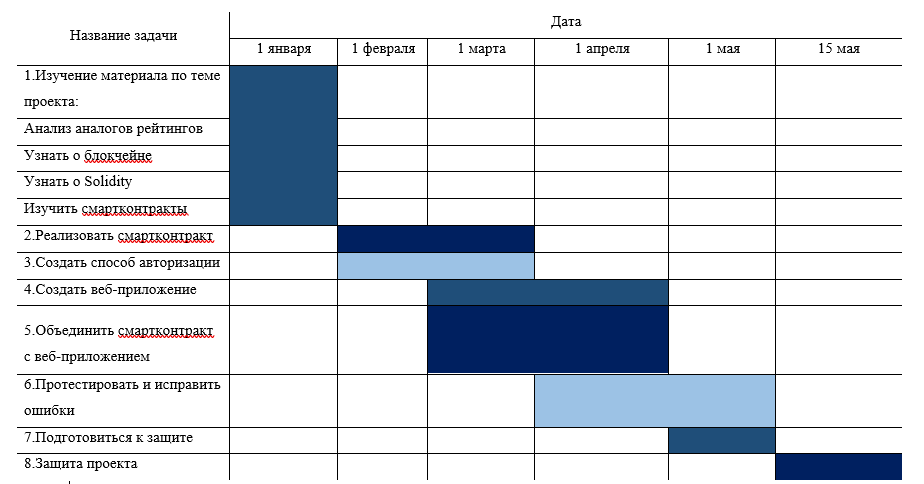
* 1. План реализации

План реализации разбит на 3 стадии в соответствии с отчетными периодами и приведен в таблице 2.

Таблица 2 – План реализации проекта

|  |  |
| --- | --- |
| Отчетный период | Что должно быть сделано по проекту |
| До первого февраля | 1. Изучен материал  2. Распределены роли в команде  3. Определены сроки выполнения  4. Сделано введение ИЗ |
| До первого апреля | 1. Создана программа  2. Практически завершено веб-приложение  3. Сделан основной блок ИЗ |
| До первого мая | 1. Протестировано и исправлено веб-приложение  2. Завершено приложение  3. Завершена подготовка к защите |

Так же план реализации поставленных целей и промежуточных задач можно представить с помощью ленточного графика (Рисунок 1).



1. Ленточный график выполнения работы
2. Анализ подходов и решений
   1. Blockhain

Blockhain (цепочка блоков) — это распределённая база данных, у которой устройства хранения данных не подключены к общему серверу. Эта база данных хранит постоянно растущий список упорядоченных записей, называемых блоками. Каждый блок содержит метку времени и ссылку на предыдущий блок.

Применение шифрования гарантирует, что пользователи могут изменять только те части цепочки блоков, которыми они «владеют» в том смысле, что у них есть закрытые ключи, без которых запись в файл невозможна. Кроме того, шифрование обеспечивает синхронизацию копий распределённой цепочки блоков у всех пользователей.

В технологию блокчейн изначально заложена безопасность на уровне базы данных. Концепцию цепочек блоков предложил в 2008 году Сатоши Накамото (Satoshi Nakamoto). Впервые реализована она была в 2009 году как компонент цифровой валюты — биткоина, где блокчейн играет роль главного общего реестра для всех операций. Благодаря технологии блокчейн биткоин стал первой цифровой валютой, которая решает проблему двойных расходов (в отличие от физических монет или жетонов, электронные файлы могут дублироваться и тратиться дважды) без использования какого-либо авторитетного органа или центрального сервера.

Безопасность в технологии блокчейн обеспечивается через децентрализованный сервер, проставляющий метки времени, и одноранговые сетевые соединения. В результате формируется база данных, которая управляется автономно, без единого центра. Это делает цепочки блоков очень удобными для регистрации событий (например, внесения медицинских записей) и операций с данными, управления идентификацией и подтверждения подлинности источника [1].

* 1. Как работает Blockhain

В 2009 году анонимный хакер (или группа хакеров), под псевдонимом Сатоши Накамото, создал первую цифровую валюту. В этой системе деньги были лишь инструментом учета, методом абстрагирования стоимости, назначения собственности и предоставления средств для совершения сделок.

Для выполнения этих функций исторически использовались денежные средства. Обладание физическими токенами — монетами, позволяет людям лично заключать сделки между собой. Наличные деньги достаточно трудно скопировать, поэтому нет необходимости в полном учете того, кому принадлежит определенная часть денежной массы.

Тем не менее, если создать таблицу, в которой указано, кому и сколько принадлежит денег, монеты и купюры станут ненужными. Банки и обработчики платежей уже частично сублимировали физическую валюту в цифровые записи, отслеживая и обрабатывая транзакции в своих закрытых системах.

Биткоин завершил преобразование, создав единый универсальный цифровой регистр, называемый блокчейном. Эта технология получила такое название, потому что похожа на цепь — внести в неё изменения можно только в конец блоков. Каждое новое дополнение содержит набор новых транзакций. Например, если Саша заплатит Юле за биткоин, эта транзакция появится в конце цепи. А в блоках до этого будет указано, что Саше заплатил Миша, а Мише Оля.

Блокчейн для биткоина, в отличие от книг бухгалтерского учёта, которые ведут традиционные финансовые учреждения, расположен на компьютерах по всему миру. Эти данные доступны для всех, у кого есть подключение к интернету. Майнеры, — владельцы компьютеров, на которых хранится информация блокчейна, — отвечают за обнаружение запросов транзакций от пользователей, их объединение, проверку и добавление в блокчейн в виде новых блоков.

Процесс валидации устанавливает, что человек фактически владеет биткоинами после транзакции, и что он еще не потратил их в другом месте. Собственность в блокчейне определяется двумя криптографическими ключами. Первый ключ находится в блокчейне в открытом доступе. Второй доступен только его владельцу. Такие ключи используются для шифрования электронных сообщений. Когда кто-то отправляет зашифрованное сообщение, он использует открытый ключ. Получатель при открытии письма использует закрытый ключ и расшифровывает сообщение.

В технологии блокчейн транзакции подписываются при помощи закрытых ключей, соответствующих открытым ключам, присвоенным монетам, которые хотят потратить. И когда транзакция обрабатывается, этим монетам присваивается новый открытый ключ.

Когда в оформлении операции участвует несколько лиц, становится важен вопрос необратимости. Если бы блокчейн управлялся одним банком с набором известных валидаторов, работающих в рамках одной юрисдикции, то выполнение транзакций было бы простым делом.

Но для биткоина нет центрального банка, обеспечивающего соблюдение правил. Майнеры работают анонимно во всем мире, несмотря на разнообразие культур, различие правовых систем и нормативных обязательств. Поэтому нет способа привлечь их к ответственности. Необратимость операции обеспечивает код биткоина. Он использует схему, которая называется доказательством работы [2].

* 1. Аналоги систем рейтинга

Существует не много систем для составления рейтинга студентов. Среди них можно выделить «1С Университет» и ее более продвинутую версию «1С Университет ПРОФ».

Характеристики этих систем представлены в таблице 3[3].

Таблица 3 – Характеристика систем «1С Университет»

|  |  |  |
| --- | --- | --- |
| Функционал | 1С Университет | 1С: Университет ПРОФ |
| Организация и проведение приемной компании | + | + |
| Планирование учебного процесса, поддержка ФГОС-3, расчет и распределение учебной нагрузки | + | + |
| Управление контингентом студентов (личные дела, учет успеваемости и посещаемости, воинский учет) | + | + |
| Формирование приказов и отчетности | + | + |
| Формирование регламентированной отчетности и документов (форма ВПО-1, дипломы, приложения, справки, приложение №2 к Перечню | + | + |
| Учет трудоустройства выпускников | + | + |
| Учет оплаты за обучение и назначение стипендий | + | + |
| Обмен данными с информационными системами: GosInsp, ФИС ЕГЭ и приема, 1С: Зарплата и кадры государственного учреждениия, ФРДО, ГИС Контингент | + | + |
| Расписание учебных занятий | - | + |
| Автоматизация проведения итоговой государственной аттестации | - | + |
| Формирование регламентированной отчетности и документов (отчет о научной деятельности, форма №2-наука, форма №1-технология, форма №1-НК, информационные карты, договор найма, ордер, карточки и формы для постановки проживающих в общежитии на учет) | - | + |
| Управление научной деятельностью и инновациями | - | + |
| Управление послевузовским образованием (аспирантура, докторантура) и аттестацией научных кадров | - | + |
| Автоматизация деятельности диссертационных советов | - | + |
| Управление дополнительным образованием | - | + |
| Управление кампусом вуза | - | + |
| Личные Кабинеты абитуриента, студента и преподавателя | - | + |

* 1. Преимущества Blockchain

Преимущества Blockchain-технологий по сравнению с данными системами заключается в:

- Отсутствие центрального сервера, распределение базы данных между всеми участниками сети;

- Каждый пользователь имеет на своем компьютере копию базы данных со всеми совершенными транзакциями в зашифрованном виде. Чтобы базы у всех были одинаковыми, они синхронизируются;

- Система абсолютно прозрачна, так как каждый участник может видеть любую транзакцию;

- Новая информация добавляется в базу данных, путем открытия новых блоков. Это добавление согласовывается со всеми пользователями сети путем проверки ими хеша нового блока. Только после этого новый блок прописывается в базе данных каждого пользователя.

- Благодаря распределению базы среди всех участников, взломать ее невозможно, так как подмена, замена или добавление блоков сразу выявляется путем сравнивания баз на разных компьютерах.

* 1. Функционал проекта

Сам по себе смарт-контракт будет активироваться при транзакции(сообщении\передачи) сначала от преподавателя, а потом от учащегося. Для того, чтобы активировать умные контракты Etherium, потребуется необходимое количество газа (Gas), которым оплачиваются транзакции в сети Эфира. То есть на преподавателя ложится обязанность заполнить контракт оценкой и нажать отправить, а от учащегося требуется принять контракт(вызвать) и «замайнить» контракт.

Права доступа пользователя: разделение прав доступа обусловлены тем, что в системе БРС обычно используется система студент-преподаватель, где студенту представляется возможность рассматривать свою успеваемость, а преподавателю заполнять электронный журнал.

Веб-приложение – с присутствующей формой авторизации-регистрации, где преподавателей нужно регистрировать по особой форме. Имеется личный кабинет (электронный журнал), где хранятся данные по отметкам по каждой дисциплине. Сама по себе учебная программа должна быть разработана преподавателем до начала учебного курса (чтобы создать «таблицу», в которую и будут заносится оценки). В систему будут входить данные о посещениях, письменных и индивидуальных работах, бонусных баллах. Будет строгое ограничение по времени в отношении сдачи работы (это одно из условий для выполнения смарт контракта).

Поскольку смарт-контракты будут надежными, то отпадет нужда в центральном органе, проверяющем сертификаты. Дипломы будут доступны всем, и проверить информацию станет гораздо проще.

Оплата обучения в заведениях (из-за того, что работа смарт-контрактов требует Gas). В итоге ученики смогут «условно говоря» получать криптовалюту обрабатывая смарт-контракты. В таком случае выплаты будут совершаться мгновенно и автоматически.

1. Разработка бэкэнда приложения
   1. Язык программирования смарт-контрактов Solidity

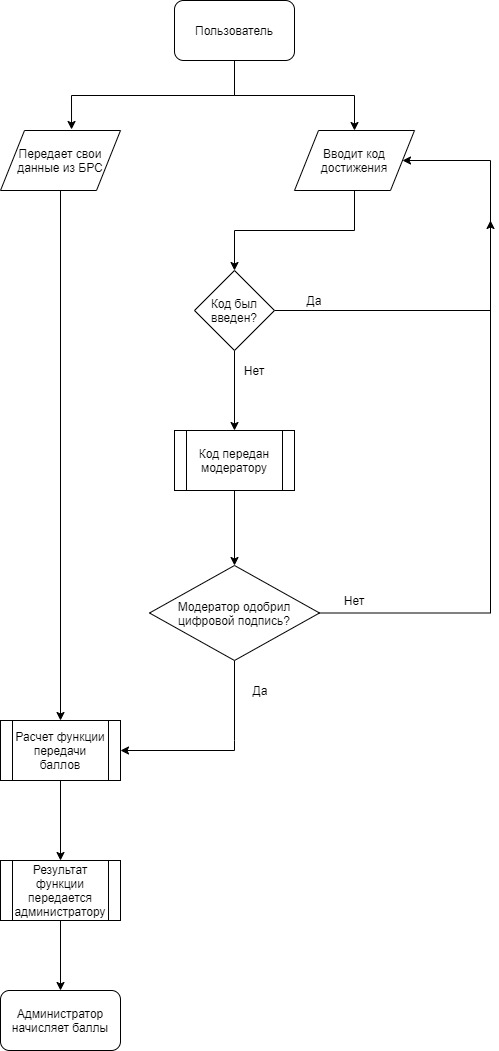
Для реализации Blockchain в данном проекте будет использоваться язык «смарт-контрактов» Solidity, с помощью которого будет организовываться цепочка проверок условий, в результате которой будет происходит передача нашей «криптовалюты» - баллов.

Немного подробнее о самом языке. Важным элементом в системе Ethereum являются так называемые смарт-контракты. Грубо говоря, это довольно мощные и функциональные программы для реализации определенных обязательств. Виртуальная машина Ethereum (EVM) позволяет узлам сети Ethereum фактически хранить и обрабатывать данные в обмен на оплату эфиром, Она предоставляет множество новых возможностей для поддержки сетевых приложений, которые никогда ранее не были доступны разработчикам и обычным пользователям.

Solidity – это инструмент, который применяется для создания кода машинного уровня, который может выполняться на EVM. По сути, это язык с компилятором, который берет высокоуровневый читаемый человеком код и разбивает его на простые инструкции, которые составляют основу любой исполняемой программы микропроцессора. Solidity позволяет создавать и выполнять интеллектуальные контракты, не требуя централизованных или доверенных сторон. Это статически типизированный язык программирования контрактов, который имеет сходство с такими языками программирования, как Javascript и C. Подобно объектам в ООП, каждый контракт содержит переменные состояния, функции и общие типы данных.

Solidity является всего лишь одним из нескольких языков, которые могут компилировать написанные программы в байт-код EVM. Другой язык, который делает то же самое, называется Serpent. Каждый из них может иметь несколько инструментов компиляции, но все они делают то же самое – генерируют байт-код на машинном уровне EVM для запуска приложений на узлах сети Ethereum [4].

Смарт-контракт будет реализован по блок-схеме (Рисунок 2)



1. Блок-схема «Смарт-контракта»

Контракт поделен на модули и за каждым модулем закреплен определенный функционал.

Первый модуль – контракт для обозначения создателя (администратора) и студентов. В нем хранится адрес создателя, конструктор контракта присваивающий адрес создателя и два модификатора: доступ создателя и доступ для всех кроме создателя.

Второй модуль хранит в себе безопасные атематические операции с uint/uint256: умножение, деление, вычитание и сложение.

Третий модуль предназначен для обозначения уникального диплома или грамоты. В нем хранится функция занесения дипломов, события отправки и занесения дипломов. Также в нем хранится структура диплома, массив дипломов для хранения, переменные для хранения количества дипломов у определенного студента и переменные для проверки наличия диплома у определенного студента. В этом же модуле находятся функции вывода количества дипломов у определенного студента, вывода владельца диплома по уникальному номеру и функция передачи диплома определенному студенту.

Четвертый модуль выделен под баллы. В нем обозначаются общее количество баллов в системе, переменная для хранения суммы баллов и переменная для хранения количества учебных баллов у определенного студента. Также в нем находятся функция по выводу количества баллов у определенного студента и функция по передаче баллов определенному студенту. Еще в этом же модуле находится событие передачи баллов определенному студенту.

Пятый модуль – основной контракт. В нем хранится переменная для хранения суммы баллов, конструктор контракта для передачи общего количества баллов администратору. В этом же модуле находятся несколько функций: для начисления баллов за заслуги, проверки диплома, для передачи баллов в блокчейн. Еще здесь обозначаются события о передаче оценок в блокчейн и о создании количества всех баллов. Полный листинг контракта (см. Приложение А)

Ролевая система: администратору выдаются права по распоряжению баллов в системе, добавление дипломов в систему, а пользователям (учащимся) выдаются права по активации дипломов и просмотр баллов.

* 1. Парсинг сайта БРС

Вся информация о пользователе будет «парситься» с сайта БРС ЮФУ, по средствам языка Python. Для данного действия в программе подключатся открытые библиотеки BeutifulSoup, Selenium и Requests.

Beautiful Soup - это парсер для синтаксического разбора файлов HTML/XML, написанный на языке программирования Python, который может преобразовать даже неправильную разметку в дерево синтаксического разбора. Он поддерживает простые и естественные способы навигации, поиска и модификации дерева синтаксического разбора. В большинстве случаев он поможет программисту сэкономить часы и дни работы. Написанный на языке программирования Ruby порт называется Rubyful Soup [5].

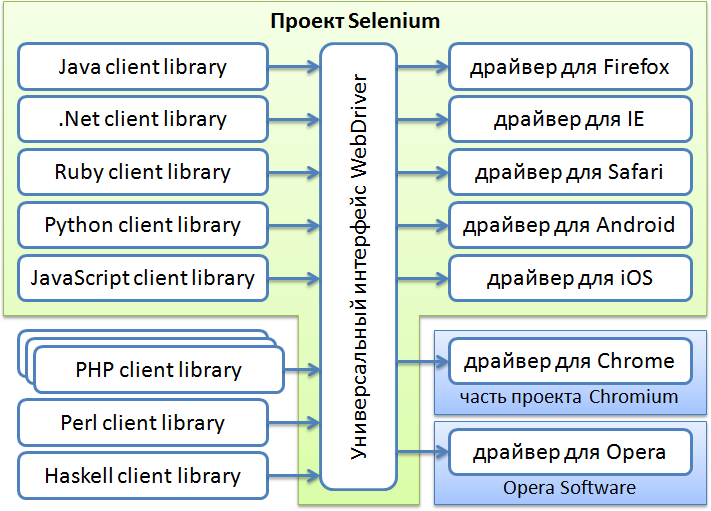
Selenium WebDriver – это программная библиотека для управления браузерами. Часто употребляется также более короткое название WebDriver.

Иногда говорят, что это «драйвер браузера», но на самом деле это целое семейство драйверов для различных браузеров, а также набор клиентских библиотек на разных языках, позволяющих работать с этими драйверами.

Это основной продукт, разрабатываемый в рамках проекта Selenium.

Selenium WebDriver называется также Selenium 2.0, причина этого будет объяснена ниже.

Как уже было сказано, WebDriver представляет собой семейство драйверов для различных браузеров плюс набор клиентских библиотек для этих драйверов на разных языках программирования (Рисунок 3).



1. Проект Selenium

В рамках проекта Selenium разрабатываются драйверы для браузеров Firefox, Internet Explorer и Safari, а также драйверы для мобильных браузеров Android и iOS. Драйвер для браузера Google Chrome разрабатывается в рамках проекта Chromium, а драйвер для браузера Opera (включая мобильные версии) разрабатывается компанией Opera Software. Поэтому они формально не являются частью проекта Selenium, распространяются и поддерживаются независимо. Но логически, конечно, можно считать их частью семейства продуктов Selenium [6].

Requests - это модуль Python, который вы можете использовать для отправки всех видов HTTP-запросов. Это простая в использовании библиотека с множеством функций, начиная от передачи параметров в URL-адресах до отправки пользовательских заголовков и проверки SSL [7].

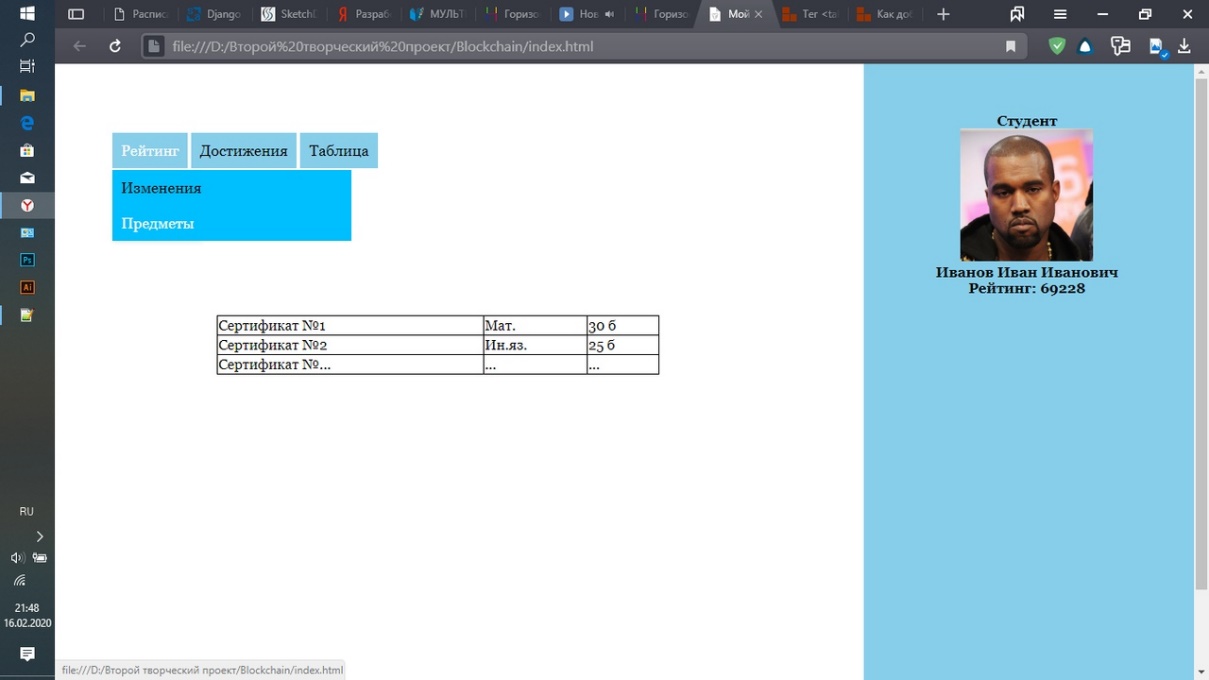
1. Разработка веб-приложения
   1. Разработка интерфейса приложения

Для создания веб-приложения используются стандартные методы верстки сайтов с использованием CSS – стилей, а также языка программирования PHP.

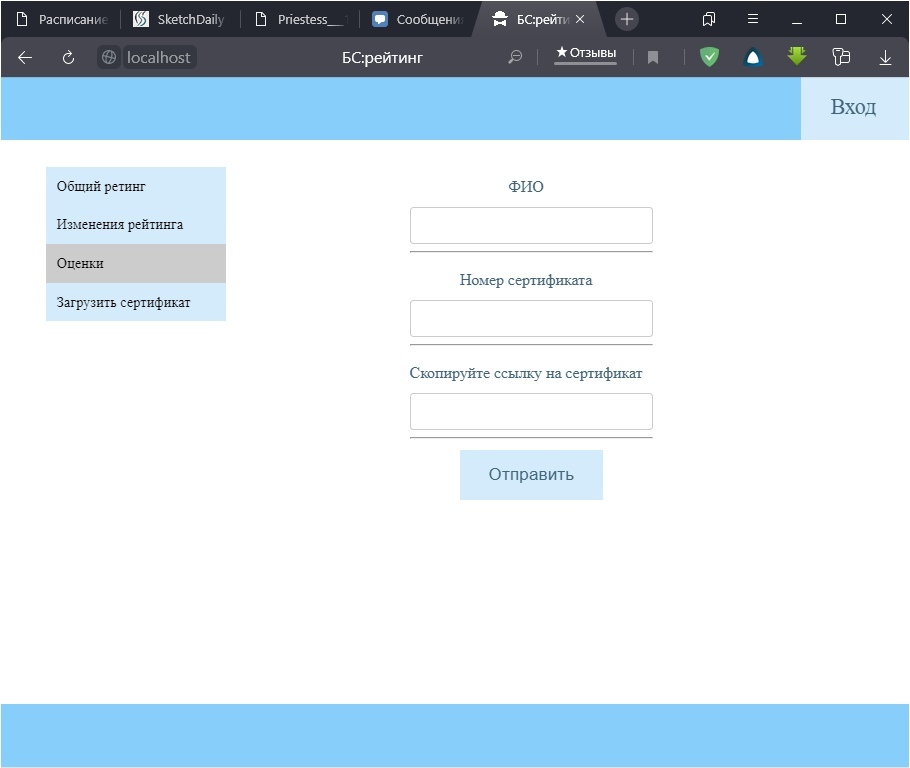
CSS — формальный язык описания внешнего вида документа, написанного с использованием языка разметки. CSS используется создателями веб-страниц для задания цветов, шрифтов, стилей, расположения отдельных блоков и других аспектов представления внешнего вида этих веб-страниц. Основной целью разработки CSS являлось отделение описания логической структуры веб-страницы (которое производится с помощью HTML или других языков разметки) от описания внешнего вида этой веб-страницы (которое теперь производится с помощью формального языка CSS). Такое разделение может увеличить доступность документа, предоставить большую гибкость и возможность управления его представлением, а также уменьшить сложность и повторяемость в структурном содержимом [8].

Кроме того, CSS позволяет представлять один и тот же документ в различных стилях или методах вывода, таких как экранное представление, печатное представление, чтение голосом (специальным голосовым браузером или программой чтения с экрана).

Начальный вариант (Рисунок 4), а также активация сертификата (Рисунок 5)



1. Веб-приложение в «сыром» виде



1. Активация сертификата

PHP: препроцессор гипертекста — скриптовый язык общего назначения, интенсивно применяемый для разработки веб-приложений. В настоящее время поддерживается подавляющим большинством хостинг-провайдеров и является одним из лидеров среди языков, применяющихся для создания динамических веб-сайтов. Синтаксис PHP подобен синтаксису языка Си. PHP является языком программирования с динамической типизацией, не требующим указания типа при объявлении переменных, равно как и самого объявления переменных. Язык автоматически поддерживает HTTP Cookies в соответствии с стандартами Netscape. Это позволяет проводить установку и чтение небольших сегментов данных на стороне клиента [9].

Иллюстрации дизайна веб-приложения (см. Приложение Б)

1. Объединение результатов работы
   1. Платформа NodeJS

Для объединения результатов работы над проектом был выбран Node или Node.js — программная платформа, основанная на движке V8 (транслирующем JavaScript в машинный код), превращающая JavaScript из узкоспециализированного языка в язык общего назначения. Node.js добавляет возможность JavaScript взаимодействовать с устройствами ввода-вывода через свой API (написанный на C++), подключать другие внешние библиотеки, написанные на разных языках, обеспечивая вызовы к ним из JavaScript-кода. Node.js применяется преимущественно на сервере, выполняя роль веб-сервера, но есть возможность разрабатывать на Node.js и десктопные оконные приложения (при помощи NW.js, AppJS или Electron для Linux, Windows и macOS) и даже программировать микроконтроллеры (например, tessel, low.js и espruino). В основе Node.js лежит событийно-ориентированное и асинхронное (или реактивное) программирование с неблокирующим вводом/выводом [10]. В состав Node.js входит собственный установщик пакетов npm. Установка производится при помощи команды: npm install <packagename>.

Листинг главного файла веб-приложения (см. Приложение В)

* 1. Фреймворк Express

Express используется для разработки приложений достаточно давно и благодаря своей стабильности прочно занимает позицию одного из самых популярных фреймворков Node.js. Для этого фреймворка существует большое количество подробных инструкции и описаний, которые составлены разработчиками, проверившими его эффективность на практике. Поэтому именно с Express рекомендуется начинать работу.

Основная особенность этого фреймворка заключается в том, что для Express характерен небольшой объем базового функционала. Все остальные нужные функции нужно будет добирать за счет внешних модулей. По сути, Express в чистом виде – это сервер и у него может не быть ни одного модуля.

Благодаря такому минимализму разработчик изначально получает в свое распоряжение легкий и быстрый инструмент, который он может расширять и развивать. При этом немаловажно, что выбор модулей для Express не связан ни с какими ограничениями: ни с количественными, ни с функциональными.

В результате, этот фреймворк обеспечивает разработчику возможность решать любые задачи, не ограничивая его при этом в выборе средств.

С одной стороны, не может не радовать тот факт, что отсутствие готовых универсальных решений фактически означает, что каждое создаваемое приложение будет уникальным.

С другой стороны, разработчику нужно самостоятельно отбирать и организовывать модули, а это предполагает большой объем работы и соответственно, требует от разработчика больше времени и усилий [11].

ПЛЮСЫ: простота, гибкость, хорошая масштабируемость, развитое сообщество, подробная документация, широкий выбор подключаемых модулей

МИНУСЫ: большой объем ручной работы, используется устаревший подход callbacks функций

* 1. Шаблонизатор EJS

EJS или встроенный шаблон Javascript — это шаблонизатор, используемый Node.js. Шаблонный движок помогает создавать HTML-шаблон с минимальным кодом. Кроме того, он может вводить данные в шаблон HTML на стороне клиента и создавать окончательный HTML. EJS — это простой язык шаблонов, который используется для генерации разметки HTML с простым JavaScript. Это также помогает встраивать JavaScript в HTML-страницы.

Поведение EJS по умолчанию заключается в том, что он просматривает папку 'views' для шаблонов для рендеринга. Он используется для встраивания динамического содержимого в страницу и для встраивания обычного JavaScript [12].

Результат проделанной работы

Итогом проделанной нами работы стало децентрализованное приложение – веб-приложение с поддержкой blockchain-технологий. Смарт-контракты тесно связаны с финансовыми транзакциями и имеют ограничения по числу участников в момент времени. Децентрализованные приложения расширяют возможности этой технологии.  
В децентрализованных приложения имеется возможность участия неограниченного числа сторон со всего рынка. В приложении имеется две роли: учащиеся и администратор.

Администратору выдаются права по распоряжению баллов в системе (рисунок 6 – 7), добавление дипломов в систему.

Пользователям (учащимся) выдаются права по активации дипломов и просмотр баллов.

Таблица 4 – Стоимость разработки

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| № | Название этапа разработки | Количество человек, задействованных в разработке этапа | Количество часов, потраченных на этап | Цена часа работы (руб.) | Стоимость разработки |
| 1 | Изучение смарт контрактов | 3 | 6 | 1500 | 9000 |
| 2 | Реализовать смарт контракт | 3 | 14 | 1100 | 15400 |
| 3 | Создать способ авторизации | 2 | 6 | 400 | 1600 |
| 4 | Создать веб-приложение | 2 | 14 | 400 | 5600 |
| 5 | Объединить смарт контракт с веб-приложением | 3 | 14 | 600 | 8400 |
| 6 | Исправить ошибки | 3 | 8 | 300 | 2400 |
|  | ИТОГО |  |  |  | 42400 |

Таблица 5– Стоимость не вошедшая в процесс разработки

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| № | Вид деятельности | Объем работы | Заработная плата на рынке труда | Общая стоимость |
| 1 | Дизайнер | 11 макетов | 1100 |  |
| 2 | Аналитик | Конкуренты, выбор ЯП, способ реализации, целевая аудитория | 8800 |  |
| 3 | Тестировщик | Тест веб-приложения | 900 |  |
|  | ИТОГО |  |  | 10800 |

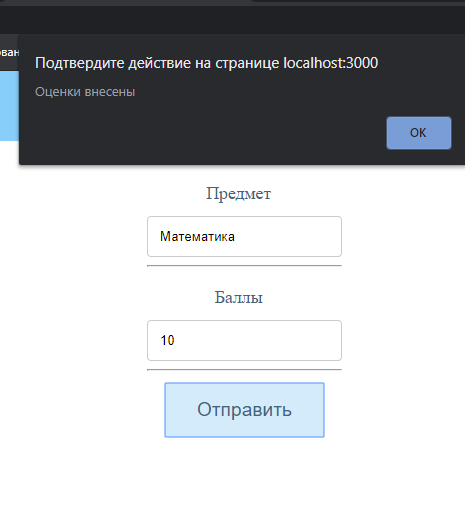


Рисунок 6 – Занесение баллов

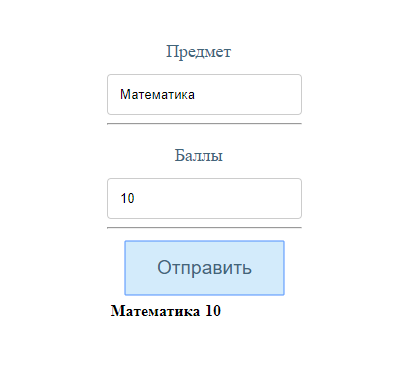


Рисунок 7 – Изменение баллов по Математика на 10

В это время все события попадают в блокчейн, затем обрабатываются смартконтрактом. Это можно отследить с помощью сервера Ganache (рисунок 8).

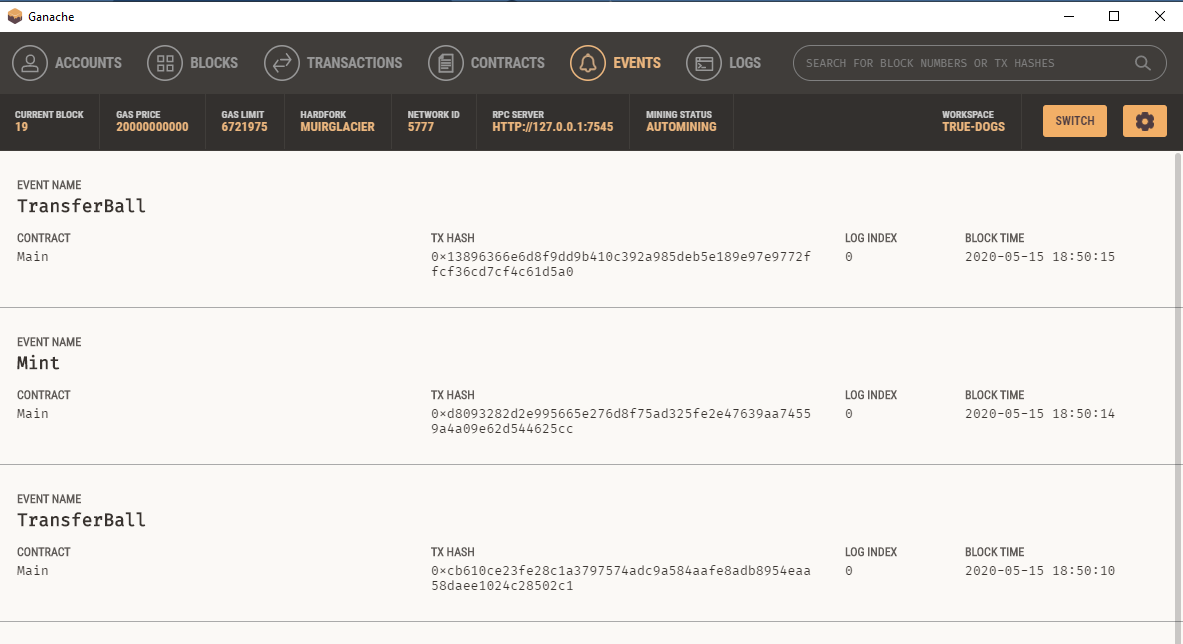


Рисунок 8 – TransferBall – операция занесения баллов

Заключение

Наша работа в рамках проекта позволила нам рассмотреть некоторые аспекты blockchain-технологий и web-разработки. Все началось с того, что мы начали выбирать язык программирования для реализации blockchain. Выбор был между python и javascript – в итоге мы выбрали javascript, поскольку нам показалось, что изучить javascript легче и быстрее (это для нас был довольно серьезный критерий). Когда мы хотели соединить смарт-контракт с браузером перед нами возникла проблема как это сделать. Сначала мы хотели сделать single page application, но потом нам пришлось отказаться от SPA и нам пришлось делать децентрализованное приложение. Это оказалось отличным решением так как децентрализованные приложения ориентированы на blockchain. Что на счет актуальности DApp – его можно монетизировать двумя способами: майнинг и выполнение тех или иных действий в приложении с оплатой в токенах. Цена токена зависит от его привлекательности (востребованности). Традиционные blockchain-приложения (централизованные) практически не интересны участникам крипто-рынка, нежели децентрализованные. Это достигается за счет трех факторов: востребованности, инвестиционной привлекательности нативного токена и беспрецедентной безопасности.

У нас возникали проблемы с документацией на протяжении выполнения проекта – она была приоритетно на английском языке, что, во-первых, затрудняло полное понимание информации и во-вторых замедляло процесс разработки.

Также серьезной проблемой перед нашей командой был уход некоторых участников – изначально нас было 8, далее некоторые люди просто перешли в другие проекты, а некоторые были отчислены. Количество оставшихся участников – 4, что является минимальным порогом для существования команды. Конечно, при уменьшении количества участников обязанности распределились на оставшихся членов команды. Это заставило нас пересмотреть план разработки и научится более организованно расходовать время.

Список использованных источников

1. Что такое блокчейн? Расскажем простыми словами // Coinspot URL: https://coinspot.io/beginners/chto-takoe-blokchejn-rasskazhem-prostymi-slovami/ (дата обращения: 17.12.2019).
2. Блокчейн: как он работает, и почему эта технология изменит мир // Хабр URL: https://habr.com/ru/company/iticapital/blog/340992/ (дата обращения: 17.12.2019).
3. 1С:Университет - Сравнение версий // 1С:Отраслевые и специализированные решения фирмы "1С" URL: https://solutions.1c.ru/catalog/university/comparison (дата обращения: 17.12.2019).
4. Что такое язык программирования смарт-контрактов Solidity // Digiterode URL: http://digitrode.ru/articles/1069-chto-takoe-yazyk-programmirovaniya-smart-kontraktov-solidity.html (дата обращения: 20.02.2020).
5. Beautiful Soup // Wiki.python URL: http://wiki.python.su/Документации/BeautifulSoup (дата обращения: 20.02.2020).
6. Что такое Selenium? // Хабр URL: https://habr.com/ru/post/152653/ (дата обращения: 20.02.2020).
7. Использование модуля Requests в Python // EnvatoTuts+ URL: https://code.tutsplus.com/ru/tutorials/using-the-requests-module-in-python--cms-28204 (дата обращения: 20.02.2020).
8. CSS // Википедия URL: https://ru.wikipedia.org/wiki/CSS (дата обращения: 26.04.2020).
9. PHP // Википедия URL: https://ru.wikipedia.org/wiki/PHP (дата обращения: 26.04.2020).
10. NodeJS // Википедия URL: https://ru.wikipedia.org/wiki/Node.js (дата обращения 01.05.2020
11. Выбор лучшего Node.js фреймворка: Express, Koa или Sails // Umbrella IT URL: https://umbrellait.com/ru/blog/choosing-the-best-nodejs-framework/ (дата обращения 01.05.2020)
12. Используйте EJS как шаблонизатор в NodeJs // портал информатики для гиков URL: http://espressocode.top/use-ejs-as-template-engine-in-node-js/ (дата обращение 01.05.2020)

**ПРИЛОЖЕНИЯ**

Приложение А

Листинг смарт контракта

pragma solidity >=0.5.0 <0.6.0;

//Контракт для обозначения создателя (администратора) и студентов

contract Ownable {

//Адрес создателя

address private owner;

//Конструктор контракта, присваивающий адрес создателя

constructor() public {

owner = msg.sender;

}

//Модификатор доступа (только для создателя)

modifier onlyOwner() {

require(msg.sender == owner);

\_;

}

//Модификатор доступа для всех, кроме создателя

modifier dontOwner(){

require(msg.sender != owner);

\_;

}

}

//Библиотека для безопасных математических операций с uint/uint256

library SafeMath {

//Умножение

function mul(uint256 a, uint256 b) internal pure returns (uint256) {

if (a == 0) {

return 0;

}

uint256 c = a \* b;

assert(c / a == b);

return c;

}

Продолжение приложения А

//Деление

function div(uint256 a, uint256 b) internal pure returns (uint256) {

uint256 c = a / b;

return c;

}

//Вычитание

function sub(uint256 a, uint256 b) internal pure returns (uint256) {

assert(b <= a);

return a - b; }

//Сложение

function add(uint256 a, uint256 b) internal pure returns (uint256) {

uint256 c = a + b;

assert(c >= a);

return c;

}

}

//Контракт для обозначения уникального диплома или грамоты

contract Diplom is Ownable {

using SafeMath for uint256;

//Переменная для хранения кол-ва дипломов у определенного студента

mapping(address => uint) public OwnerDiplomsCount;

//Переменная проверки наличия диплома у определенного студента

mapping(uint => address) DiplomsToOwner;

//Структура диплома

struct diplom {

//Уникальный номер

uint number;

//Владелец

address owner;

}

//Массив дипломов для хранения

diplom [] diploms;

Продолжение приложения А

//Функция вывода кол-ва дипломов у определенного студента

function balanceOf(address owner) external view returns (uint) {

return OwnerDiplomsCount[owner];

}

//Функция вывода владельца диплома по уникальному номеру

function ownerOf(uint tokenId) external view returns (address) {

return DiplomsToOwner[tokenId];

}

//Функция передачи диплома определенному студенту

function transferDiplom(address to, uint tokenId) internal {

OwnerDiplomsCount[to] = OwnerDiplomsCount[to].add(1);

OwnerDiplomsCount[msg.sender] = OwnerDiplomsCount[msg.sender].sub(1);

DiplomsToOwner[tokenId] = to;

emit TransferDip(msg.sender, to, tokenId);

}

//Функция занесения дипломов

function setDiploms(uint tokenId) onlyOwner public {

diploms.push(diplom(tokenId, msg.sender));

DiplomsToOwner[tokenId] = msg.sender;

OwnerDiplomsCount[msg.sender] = OwnerDiplomsCount[msg.sender].add(1);

emit SetDiploms(msg.sender, tokenId);

}

//Событие отправки диплома

event TransferDip(address indexed \_from, address indexed to, uint tokenId);

//Событие занесение диплома

event SetDiploms(address indexed \_from, uint tokenId);

}

//Контракт для баллов

contract Balls is Ownable {

using SafeMath for uint256;

//Общее кол-во баллов в системе

uint public totalSupply = 0;

Продолжение приложения А

//Переменная для хранения кол-во учебных баллов у определенного студента

mapping(address => uint) balances;

//Переменная для хранения суммы баллов

uint internal results;

//Функция вывода кол-ва баллов у определенного студента

function balanceOf(address owner) external view returns (uint balance) {

return balances[owner];

}

//Функция передачи баллов определенному студенту

function transferBalls(address to, uint value) internal returns (bool success) {

if (balances[msg.sender] >= value && to != msg.sender){

balances[msg.sender] -= value;

balances[to] += value;

emit TransferBall(msg.sender, to, value);

return true;

} else {

return false;

}

}

//Событие передачи баллов определенному студенту

event TransferBall(address indexed \_from, address indexed \_to, uint \_value);

}

//Основной контракт

contract Main is Diplom, Balls{

//Конструктор контракта для передачи общего кол-ва баллов администратору

constructor(uint value) public{

balances[msg.sender] = value;

totalSupply = value;

emit Mint(msg.sender, value);

}

//Переменная для хранения суммы баллов

uint private results;

Окончание приложения А

//Функция начисления баллов за заслуги

function giveBall(address to) public onlyOwner {

uint count = OwnerDiplomsCount[to];

results = results / 10;

count = count\*\*2;

results = results + count;

transferBalls(to, results);

}

//Функция проверки диплома

function giveDiploms(address to, uint tokenId) public dontOwner {

for (uint i = 0; i < diploms.length; i++){

if(diploms[i].number == tokenId){

transferDiplom(to, tokenId);

OwnerDiplomsCount[to]++;

}

}

}

//Функция передачи баллов в блокчейн

function givetData(address owner, uint grade) public dontOwner {

{

results = grade;

emit Data(owner, grade);

}

}

//Событие о передачи оценок в блокчейн

event Data(address indexed owner, uint grade);

//Событие о создании кол-ва всех баллов

event Mint(address indexed \_from, uint \_value);

}

Приложение Б

Иллюстрации дизайна сайта



Рисунок Б1 - стартовая страница сайта

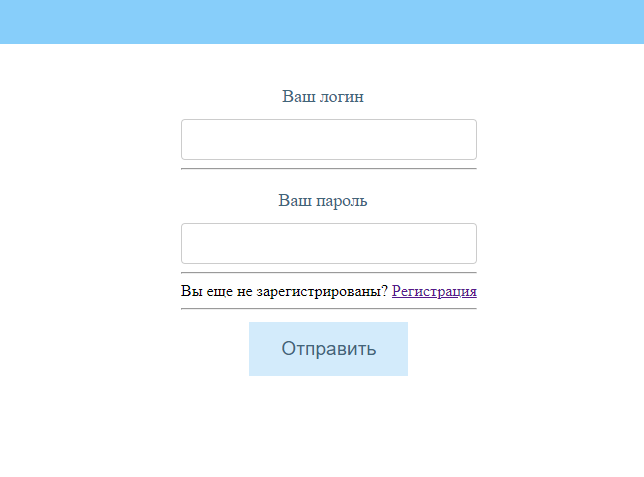


Рисунок Б2 - форма для авторизации

Продолжение приложения Б

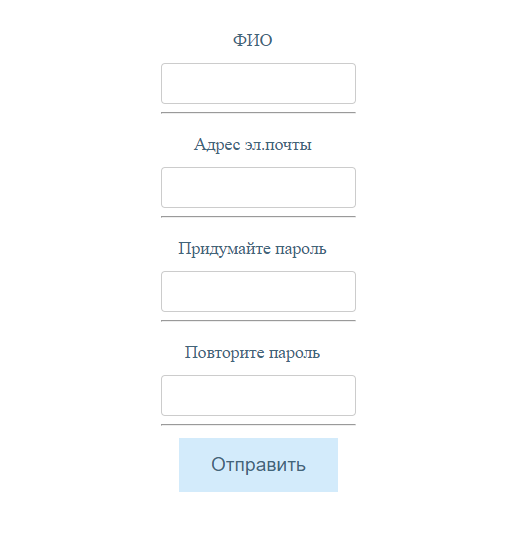


Рисунок Б3 - Форма для регистрации

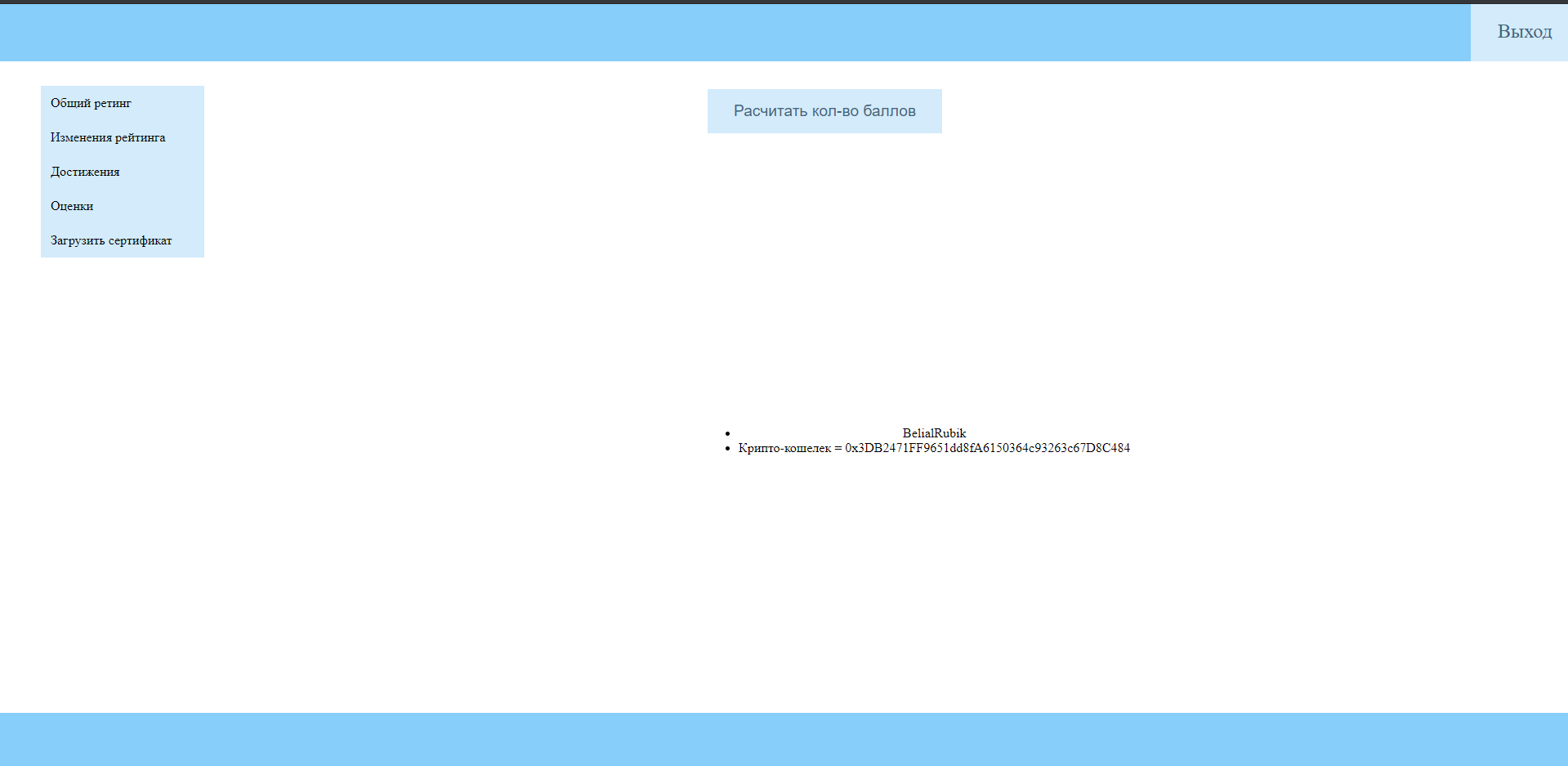


Рисунок Б4 - Личный кабинет

Окончание приложения Б

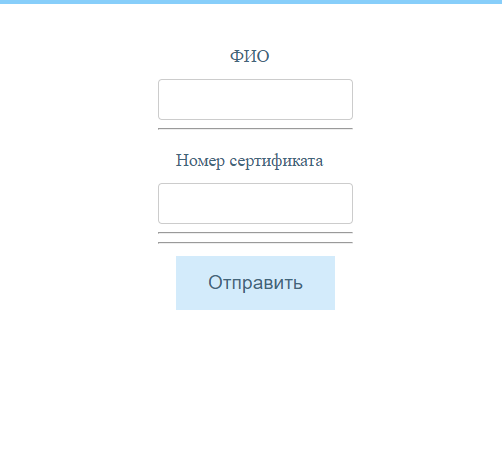


Рисунок Б5 - Форма для активации сертификата

Приложение В

Листинг главного файла веб-приложения

const express = require('express');

const bodyParser = require('body-parser');

const path = require('path')

const mongoose = require('mongoose');

const session = require('express-session')

const MongoStore = require('connect-mongo')(session)

const staticAsset = require('static-asset')

const config = require('./config');

const routes = require('./routes')

const app = express();

app.use(

session({

secret: config.SESSION\_SECRET,

resave: true,

saveUninitialized: false,

store: new MongoStore({

mongooseConnection:mongoose.connection

})

})

)

// database

mongoose.Promise = global.Promise;

mongoose.set('debug', config.IS\_PRODUCTION);

mongoose.set('useUnifiedTopology', true)

mongoose.connection

.on('error', error => console.log(error))

.on('close', () => console.log('Database connection closed.'))

.once('open', () => {

const info = mongoose.connections[0];

console.log(`Connected to ${info.host}:${info.port}/${info.name}`);

});

Продолжение приложения В

mongoose.connect(config.MONGO\_URL, {useNewUrlParser: true, useCreateIndex: true});

app.use(staticAsset(path.join(\_\_dirname, 'public')))

app.use('/public', express.static(path.join('public')))

app.use(

'/scripts',

express.static(path.join(\_\_dirname, 'node\_modules', 'jquery', 'dist'))

);

app.set('view engine', 'ejs');

app.use(bodyParser.json());

app.use(bodyParser.urlencoded({ extended: true }));

app.get('/', (req, res) => res.render('index'));

app.get('/login', (req, res) => res.render('login'));

app.get('/registr', (req, res) => res.render('registr'));

app.get('/accaunt' , (req, res) => {

const id = req.session.userID;

const login = req.session.userLogin

const token = req.session.userToken

res.render('accaunt', {

user: {

id,

login,

token

}

})

})

app.get('/download' , (req, res) => {

const id = req.session.userID;

const login = req.session.userLogin

const token = req.session.userToken

Продолжение приложения В

res.render('download', {

user: {

id,

login,

token

}

})

})

app.get('/change' , (req, res) => {

const id = req.session.userID;

const login = req.session.userLogin

const token = req.session.userToken

res.render('change', {

user: {

id,

login,

token

}

})

})

app.get('/predmet', (req, res) => {

const id = req.session.userID;

const login = req.session.userLogin

const token = req.session.userToken

res.render('predmet', {

user: {

id,

login,

token

}

Окончание приложения В

})

})

-

app.get('/achiv' , (req, res) => {

const id = req.session.userID;

const login = req.session.userLogin

const token = req.session.userToken

res.render('achiv', {

user: {

id,

login,

token

}

})

})

app.use('/api/auth', routes.auth);

app.use('/api/main', routes.main)

app.listen(config.PORT, () =>

console.log(`Example app listening localhost:${config.PORT}`)

);

module.exports = app