VoteChain

Децентрализованная система онлайн-голосований, основанная на блокчейне

Научный руководитель

Калинин Антон Игоревич, приглашенный преподаватель факультета компьютерных наук; ведущий программист центра трансфера и управления социально-экономической информацией

Автор

Беликова Мария, студентка БПИ223

Введение в проблематику. Для кого?

Корпорации

- собрания акционеров
- советы директоров
- внутренние голосования сотрудников

Политика

- выборы президентов
- выборы мэров
- референдумы

Социология и наука в целом

- социальные исследования
- психологические опросы

Коммерция

- клиентские исследования
- рейтинг-опросы

Введение в проблематику. Какие проблемы?

Анонимность

Важно, чтобы **голос** и **тот, кто проголосовал** не были связаны, но при этом была четкая связь “1 человек – 1 голос”

Непредвзятость

Голоса должны быть **подсчитаны честно и независимо**

Проверяемость

Результаты и сам факт отдачи голосов должны быть **проверяемы**

Безопасность

Голоса должны быть **не подделываемыми**, и не должны быть доступны в раскрытом виде никому

Предметная область

Создатель
Создает и управляет голосованием, дает права на голосование и наблюдение

Голосователь
Отдает свой голос в конкретном голосовании

Наблюдатель
Наблюдает за голосованием – то есть имеет возможность убедиться, что весь процесс корректен

Голосование
Процесс, дающий честно, анонимно и безопасно узнать мнения голосователей по конкретному вопросу с детерминированными вариантами ответов

Голос
Конкретный выбор конкретного участника в конкретном голосовании (в зависимости от контекста может включать в себя метаданные)

Анализ существующих решений

	Анонимность	Непредвзятость	Проверяемость	Безопасность	Открытость	Стоимость
ДЭГ РФ	🟡	🟡	🟡	🟢	🔴	🔴
Helios	🟢	🟢	🟢	🟢	🟡	🟢
Voatz	🟡	🟡	🟢	🟢	🟡	🔴
VoteChain	🟢	🟢	🟢	🟢	🟢	🟢

Цель

Разработать систему с пользовательским интерфейсом, которая обеспечивает проведение безопасных, анонимных, проверяемых голосований

Задачи

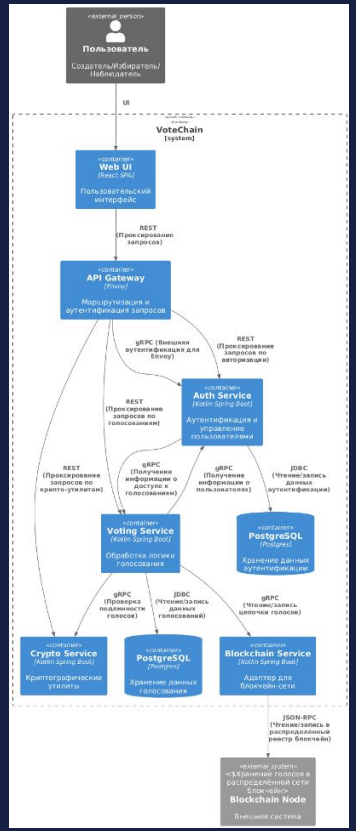
Реализация **функциональных** требований:

FT1: Аутентификация
FT2: CRUD голосований
FT3: Управление/получение прав голосования, наблюдения
FT4: Голосование/получение результатов
FT5: Наблюдение

Реализация **нефункциональных** требований:

NFT1: Анонимность
NFT2: Безопасность
NFT3: Проверяемость
NFT4: Масштабируемость

Общая архитектура. С высоты птичьего полета



Web UI

Веб приложение, пользовательский интерфейс

API Gateway

Управление REST-запросами к бэкендовым микросервисам

Auth-Service

Управление аутентификацией

Voting-Service

Основная логика голосований и прав

Crypto-Service

Сервис со всей криптографической логикой

Blockchain-Service

Адаптер к блокчейн-сети

Blockchain Node

Блокчейн-сеть для хранения голосов

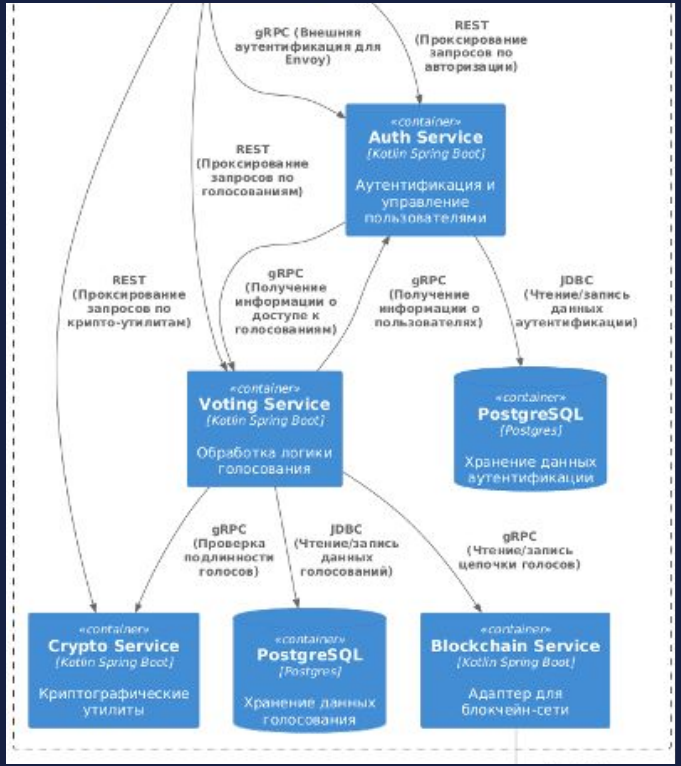
Микросервисы. Почему?

Реализация NFT4 – масштабируемости

Микросервисы позволяют масштабировать только определенные части – каждая сфера имеет свой пик

Косвенная реализация всех остальных требований

Каждый микросервис инкапсулирует свою доменную зону



API–Gateway. Envoy. Почему?

Реализация FT1 – аутентификации

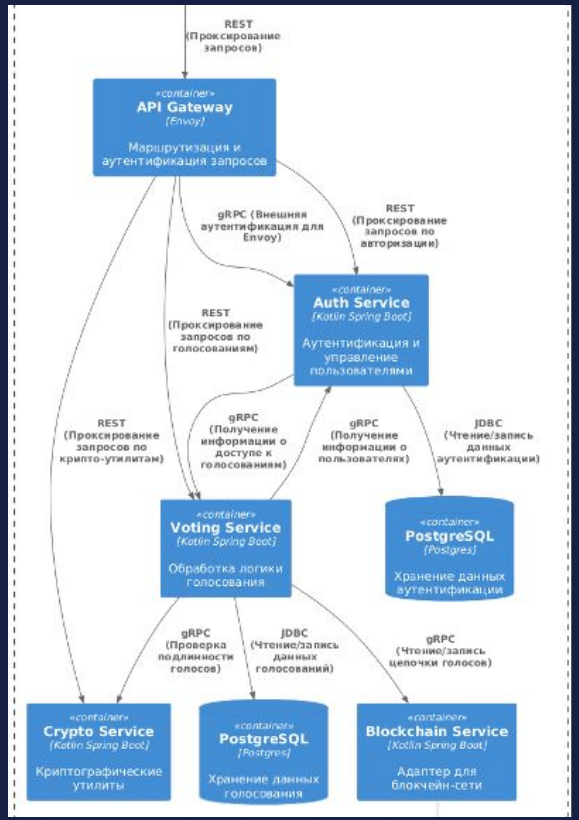
Envoy отправляет ВСЕ запросы в Auth-Service, благодаря чему определяется permission, а также в заголовки запроса добавляется клиентский ID – для дальнейшего использования другими микросервисами

Реализация NFT2 – безопасности

У клиента отсутствует любая уязвимая нагрузка, кроме Bearer-токена. Микросервисы не знают ничего об аутентификации, кроме самого нужного им клиентского кода

Реализация NFT4 – масштабируемости

Является маршрутизатором для микросервисов, точкой входа к ним со стороны клиента



Криптография. Как?

Основой криптографический пайплайн – реализация FT4, NFT1 – NFT3

1. Клиент запрашивает у сервера **публичный ключ PK** для того, чтобы использовать его в дальнейших криптографических манипуляциях;

2. Клиент шифрует голос (идентификатор ответа, выбранного пользователем – id) и получает **encrypted(id)**;

3. Клиент слепит полученный на шаге 2 encrypted(id) и получает **blinded(encrypted(id))**;

4. Клиент отправляет **blinded(encrypted(id))** на сервер;

5. Сервер подписывает заслепленное сообщение, не видя содержимого, и возвращает клиенту **signed(blinded(encrypted(id)))**;

6. Клиент расслепляет сообщение – и получает валидную подпись на собственное сообщение, **signed(encrypted(id))**;

7. Клиент отправляет на сервер **signed(encrypted(id))** и **(encrypted(id)**;

8. Сервер проверяет подпись **signed(encrypted(id))** с помощью **приватного ключа pK** (он убеждается, что подпись настоящая, и что голос пришёл от уполномоченного). При этом важно, что он не знает, какой именно это был голос. Далее сервер сохраняет **signed(encrypted(id))** и **(encrypted(id)** в блокчейн;

9. При подсчете голосов (encrypted(id)) дешифруются с помощью **приватного ключа pK** и получается **id**, но в них уже нет связи с конкретным пользователем (чем обеспечивается анонимность).

Криптография. Почему?

Слепая подпись

- сама по себе слепая подпись позволяет серверу **подписывать сообщения, не зная их содержимого**, что критически важно для обеспечения **анонимности**
- RSA-PSS обеспечивает высокий уровень безопасности благодаря вероятностному характеру подписи (каждая подпись уникальна даже для одного и того же сообщения). Этот алгоритм предназначен **специально для подписи**

Шифрование RSA-OAEP с SHA-256

- в отличие от классического RSA, **OAEP** – это алгоритм, специально **сделанный именно для шифрования**: он обеспечивает семантическую безопасность и устойчивость к различным атакам
- имеет куда более **широкую поддержку** в различных криптографических библиотеках, что помогло при реализации веб-клиента

Блокчейн. Что это?

Блокчейн-цепочка

Блок 1

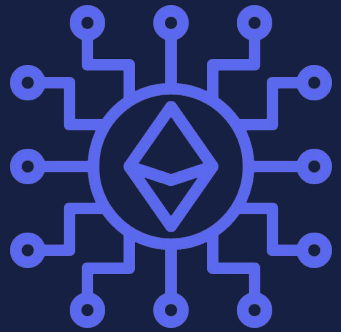
- время
- хэш
- зашифрованный голос
- подпись
- уникальный токен

Блок 2

- время
- хэш
- зашифрованный голос
- подпись
- уникальный токен

Блокчейн –

это распределенный цифровой реестр (БД), где записи хранятся в связанных друг с другом блоках. Каждый блок защищён криптографией, что почти исключает возможность подделки или изменения данных. Блокчейн характеризует работа многих участников (децентрализация) и криптография.



Блокчейн. Как и почему?

Примерный пайплайн

1. Пользователь отправляет **зашифрованный голос и метаданные** в сеть со смарт-контрактом
2. **Смарт-контракт на Ethereum** (локально — Ganache) сохраняет голос с меткой времени
3. Для каждого голоса создаётся уникальный **токен** — идентификатор записи
4. Все голоса **хранятся в блокчейне**, доступны для **проверки** по токену или идентификатору голосования (для наблюдателей)

Обеспечивает NFT2 – безопасность

- Данные хранятся в зашифрованных блоках, связанных друг с другом, поэтому **изменить или подделать запись** без изменения всех последующих блоков **невозможно**
- Управление сетью распределено между множеством независимых узлов, что **предотвращает взлом или манипуляции одним участником**

Обеспечивает NFT3 – проверяемость

- Все записи доступны для просмотра всем участникам сети — каждый может **проверить**, что данные не изменены и что голос был учтён

Технологический стэк. Какой и почему?






Web3J


Демо

Развитие. Что дальше?

- Внедрение подсчетов голосов в сам смарт-контракт
- Асинхронная обработка голосов (например, через Kafka)
- Расширенный функционал для юридических лиц
- Информационные блоки для пользователя о блокчейне и криптографии
- Внедрение zk-proof для голосов и для их подсчета
- Улучшение нативного доступа к самой Etherium-сети для проверки

Список источников

1. Обзор системы дистанционного электронного голосования ЦИК РФ https://habr.com/ru/companies/rostelecom/articles/518090/
2. How Blockchain Developers Create Transparent Voting Systems? https://www.rapidinnovation.io/post/how-blockchain-developers-create-transparent-voting-systems
3. Introduction to Smart Contracts https://ethereum.org/en/developers/docs/smart-contracts/
4. RSA Blind Signatures https://www.geeksforgeeks.org/rsa-blind-signatures/
5. Blind Signatures – an Overview https://www.sciencedirect.com/topics/computer-science/blind-signature
6. A Survey on Blind Digital Signatures https://nabihach.github.io/co685.pdf
7. Optimal asymmetric encryption padding https://en.wikipedia.org/wiki/Optimal_asymmetric_encryption_padding
8. Web3J Docs https://docs.web3j.io/4.11.0/

Спасибо!