Технічне завдання  
на розробку веб-додатка  
**SolSpy**

Версія документа: 2.5

Замовник:

Виконавець:

М. Запоріжжя  
11 серпня 2025

[1. Вступ 3](#_Toc205993356)

[1.1. Мета проєкту 3](#_Toc205993357)

[1.2. Цільова аудиторія 3](#_Toc205993358)

[1.3. Аналоги 3](#_Toc205993359)

[1. Функціональні вимоги 4](#_Toc205993360)

[2.1. Основні вимоги 4](#_Toc205993361)

[2.2. Головна сторінка 4](#_Toc205993362)

[2.3. Сторінка валідатора 6](#_Toc205993363)

[2.4. Особистий кабінет 7](#_Toc205993364)

[2.5. Сторінка порівняння 7](#_Toc205993365)

[2.6. Адміністративна частина 7](#_Toc205993366)

[2.7. Текстові сторінки 8](#_Toc205993367)

[2.8. Додаткові функції 8](#_Toc205993368)

[2. Технічні вимоги 9](#_Toc205993369)

[3.1. Frontend 9](#_Toc205993370)

[3.2. Backend 9](#_Toc205993371)

[3.3. Інтеграції 10](#_Toc205993372)

[3.3.1. PRC Solana 10](#_Toc205993373)

[3.3.2. Локалізація 10](#_Toc205993374)

[3.3.3. Discord API 10](#_Toc205993375)

[3. UI/UX дизайн 11](#_Toc205993376)

[4.1. Дизайн 11](#_Toc205993377)

[4.2. Логотип - SolSpy 11](#_Toc205993378)

[4.3. Карта сайта 12](#_Toc205993379)

[4.4. Wireframes 13](#_Toc205993380)

[4. Розширення 19](#_Toc205993381)

[5.1. Мобільний додаток 19](#_Toc205993382)

[5.2. Додаткові інтеграції 19](#_Toc205993383)

[5. Нефункціональні вимоги 20](#_Toc205993384)

[6. Умови підтримки та навчання 21](#_Toc205993385)

[7.1. Технічна підтримка 21](#_Toc205993386)

[7.2. Контент-підтримка 21](#_Toc205993387)

[7.2. Навчання 21](#_Toc205993388)

[7. Кошторис 22](#_Toc205993389)

[8. Етапи реалізації 23](#_Toc205993390)

[Додаток 1 - Ризики 24](#_Toc205993391)

[Додаток 2 - Налаштування хостингу та деплой 25](#_Toc205993392)

[Додаток 3 - Розрахунок нагород Awards 26](#_Toc205993393)

[Додаток 4 - Методи і розрахунки 27](#_Toc205993394)

[A.4.1. Номер поточної епохи, прогресс та час до кінця епохи 27](#_Toc205993395)

[A.4.2. Поточний курс SOL до USD 28](#_Toc205993396)

[A.4.3. Отримати список валідаторів 29](#_Toc205993397)

[A.4.4. Метадані валідатора 30](#_Toc205993398)

[A.4.5. Identity Key (обрізаний до 4+...+4) 34](#_Toc205993399)

[А.4.6. Vote Key та Withdrawer Key 34](#_Toc205993400)

[А.4.7. Статус (active / delinquent) 35](#_Toc205993401)

[А.4.8. TVC Score (місце за stake) 35](#_Toc205993402)

[А.4.9. Stake Pools (іконки пулів) 36](#_Toc205993403)

[А.4.10. Inflation Commission 36](#_Toc205993404)

[А.4.11. MEV Commission 37](#_Toc205993405)

[А.4.12. Uptime 38](#_Toc205993406)

[А.4.13. Client (з version) 38](#_Toc205993407)

[А.4.14. Статус SFDP 38](#_Toc205993408)

[А.4.15. Vote Rate 39](#_Toc205993409)

[А.4.16. Jito Score 40](#_Toc205993410)

[А.4.17. Leader Slots 40](#_Toc205993411)

[А.4.17. Time Next Slot 41](#_Toc205993412)

[А.4.18. Skipped Slots / Produced Slots 41](#_Toc205993413)

[А.4.19. Account Assets 42](#_Toc205993414)

[А.4.20. Сервер (CPU, RAM, SSD) 42](#_Toc205993415)

[А.4.21. Обчислення Spy Rank 43](#_Toc205993416)

[А.4.22. Налаштування RPC на ноді в mainnet 45](#_Toc205993417)

[А.4.23. Vote Credits 47](#_Toc205993418)

[А.4.24. Active Stake, Stake Changes (Pending Stake), Activating, Deactivating, Deligator’s Name 48](#_Toc205993419)

# Вступ

## 1.1. Мета проєкту

Створення інформативного, візуально-привабливого та інтерактивного веб-додатка (веб-сайта) для моніторингу актуальної інформації валідаторів Solana.

Підвіщення привабливості майннет ноди SolSpy для стейкерів Solana.

Користувачі повинні мати змоги фільтрувати інформацію, додавати до обраних, порівнювати, бачити історію змін, сповіщень, стейкінгу (Kiwi) та функції аналітики (графіки). Кожен валідатор може отримувати нагороди (зірочки).

## 1.2. Цільова аудиторія

* Валідатори
* Інвестори (стейкери)
* Аналітики блокчейн
* Розробники (власне API з JSON-відповідями)

## 1.3. Аналоги

* <https://stakewiz.com/>
* <https://solanabeach.io/>
* <https://www.validators.app/>
* <https://topvalidators.app/>
* [https://1000x.sh](https://1000x.sh/validators?fields=name,rank,credits,stake,stake_changes,leader_slots,version,city,commission,slots_skipped,score,stake_accounts_active,ip,country,asn,organization,slots_done&q=Biw%2Ccorgi%2C%D1%81esar%2Cromanh%2Cstarpaw%2Cneverback%2Ccoyote%2Csunshine%2CAndrewinua%2Ccryptovik%2Csolya%2Csolraindrops%2Cstakenode777%2Cvladika%2Cwinstake%2CSollana%2CSunshine%2CStakenode777%2CDarko%2Cmeissa%2CDarko%2Cpig%2Ccorgi%2Cpuerta%2Cmeann%2CSHP%2Cstealth%2CtheBest%2Cneverback) - параметри валідатора та вибір стовпців;
* <https://marinade.finance/> - графіки, як тут.
* Додаток [SolCircl](https://apps.apple.com/us/app/solcircl/id6502469087)
* <https://app.jpool.one/validators>

# Функціональні вимоги

## 2.1. Основні вимоги

* Актуалізація даних кожні 1 секунду (2 с або 5 с, 60 с, 5 хв, 15 хв, 30 хв) для TVC Score та Vote Credits. Якщо відбуваються зміни значень, то кожне нове значення виводити по черзі дефолтним або зеленим кольором. Якщо зміни не відбуваються протягом секунди, то виводити лише дефолтним кольором.
* Фільтрація валідаторів
* Сортування, змінення порядку стовбців
* Додавання валідатора до обраних (профіль користувача, localstorage)
* Порівняння валідаторів
* Стейкінг (перенаправлення на Kiwi)
* Вказання пулів кожного валідатора та сортування по пулам (від більшого)

## 2.2. Головна сторінка

* Кнопка “Stake With Us” і посилання на гаманець Kiwi (буде надано)
* Панель поточної епохи (epoch):
  + Epoch
  + Epoch progress
  + Час до кінця (Epoch time remaining)
* Поточний курс SOL/$
* Перемикач світлої/темної теми
* Перемикач мови – Англійська (за замовчуванням), Німецька, Українська. Якщо IP користувача в Україні, рашці або Білорусі, то мова автоматично перемикається на українську. Німецький IP – автоматично перемикається на німецьку мову. Всі інші IP – англійська. Перелік елементів, які треба відображати надає замовник до початку розробки Backend. Також замовник робить переклад всіх необхідних елементів і надає у вигляді файлу.
* Іконки:
  + Порівняти обраних валідаторів (без авторізації користувач має змогу порівняти тільки двох - зберігається в localstorage)
  + Додати до обраних (без авторизації користувач може додати до обраних не більше 5 - зберігається в localstorage)
  + Сповіщення сайту (для авторизованих користувачів)
* Кнопка Login / Register (попап-вікно)
* Банер-карусель з карточками топ-10 валідаторів (власний score на основі [Stakewiz](https://stakewiz.com/faq#faq-wizscore)) або обраними в адміністративній панелі)
* Банер-карусель з новинами з Discord (API-інтеграція) або із адмінки
* Пошук валідатора за ім’ям, identity key, vote key, withdrawer
* Таблиця валідаторів:
  + Колонки (приклад вибору колонок як на сайті [1000х](https://1000x.sh)):
    - Checkbox для групових дій (порівняти, додати до обраних, налаштувати сповіщення, додати до blacklist)
    - Spy Rank (додаток А.4.21)
    - Аватар (якщо є)
    - Назва/Ім’я та Identity Key (образзаний до 4 символа на початку і 4 символа на прикінці)
    - Статус (active/delink)
    - TVC Score (місце в поточну епоху)
    - Vote Credits (на wireframes цього нема, але ця колонка потрібна в таблиці або в попапі вибору колонок)
    - Active Stake (показувати в два рядки: перший рядок – значення active stake (стейк показувати, використовуючи тисячі та міліони – якщо стейк менше 100, то округляти до першої цифри після крапки, інакше – округляти до цілого), другий рядок - набір пулів у вигляді іконок, як на [validators.app](https://www.validators.app/validators?locale=en&network=mainnet)) – цього нема на wireframes, але треба зробити в таблиці, як обов’язкове поле
    - Inflation Commission
    - MEV Commission
    - Uptime
    - Client (з version)
    - Статус SFDP (none, pending, onboard, retired, rejected)
    - Location (country)
    - Awards (Додаток 3)
    - Vote Rate (як в додатку [SolCirl](https://play.google.com/store/apps/details?id=com.gateomega.solcircl))
    - Website
    - City
    - ASN
    - IP
    - Jito Score (30 епох - [місце в рангу Jito)](https://www.jito.network/stakenet/steward/)
  + Кнопки дій:
    - Відкрити сторінку валідатора
    - Додати до порівняння
    - Додати до обраних
    - Сповіщення (попап-вікно з табами Telegram/Email - для незареєстрованих користувачів доступні не всі сповіщення (Inf%, MEV%, Status) - зберігається в localstorage)
    - Stake (посилання на гаманець Kiwi, [наприклад](https://staking.kiwi/app/DHoZJqvvMGvAXw85Lmsob7YwQzFVisYg8HY4rt5BAj6M?cluster=mainnet-beta))
    - Заблокувати (не відображати в таблиці)
  + Візуальна підсвітка валідатора за певними умовами (наприклад, з адміністративної панелі)
  + Опції відображення (множинний вибір):
    - Всі
    - Тільки обрані
    - Тільки ті, що обрані для порівняння
    - Заблоковані
    - Не з рашки (не релізі це треба приховати, але в майбутньому треба мати можливість увімкнути)
    - Тільки з іменем
    - Тільки ті, що мають сайт
    - Тільки ті, що валідують (operated) більше року
    - Тільки ті, що мають inflation commission та MEV com. - 0
  + Колонки можна міняти містами (drug-&-drop) - порядок повинен зберігатись, додавати або прибирати необхідні (попап-вікно), міняти вигляд виводу даних (строчки або картки).

## 2.3. Сторінка валідатора

* Вся інформація, що відображається в таблиці на головній сторінці з повним її функціоналом, включно з полями, що не відображаються (якщо можна отримати доступ до певної інформації, наприклад, Uptime). Всюди поточний час користувача сайтом.
  + Сповіщення про зміну (зміна або оновлення)
  + Awards (Додаток 3)
* Active Stake і Stake Changes (різниця між Activating та Deactivating)
* LeaderSlots (поточна та наступна епоха), Time Next Slot, Skipped, Produced
* Account Assets (Identity, Vote, Withdriver)

<https://pro-api.solscan.io/pro-api-docs/v2.0> - як тут

* Мониторинг серверів:
  + Локалізація (ASN Lookup/GeoIP)
  + CPU (якщо доступно)
  + RAM (якщо доступно)
  + SSD (якщо доступно)
  + Візуалізація на карті
* Графіки (історичні дані за 30 епох):
  + Active Stake (гістограмма)
  + Skip Rate (гістограмма)
  + Uptime (кольорові кластери)
  + Commission Change (таблиця)
* Таблиця Stake By Delegators
* Експорт даних (CSV/JSON)

## 2.4. Особистий кабінет

* Реєстрація/авторизація (Email + 2FA / OAuth / Google, Apple)
* Таби:
  + Список обраних валідаторів у вигляді таблиці (як на головній сторінці з аналогічним функціоналом)
  + Список заблокованих валідаторів (blacklist)
* Налаштування сповіщень (Email/Telegram-бот)
  + Вибір, які сповіщення надсилати у Telegam-бот, а які на Email
  + Збереження Email-адреси
* Додаткові функції
  + Порівняння вибраних валідаторів
  + Експорт даних (CSV/JSON)

## 2.5. Сторінка порівняння

* Таблиця з порівнянням обраних валідаторів (можливість видалення валідатора з таблиці)
* Повинна бути можливість перетягування валідаторів
* Експорт даних
* Порівняння у вигляді графіків:
  + Uptime (лінійний графік)
  + Skip rate (лінійний графік)
  + Active Stake (лінійний графік)
  + Commission Change (лінійний графік)

## 2.6. Адміністративна частина

* Перегляд/редагування зареєстрованих користувачів
* Визначення топ-валідаторів (checkbox)
* Визначення яких валідаторів треба підсвічувати (візуально виділяти) в таблиці (checkbox або алгоритм)
* Налаштування сповіщень
  + Рекламні сповіщення в Telegram-бот
* Моніторинг/завантаження новин у карусель
* Статистика відвідувань
* Перегляд логів

## 2.7. Текстові сторінки

* About Project
* Security
* FAQ (акордеон з питаннями та відповідями)
* API (опис API)
* Contacts

Дані надає замовник на всіх мовах.

## 2.8. Додаткові функції

* Зберігання налаштувань користувача в браузері (localstorage)
* Підтримка мобільної версії (адаптивний дизайн)
* API

# Технічні вимоги

Розробник сам обирає технології, якими він буде користуватись у розробці. Рекомендовані (не обов’язкові) технології надані у розділі нижче. Розробка Frontend+Backend розпочинається тільки після погодження макету Figma.

## 3.1. Frontend

* Стек:
  + [React.js](http://react.js)
  + [Next.js](http://next.js)
  + [Node.js](http://node.js)
  + TypeScript
* Графіки:
  + Rechart
  + [Chart.js](http://chart.js)
  + [D3.js](http://d3.js)
  + або аналоги
* Стилі:
  + Tailwind CSS
  + або аналоги
* Адаптивність:
  + Підхід Mobile-first

## 3.2. Backend

* [Node.js](http://node.js) + Express або Python Fast API
* Cron/Worker для періодичного опитування RPC
* Кешування: Redis або PostgreSQL
* База даних: SQL або PostfreSQL
* API: REST + WebSocket

## 3.3. Інтеграції

### 3.3.1. PRC Solana

Інтеграція з RPC Solana реалізується завдяки запросам до власного RPC замовника, який розташований на окремому спеціально облаштованому під цю задачу сервері. RPC надає замовник, налаштовує - виконавець. Всі методи взаємодії з RPC Solana описані в офіційній документації - <https://solana.com/uk/docs/rpc/http>

Загальна інформація:

* getEpochInfo - поточна епоха, слоти
* getBlockHeight - висота блоку
* getVersion - версія Solana
* getLeaderSchedule - розклад лідерів
* getEpochSchedule - довжина епохи
* Сторонні сервіси (дивись додатки).

Інформація про валідатора:

* getVoteAccounts - список валідаторів з комісією, lastVote, rootSlot
* getInflationReward - нагороди валідаторів
* getStakeActivation - активність стейків (активний/неактивний стейк)
* getBlockProduction - активність валідатора
* getIdentity - з блокчейн або Jito
* Сторонні сервіси (дивись додатки).

### 3.3.2. Локалізація

Геолокація серверів визначається через IP-аналіз або ручне внесення.

* IP валідатора - GeoIP (через сторонні API)
* або аналог

### 3.3.3. Discord API

* Парсінг Discord через API або RSS

# UI/UX дизайн

## 4.1. Дизайн

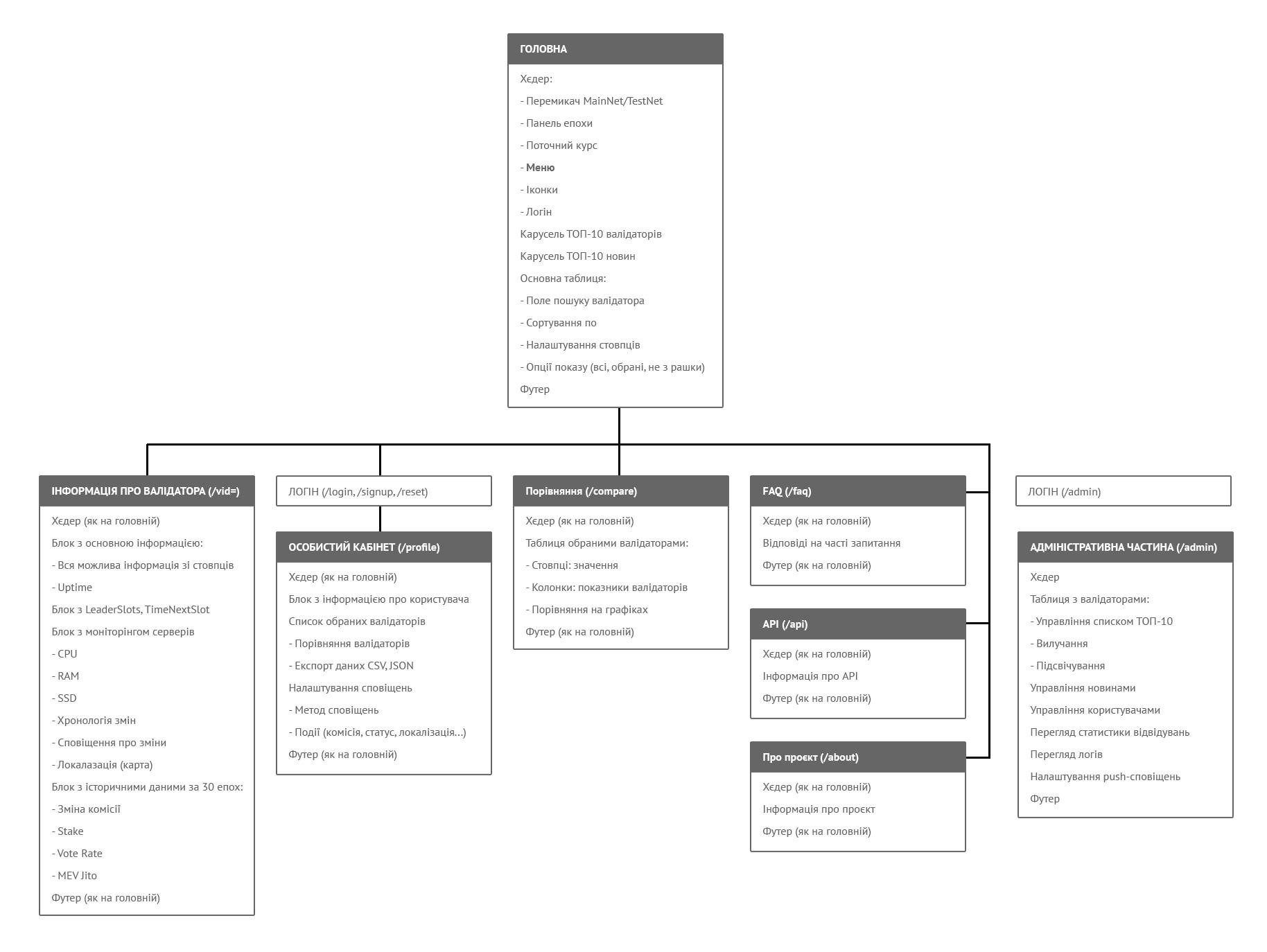
Дизайн надає замовник у вигляді посилання на проєкт Figma, який містить десктопну та мобільну версію для світлої та темної тем. Має елементи навігації, анімації та опису необхідних елементів дизайну (шрифти, кольорова схема тощо).

Додатково замовник надає всі необхідні іконки (елементи інтерфейса, іконки pool) у форматі SVG та PNG.

## 4.2. Логотип - SolSpy

Логотип надає замовник у вигляді PNG та SVG файлу в нормальних та інвертованих кольорах.

## 4.3. Карта сайта

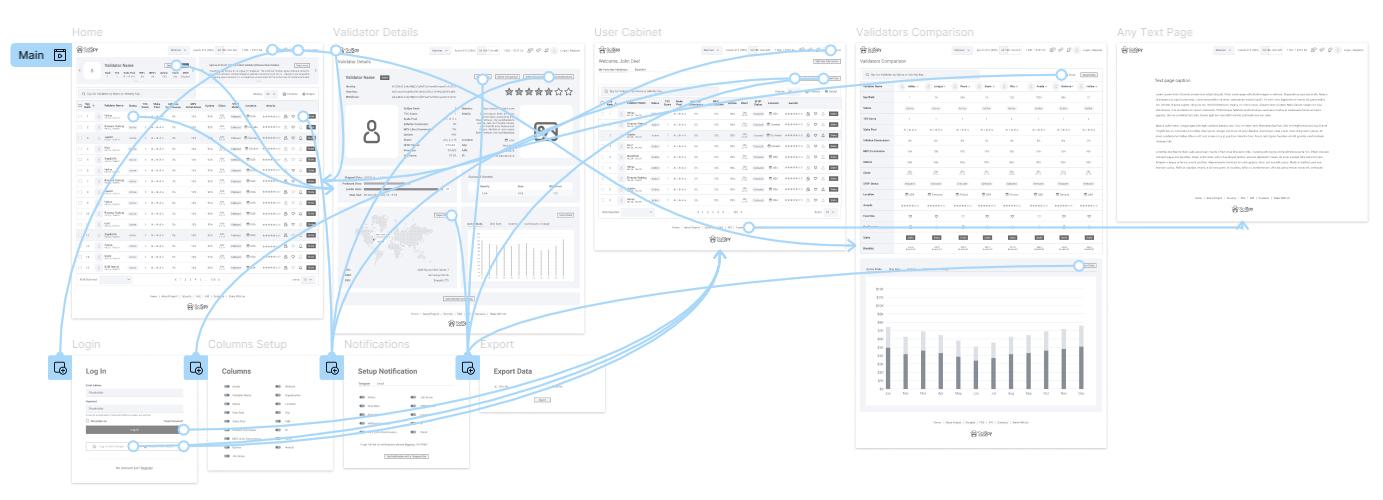


## 

## 4.4. Wireframes

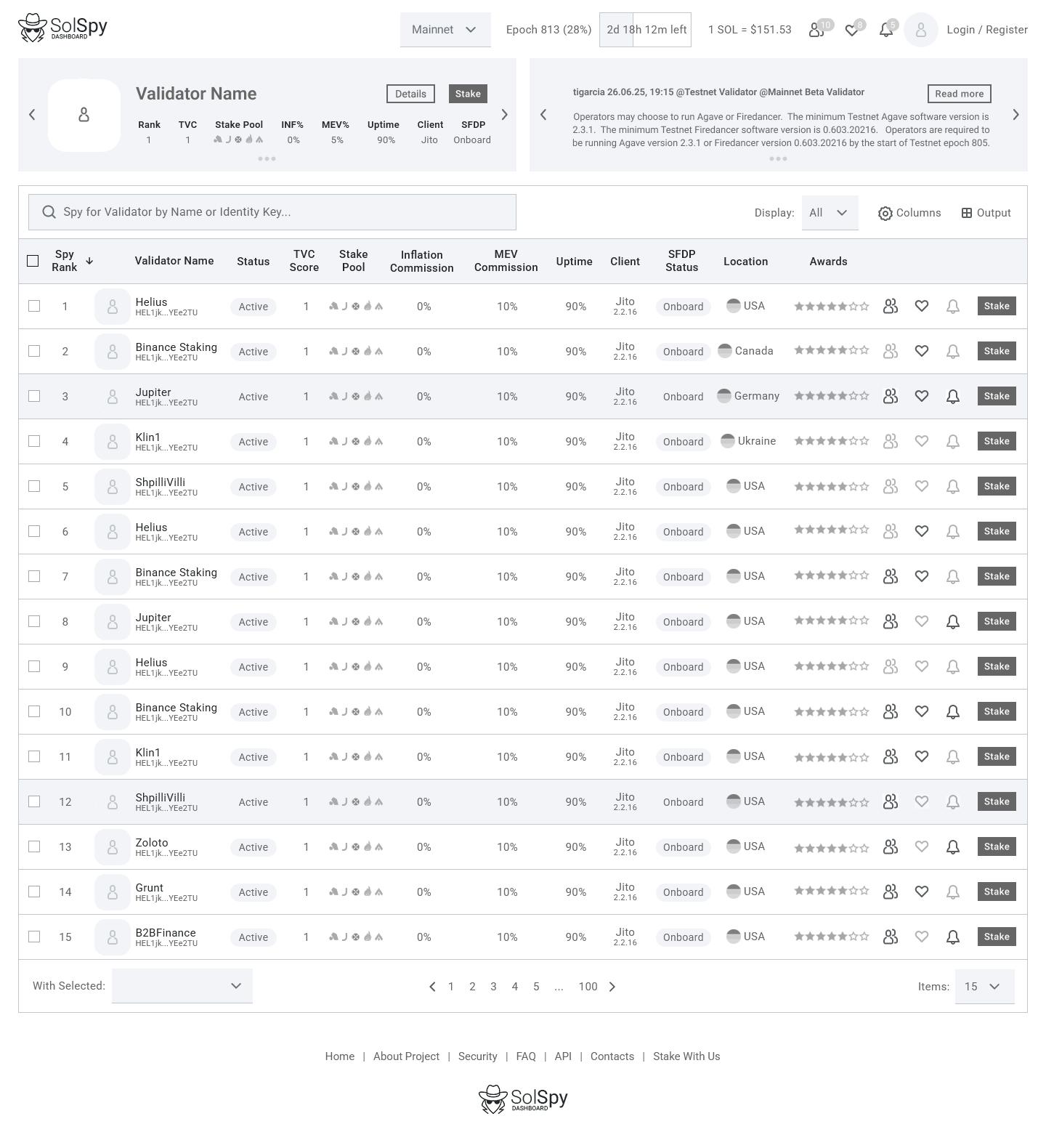
Каркас (wireframe) структури веб-додатку узгоджується окремо перед початком роботи над прототипом веб-додатку у Figma:

* Головна
* Сторінка валідатора
* Сторінка порівняння
* Особистий кабінет
* Текстова сторінка

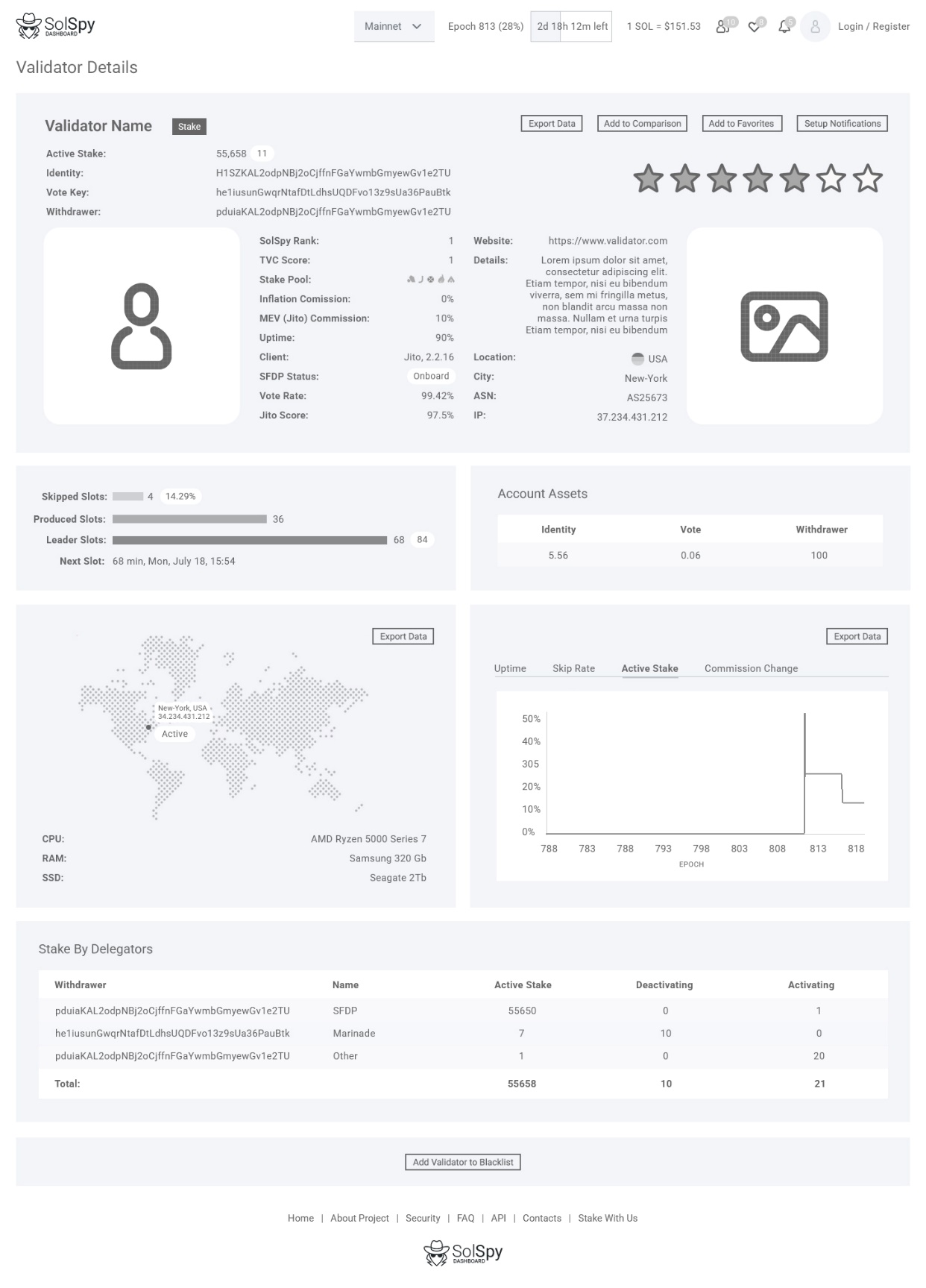


[Wireframes в Figma](https://www.figma.com/proto/kNn93tVLoNCbxlRr1kLL7r/Solana-Validators-Dashboard?node-id=0-1&t=jJwtskMrD2crL6it-1)

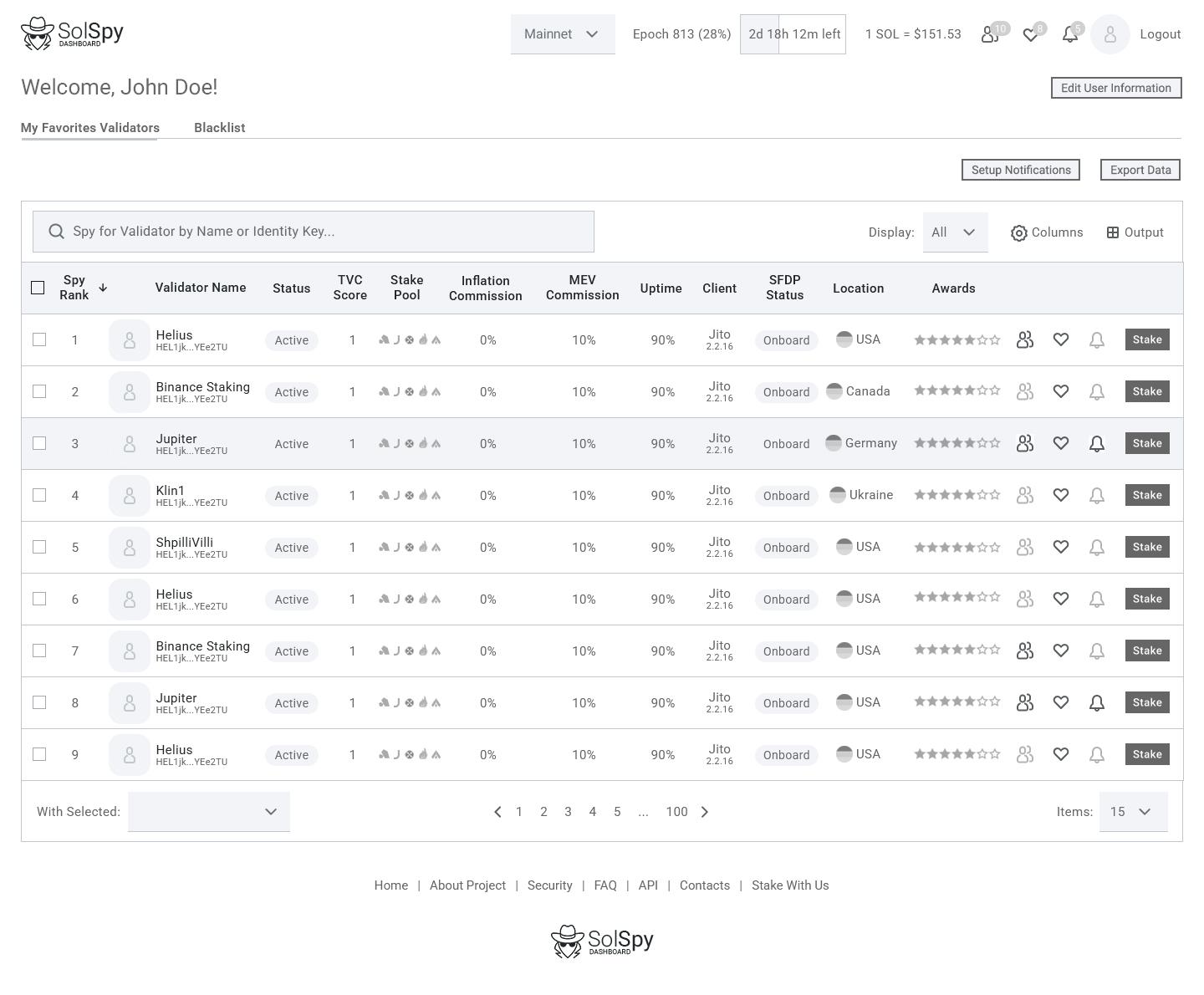
4.4.1. Wireframe - Головна сторінка



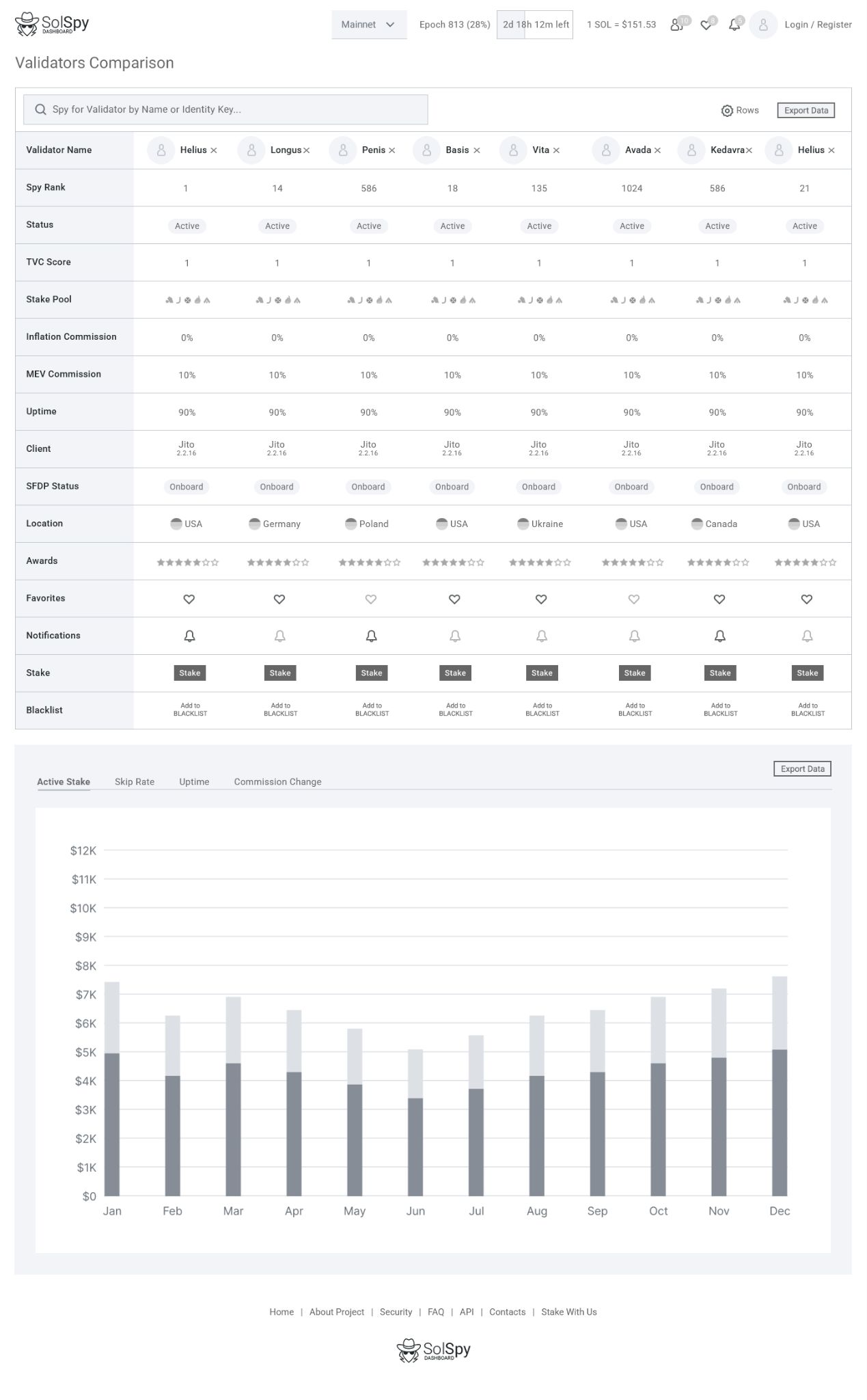
4.4.2. Wireframe - Сторінка валідатора



4.4.3. Wireframe - Особистий кабінет



4.4.4. Wireframe - Сторінка порівняння



4.4.5. Wireframe - Текстова сторінка



## 

# Розширення

## 5.1. Мобільний додаток

Це окремий проєкт на майбутнє - на даному етапі не реалізується.

## 5.2. Додаткові інтеграції

* Telegram-бот (Python) - на першому етапі просто отримує сповіщення (як у Stakewiz)
* API для зовнішнього користування (JSON)

5.2.1. Telegram-бот

Телеграм-бот призначений для отримання обраних користувачем сповіщень з сайту про зміну конфігурації обраних валідаторів. Відписка від сповіщень реалізується через бот (кнопка “Unsubscribe for this notifications”).

Вимоги:

* Python
* Сервер для розміщення бота (Heroku, DigitalOcean, AWS, Google Cloud або хостинг сайту)

# Нефункціональні вимоги

* Швидкість завантаження (до 2 - 5 секунд)
* Мінімальна затримка оновлень (до 10 - 30 секунд)

# Умови підтримки та навчання

## 7.1. Технічна підтримка

* Гарантія технічної підтримки - 6 місяців з моменту здачі проекту (1-3 години на тиждень). У підтримку входить виправлення багів і помилок роботи веб-додатку згідно узгодженого технічного завдання.
* Оновлення коду веб-додатка або додавання нового функціоналу виконується за додатковою угодою (окреме технічне завдання, окремі терміни, окрема оплата).
* Написання технічної документації.
* Засоби комунікації:
  + Telegram
  + Email
  + GoogleMeet

## 7.2. Контент-підтримка

Підтримка контенту виконується менеджером, який слідкує за правильним функціонуванням сайту, статистикою та наповнює/змінює новини.

## 7.2. Навчання

* Написання короткого мануалу (PDF/Docx)
* Онлайн-навчання через (GoogleMeet) - 2 години
* Якщо необхідно відеоінструкції (YouTube/GoogleDrive)

# Кошторис

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **#** | **Назва роботи** | **Години** | **Ставка $** | **Сума $** |
| 1 | Технічне завдання + Wireframes | 40 |  |  |
| 2 | Налаштування RPC-сервера | 12 |  |  |
| 3 | Frontend розроба | 150 |  |  |
| 4 | Backend розробка | 120 |  |  |
| 5 | Інтеграція з RPC + логіка | 80 |  |  |
| 6 | Налаштування сервера та деплой | 8 |  |  |
| 7 | Технічна підтримка \* | 72 \* |  |  |
| 8 | Навчання | 8 |  |  |
| 9 | Резерв (домен, хостинг та інше) | - |  |  |
|  | **Разом** | **522 \*\*** |  |  |

\* В залежності від потреби

\*\* З урахуванням часів технічної підтримки, яких може бути менше

\*\*\* З урахуванням повної технічної підтримки та резервних коштів

# Етапи реалізації

|  |  |  |
| --- | --- | --- |
| **#** | **Етап** | **Тривалість (робочих днів)** |
| 1 | Технічне завдання | 5 |
| 2 | Налаштування RPC-сервера | 1-2 |
| 3 | Розробка Backend | 20 |
| 4 | Розробка Frontend | 30 |
| 5 | Інтеграція з RPC + логіка | 10 |
| 6 | Налаштування сервера та деплой | 1-2 |
| 7 | Тестування | 3-5 |
| 8 | Навчання персоналу | 1 |
|  | **Загалом** | **70-75** |

Реліз - 2.5 - 3 місяці з моменту підписання договору про виконання робіт.

Повний реліз - 4 місяці з моменту підписання договору про виконання робіт.

# Додаток 1 - Ризики

1. Нестабільність RPC Solana - необхідність резервних серверів.
2. Нестабільність сторонніх API.
3. API Discord може змінити формат.
4. Блокування IP для визначення геолокації серверів.
5. Використання сторонніх сервісів.
6. Форс-мажори у зв’язку з війною в Україні.

# Додаток 2 - Налаштування хостингу та деплой

* Хостинг та домен надає замовник (а також логін/пароль)
* Бажані платформи:
  + Vercel
  + Digital Ocean
  + AWS
  + GoDaddy
  + або інші за рекомендацією розробника
* Протоколи:
  + SSH
  + FTP
  + HTTPS
  + SSL

# Додаток 3 - Розрахунок нагород Awards

Зірочки нараховуються за:

**1** — Skipped Slots Score – якщо 5 епох немає пропуску слотів, тоді зірочка

**2** — Software Version Score – якщо версія клієнта > або = мінімальній дозволеній версії від SFDP <https://discord.com/channels/428295358100013066/895740485140906054>

**3** — Uptime Score – якщо відсоток аптайму за 30 епох більше або дорівнює 98%

**4** — Inflation Commission Score – якщо комісія валідатора 0% (якщо валідатор збільшив її і потім повернув на нуль то зірочка не зараховується впродовж пів року)  
  
**5** — MEV Commission Score – якщо комісія валідатора 0% (якщо валідатор збільшив її і потім повернув на нуль то зірочка не зараховується впродовж пів року)

**6** – Published Information Score – бали за наповнення інформацією (імя, про ноду, картинка)

**7** — Site Score – бали за нормальний сайт (це зірочка отримується тільки за нашою згодою. Валідатор має написати нам на пошту з проханням перевірити його сайт, і якщо ми визнаємо його сайт достатньо інформативним для валідатора, ми тоді даємо йому цю зірочку)

На сторінці валідатора при наведенні на зірку повинно висвічуватись пояснення.

# Додаток 4 - Методи і розрахунки

У процесі розробці методи та розрахунки можуть змінюватись.

## A.4.1. Номер поточної епохи, прогресс та час до кінця епохи

Метод [getEpochInfo RPC](https://solana.com/uk/docs/rpc/http/getepochinfo):

curl https://api.devnet.solana.com -s -X \

POST -H "Content-Type: application/json" -d '

{

"jsonrpc": "2.0",

"id": 1,

"method": "getEpochInfo",

"params": [

{

"commitment": "finalized"

}

]

}

'

Повертає:

{

"jsonrpc": "2.0",

"result": {

"**absoluteSlot**": 395530735,

"blockHeight": 383485388,

"**epoch**": 915,

"**slotIndex**": 250735,

"**slotsInEpoch**": 432000,

"transactionCount": 16142346412

},

"id": 1

}

epoch: номер поточної епохи

slotIndex: номер поточного слота в епосі

slotsInEpoch: загальна кількість слотів у цій епосі

absoluteSlot: абсолютний номер слота

Прогрес в процентах:

const progressPercent = (slotIndex / slotsInEpoch) \* 100;

console.log(`Прогрес епохи: ${progressPercent.toFixed(2)}%`);

Час до кінця епохи (середня тривалість одного слота ~ 400 мс):

const progress = slotIndex / slotsInEpoch;

const slotsLeft = slotsInEpoch - slotIndex;

const timeLeftSeconds = slotsLeft \* 0.4; // час до кінця епохи в секундах

const days = Math.floor(timeLeftSeconds / (24 \* 3600));

const hours = Math.floor((timeLeftSeconds % (24 \* 3600)) / 3600);

const minutes = Math.floor((timeLeftSeconds % 3600) / 60);

const seconds = Math.floor(timeLeftSeconds % 60);

console.log(`Залишилось: ${days} дн, ${hours} год, ${minutes} хв, ${seconds} сек`);

## A.4.2. Поточний курс SOL до USD

Метод:

curl -s "<https://api.coingecko.com/api/v3/simple/price?ids=solana&vs_currencies=usd>"

Повертає:

{

"solana": {

"usd": 123.45

}

}

## 

## A.4.3. Отримати список валідаторів

Метод: [getVoteAccounts RPC](https://solana.com/uk/docs/rpc/http/getvoteaccounts)

В майн-нет:

curl https://api.mainnet-beta.solana.com -s -X POST -H "Content-Type: application/json" -d '

{

"jsonrpc": "2.0",

"id": 1,

"method": "getVoteAccounts"

}'

В тест-нет:

curl https://api.testnet.solana.com -s -X POST -H "Content-Type: application/json" -d '

{

"jsonrpc": "2.0",

"id": 1,

"method": "getVoteAccounts"

}'

Відповідь містить два масиви:

* current — валідатори, які зараз активні та голосують.
* delinquent — валідатори, які не голосують (відстали або неактивні).

Кожен елемент містить:

{

"votePubkey": "Vote111...",

"nodePubkey": "Node111...",

"activatedStake": 1234567890,

"commission": 10,

"epochVoteAccount": true,

"epochCredits": [

[123, 4567, 4000],

[124, 5000, 4567]

],

"lastVote": 12345678,

"rootSlot": 12345000,

"credits": 789012

}

* votePubkey — публічний ключ vote account валідатора (vote key).
* nodePubkey — публічний ключ вузла валідатора (validator identity key).
* activatedStake — кількість активного стейку (у лампортах).
* commission — комісія валідатора у відсотках (0–100).
* epochVoteAccount — чи є акаунт активним у поточній епосі (true/false).
* epochCredits — масив, що містить інформацію про кількість голосів у кожній епосі:
* [epochNumber, credits, previousCredits]
* lastVote — номер останнього слота, за який валідатор проголосував.
* rootSlot — останній root slot, який повідомив валідатор (може бути null).
* credits — загальна кількість кредитів, зароблених цим акаунтом (відображає активність голосування).

## A.4.4. Метадані валідатора

#### TypeScript налаштування:

import { Connection, PublicKey } from "@solana/web3.js";

const connection = new Connection("https://api.mainnet-beta.solana.com", "confirmed");

#### Аватар

Джерело: https://raw.githubusercontent.com/solana-labs/validators/main/validators/mainnet/validators.json

Метод: GitHub JSON конфіг

const validatorAvatar = `https://raw.githubusercontent.com/solana-labs/validators/main/validators/mainnet/validators.json` // поле "avatarUrl"

Альтернатива:

https://raw.githubusercontent.com/solana-labs/validators/main/validators/mainnet/avatars/<identity>.png

#### Назва/Ім’я

Джерело: getVoteAccounts або validators.app GitHub JSON

Метод (nodePubkey → metadata mapping):

const voteAccounts = await connection.getVoteAccounts();

console.log(voteAccounts.current[0]?.nodePubkey); // identity

Потім - назву по мапінгу з GitHub-JSON.

Альтернатива:

curl -X POST https://api.mainnet-beta.solana.com \

-H "Content-Type: application/json" \

-d '{

"jsonrpc": "2.0",

"id": 1,

"method": "getValidatorInfo"

}'

Поле: .result[].info.name

#### Location (країна) + City

Метод: на основі IP (getClusterNodes) → IP → GeoIP API (наприклад, ipinfo.io або maxmind)

const ip = nodes[0].gossip;

fetch(`https://ipinfo.io/${ip}?token=YOUR\_TOKEN`)

Альтернатива:

Без RPC дізнатись через IP-геолокацію з getClusterNodes

# витягуємо IP з `gossip` або `tpu`

curl ipinfo.io/<ip>?token=<your\_api\_token>

# or

curl http://ip-api.com/json/<ip>

#### Website

Метод: RPC getValidatorInfo

const validatorInfo = await connection.getValidatorInfo();

console.log(validatorInfo[0]?.info.website);

Альтернатива:

Поле в getValidatorInfo().result[].info.url або website

#### Скріншот сайта

Метод: TS Puppeteer

npm install puppeteer

npm install --save-dev @types/node

import puppeteer from 'puppeteer';

import fs from 'fs/promises';

async function takeScreenshotWithTypeScript(url: string, outputPath: string): Promise<void> {

const browser = await puppeteer.launch();

const page = await browser.newPage();

await page.setViewport({ width: 1280, height: 800 });

await page.goto(url, { waitUntil: 'networkidle2' });

await page.screenshot({

path: outputPath,

fullPage: true,

type: 'png',

});

await browser.close();

}

takeScreenshotWithTypeScript('https://solana.com', './screenshots/solana.png')

.then(() => console.log('Скріншот зроблено!'))

.catch(err => console.error('Помилка:', err));

Альтернатива - Playwright:

import { chromium } from 'playwright';

import fs from 'fs/promises';

async function captureWithPlaywright(url: string, path: string) {

const browser = await chromium.launch();

const page = await browser.newPage();

await page.setViewportSize({ width: 1280, height: 800 });

await page.goto(url);

await page.screenshot({ path, fullPage: true });

await browser.close();

}

captureWithPlaywright('https://solana.com', './screenshots/solana-playwright.png');

#### Description

Метод: RPC getValidatorInfo

console.log(validatorInfo[0]?.info.details);

#### ASN (Autonomous System Number) + IP

Метод:

- IP: через RPC getClusterNodes()

- ASN: через сторонній API ( https://ipinfo.io або https://ip-api.com )

const ip = nodes[0].gossip;

fetch(`http://ip-api.com/json/${ip}?fields=as`)

Альтернатива:

- IP: .gossip з getClusterNodes

- ASN: через IP API: curl <http://ip-api.com/json/><ip>?fields=as

## A.4.5. Identity Key (обрізаний до 4+...+4)

Метод: [RPC getVoteAccounts](https://solana.com/uk/docs/rpc/http/getvoteaccounts)

const identityKey = validator.nodePubkey;

const shortIdentity = identityKey.slice(0, 4) + "..." + identityKey.slice(-4);

## А.4.6. Vote Key та Withdrawer Key

Метод: [RPC getVoteAccounts](https://solana.com/uk/docs/rpc/http/getvoteaccounts)

const voteAccounts = await connection.getVoteAccounts();

сonst voteAccount = voteAccounts.current.find(v => v.nodePubkey === identity);

console.log(voteAccount.votePubkey, voteAccount.withdrawer); // vote key + withdrawer

Альтернатива:

curl -X POST https://api.mainnet-beta.solana.com \

-H "Content-Type: application/json" \

-d '{

"jsonrpc":"2.0",

"id":1,

"method":"getVoteAccounts"

}'

Поля: .result.current[] | votePubkey, withdrawer, nodePubkey

## 

## А.4.7. Статус (active / delinquent)

Метод: [RPC getVoteAccounts](https://solana.com/uk/docs/rpc/http/getvoteaccounts)

const activeValidators = voteAccounts.current;

const delinquentValidators = voteAccounts.delinquent;

Це дві частини: .current[] (active) та .delinquent[] із getVoteAccounts.

# якщо є у .current → active

# якщо .delinquent → delinquent

## А.4.8. TVC Score (місце за stake)

Метод:

- Список валідаторів через getVoteAccounts

- Обчислюється рейтинг на основі activatedStake

const sorted = [...voteAccounts.current].sort((a, b) => Number(b.activatedStake) - Number(a.activatedStake));

Альтернатива:

# Отримати весь список активних валідаторів:

curl -X POST https://api.mainnet-beta.solana.com \

-H "Content-Type: application/json" \

-d '{

"jsonrpc": "2.0",

"id": 1,

"method": "getVoteAccounts"

}'

#Ранжування за activatedStake

const sorted = result.current.sort(

(a, b) => Number(b.activatedStake) - Number(a.activatedStake)

);

const tvcRank = sorted.findIndex(v => v.votePubkey === yourVoteKey) + 1;

## А.4.9. Stake Pools (іконки пулів)

Через RPC це не видно напряму.

Потрібні власні іконки.

Джерело:

- https://api.validators.app/api/v1/\* (сторонній)

- Solana Foundation delegation program registry

Альтернатива (<https://validators.app/api/v1>):

https://api.validators.app/api/v1/validators/<voteKey>

## А.4.10. Inflation Commission

Метод: [RPC getVoteAccount](https://solana.com/uk/docs/rpc/http/getvoteaccounts)

curl https://api.devnet.solana.com -s -X \

POST -H "Content-Type: application/json" -d '

{

"jsonrpc": "2.0",

"id": 1,

"method": "getVoteAccounts",

"params": [

{

"commitment": "finalized",

"votePubkey": "i7NyKBMJCA9bLM2nsGyAGCKHECuR2L5eh4GqFciuwNT"

}

]

}

'

Відповідь:

{

"jsonrpc": "2.0",

"result": {

"current": [

{

"activatedStake": 38263229364446900,

"**commission**": 95,

"epochCredits": [

[902, 1383125544, 1376213656],

[903, 1390037304, 1383125544],

[904, 1396949288, 1390037304],

[905, 1403861272, 1396949288],

[906, 1406766600, 1403861272]

],

"epochVoteAccount": true,

"lastVote": 391573587,

"nodePubkey": "dv2eQHeP4RFrJZ6UeiZWoc3XTtmtZCUKxxCApCDcRNV",

"rootSlot": 391573556,

"votePubkey": "i7NyKBMJCA9bLM2nsGyAGCKHECuR2L5eh4GqFciuwNT"

}

],

"delinquent": []

},

"id": 1

}

Альтернатива:

const voteAccount = await connection.getVoteAccounts();

console.log(voteAccount.current[0]?.commission);

## А.4.11. MEV Commission

Джерело: Jito API (наприклад, https://api.jito.network або Jito Dashboard GraphQL)

## 

## А.4.12. Uptime

Непрямо через skipped slots / produced slots:

- збираються дані через getConfirmedBlocks

Альтернатива: сторонні сервіси: solanabeach.io API

## А.4.13. Client (з version)

Метод: [RPC getClusterNodes](https://solana.com/uk/docs/rpc/http/getclusternodes)

const nodes = await connection.getClusterNodes();

console.log(nodes[0].version); // наприклад solana-core 1.16.18

Альтернатива:

curl -X POST https://api.mainnet-beta.solana.com \

-H "Content-Type: application/json" \

-d '{

"jsonrpc": "2.0",

"id": 1,

"method": "getClusterNodes"

}'

Поле: .version

## А.4.14. Статус SFDP

Джерело: [Foundation SFP list](https://github.com/solana-foundation/validator-identity/blob/main/registry/mainnet-beta/validators.csv) (CSV мапиться “вручну”)

Альтернатива: З GitHub або CSV файлу з Solana Foundation (<https://github.com/solana-foundation/validator-keypairs>)

## А.4.15. Vote Rate

Потрібно рахувати вручну: кількість голосів від валідатора / кількість лідер-слотів за епоху.

Приклад на TS (Node.js + fetch):

const fetch = require("node-fetch");

async function getVoteRate(votePubkey, identityPubkey) {

const rpcUrl = "https://api.testnet.solana.com";

// Отримуємо голоси

const voteResp = await fetch(rpcUrl, {

method: "POST",

headers: { "Content-Type": "application/json" },

body: JSON.stringify({

jsonrpc: "2.0",

id: 1,

method: "getVoteAccounts"

})

});

const voteData = await voteResp.json();

const voteAccount = voteData.result.current.find(v => v.votePubkey === votePubkey);

const [ , current, previous ] = voteAccount.epochCredits.slice(-1)[0];

const voteCount = current - previous;

// Отримуємо лідер-слоти

const leaderResp = await fetch(rpcUrl, {

method: "POST",

headers: { "Content-Type": "application/json" },

body: JSON.stringify({

jsonrpc: "2.0",

id: 1,

method: "getLeaderSchedule",

params: [null, { identity: identityPubkey }]

})

});

const leaderData = await leaderResp.json();

const leaderSlots = leaderData.result[identityPubkey].length;

// Обчислюємо

const voteRate = voteCount / leaderSlots;

console.log(`Vote Rate: ${(voteRate \* 100).toFixed(2)}%`);

}

getVoteRate("G1z...ABC", "9xy...XYZ");

## А.4.16. Jito Score

Витягується через Jito API (https://jito.network/) або Jito Validators Leaderboard

Альтернатива: GraphQL - leaderboard за 30 epoch

## А.4.17. Leader Slots

Щоб отримати всі, треба обходити:

connection.getLeaderSchedule(undefined, identityKey)

Альтернатива:

curl -X POST https://api.mainnet-beta.solana.com \

-H "Content-Type: application/json" \

-d '{

"jsonrpc":"2.0",

"id":1,

"method":"getLeaderSchedule",

"params":[ null, { "identity": "<validator\_identity>" } ]

}'

## А.4.17. Time Next Slot

Обчислюється на базі поточного слоту:

const currentSlot = await connection.getSlot();

const blockTime = await connection.getBlockTime(currentSlot);

Альтернатива:

- Поточний слот: getSlot

- Час слота: getBlockTime

curl -X POST https://api.mainnet-beta.solana.com \

-H "Content-Type: application/json" \

-d '{

"jsonrpc":"2.0",

"id":1,

"method":"getSlot"

}'

Або:

time\_to\_next\_slot = (next\_leader\_slot - current\_slot) \* slot\_duration

## А.4.18. Skipped Slots / Produced Slots

Метод: Через getLeaderSchedule + getConfirmedBlocks ➝ Перевірка, чи блок існує для слота лідера.

## 

## А.4.19. Account Assets

Метод: [getAccountInfo RPC](https://solana.com/uk/docs/rpc/http/getaccountinfo)

// Identity

const accInfo = await connection.getAccountInfo(new PublicKey(identityKey));

console.log(accInfo.lamports);

// Так само Votes/Withdrawer

Альтернатива:

curl -X POST https://api.mainnet-beta.solana.com \

-H "Content-Type: application/json" \

-d '{

"jsonrpc":"2.0",

"id":1,

"method":"getAccountInfo",

"params":[

"IDENTITY\_PUBKEY",

{ "encoding":"jsonParsed" }

]

}'

Поля: .lamports, .data.parsed.info

## А.4.20. Сервер (CPU, RAM, SSD)

Вимірюється або через власного агента на ноді або crowd-даними (типу Telemetry), якщо доступно.

Альтернатива:

- Користуватись дашбордом на основі Prometheus + Node Exporter

https://github.com/prometheus/node\_exporter

## А.4.21. Обчислення Spy Rank

|  |  |  |  |
| --- | --- | --- | --- |
| **Наші показники** | **Як ми рахуватимемо** | **Як рахує ці показники Stakewiz** | **Коментарі** |
| **Skipped Slots Score** | 10% | Value 0% + 15% |  |
| **Software Version Score** | 10% |  |  |
| **Uptime Score** | (30 epochs) 100% - 20% | (30 days) 100% - 20% |  |
| **Inflation Commission Score** | 10 | Up to 5% score for commission 0%, no score for 10% commission and above |  |
| **MEV Commission Score** | 10 |  |  |
| **Published Information Score** | 3 out of 3 – 5% | 5 out of 5 – 10% |  |
| **Site Score** | approved 5% | - |  |
| **Stake Weight** | Від 100k до 400k – 10%  від 400k і далі 5%  від 0k до 100k – 0% | Up to 15%, 0% for any stake that is >= 10% of the largest validator stake |  |
| **Operation History** | 5% | 75 epoch + 10% |  |
| **TVC (5 epoch)** | 5% |  |  |
| **APY**  **(APY ≈ (1 + R) ^ N - 1)** | 10% |  | R — середній прибуток (reward rate) за одну епоху  N — кількість епох у році (приблизно 134 епохи)  ^ — піднесення до степеня |
| **Version Penalty** |  |  |  |
| **Superminority Penalty** |  |  |  |
| **Withdraw Authority Penalty** |  | Hawing the vote account withdraw authority set to validator’s identity keypair is a bad security practice and incurs a -20% penalty | Differs from validator identity (good) |

Очікується приклад розрахунку.

## А.4.22. Налаштування RPC на ноді в mainnet

Замість <https://api.mainnet-beta.solana.com> (або dev та test) потрібно використовувати власний RPC-сервер.

1. В solana.service видалити:

--private-rpc

і додати (якщо нема):

--rpc-port 8899 \

--rpc-bind-address 0.0.0.0 \

--full-rpc-api \

2. Налаштувати фаервол, щоб RPC приймала запити на порт 8899 тільки з конкретного IP:

sudo ufw allow ssh

sudo ufw enable

sudo ufw allow from IP to any port 8899 // вказати IP, де буде розташований сайт

sudo ufw deny 8899

sudo ufw status numbered

3. Перезавантажити сервіси і солану:

sudo systemctl daemon-reexec

sudo systemctl daemon-reload

sudo systemctl restart solana

4. Перевірка статусу:

sudo systemctl status solana

5. Перевірка RPC:

ss -tulnp | grep 8899

повинен повернути:

tcp LISTEN 0 1024 0.0.0.0:8899 0.0.0.0:\* users:(("agave-validator",pid=1093848,fd=437864))

6. Локальний запрос (після того, як пішов catchup):

curl http://127.0.0.1:8899 -X POST -H "Content-Type: application/json" -d '{"jsonrpc":"2.0","id":1,"method":"getHealth"}'

// замість <http://127.0.0.1:8899> - вказати адресу RPC-сервера

повинен повернути:

{"jsonrpc":"2.0","result":"ok","id":1}

## А.4.23. Vote Credits

Метод: [getVoteAccounts RPC](https://solana.com/uk/docs/rpc/http/getvoteaccounts)

curl https://api.mainnet-beta.solana.com \

-X POST \

-H "Content-Type: application/json" \

-d '{

"jsonrpc": "2.0",

"id": 1,

"method": "getVoteAccounts",

"params": [

{

"votePubkey": "ВАШ\_VOTE\_PUBKEY"

}

]

}'

Поля: . voteAccounts[].epochVoteAccount[].epochCredits

Відповідь:

{

"result": {

"current": [

{

"votePubkey": "ВАШ\_VOTE\_PUBKEY",

"epochVoteAccount": {

"epoch": 652,

"epochCredits": [

[650, 1234, 1200],

[651, 1450, 1400],

[652, 300, 300]

]

}

}

]

}

}

Тобто для епохи 652 валідатор отримав 300 голосів.

## А.4.24. Active Stake, Stake Changes (Pending Stake), Activating, Deactivating, Deligator’s Name

Метод: [getVoteAccounts RPC](https://solana.com/uk/docs/rpc/http/getvoteaccounts)

curl https://api.mainnet-beta.solana.com \

-X POST \

-H "Content-Type: application/json" \

-d '{

"jsonrpc": "2.0",

"id": 1,

"method": "getVoteAccounts",

"params": [

{

"votePubkey": "ВАШ\_VOTE\_PUBKEY"

}

]

}'

Відповідь:

{

"result": {

"current": [

{

"votePubkey": "ВАШ\_VOTE\_PUBKEY",

"activatedStake": 123456789000, // Active Stake (у лампортах)

"epochVoteAccount": {...}

}

],

"delinquent": []

}

}

**activatedStake** → це Active Stake у лампортах, ділимо на 1e9 (1000000000) для SOL

Для отримання **Delegator**:

* + - 1. votePubkey
      2. getProgramAccounts → Stake Program (Stake11111111111111111111111111111111111111)
      3. Фільтруємо по votePubkey у полі delegation.voterPubkey.

Метод: [getProgramAccounts RPC](https://solana.com/uk/docs/rpc/http/getprogramaccounts)

curl https://api.mainnet-beta.solana.com \

-X POST \

-H "Content-Type: application/json" \

-d '{

"jsonrpc": "2.0",

"id": 1,

"method": "getProgramAccounts",

"params": [

"Stake11111111111111111111111111111111111111",

{

"encoding": "jsonParsed",

"filters": [

{

"memcmp": {

"offset": 124,

"bytes": "ВАШ\_VOTE\_PUBKEY"

}

}

]

}

]

}'

У відповіді буде список Stake Accounts - делегатори. Ім’я делегатора можна знайти лише через сторонні сервіси (Solana Beach, validators.app).

Для отримання **Activating**/**Deactivating** (TypeScript):

import fetch from "node-fetch";

const RPC\_URL = "https://api.mainnet-beta.solana.com";

const VOTE\_PUBKEY = "ВСТАВ\_СВІЙ\_VOTE\_PUBKEY";

const LAMPORTS\_PER\_SOL = 1\_000\_000\_000n;

async function rpcCall(method: string, params: any[] = []) {

const res = await fetch(RPC\_URL, {

method: "POST",

headers: { "Content-Type": "application/json" },

body: JSON.stringify({ jsonrpc: "2.0", id: 1, method, params }),

});

const json = await res.json();

return json.result;

}

async function main() {

// 1. Поточна епоха

const epochInfo = await rpcCall("getEpochInfo");

const currentEpoch = epochInfo.epoch;

// 2. Усі stake accounts для votePubkey

const stakeAccounts = await rpcCall("getProgramAccounts", [

"Stake11111111111111111111111111111111111111",

{

encoding: "jsonParsed",

filters: [

{

memcmp: {

offset: 124,

bytes: VOTE\_PUBKEY,

},

},

],

},

]);

let activeStake = 0n;

let activatingStake = 0n;

let deactivatingStake = 0n;

for (const acc of stakeAccounts) {

const delegation = acc.account.data.parsed.info.stake?.delegation;

if (!delegation) continue;

const stake = BigInt(delegation.stake);

const activationEpoch = Number(delegation.activationEpoch);

const deactivationEpoch = BigInt(delegation.deactivationEpoch);

if (activationEpoch < currentEpoch && deactivationEpoch === 18446744073709551615n) {

activeStake += stake;

} else if (activationEpoch === currentEpoch) {

activatingStake += stake;

} else if (deactivationEpoch !== 18446744073709551615n && deactivationEpoch >= BigInt(currentEpoch)) {

deactivatingStake += stake;

}

}

console.log(`Active Stake: ${(Number(activeStake) / 1e9).toFixed(2)} SOL`);

console.log(`Activating: ${(Number(activatingStake) / 1e9).toFixed(2)} SOL`);

console.log(`Deactivating: ${(Number(deactivatingStake) / 1e9).toFixed(2)} SOL`);

}

main().catch(console.error);