Специфікація програмного забезпечення

**Vision and Scope Document**

**for**

**“Генератор REST API сервісів на основі схеми реляційної бази даних (Backend)”**

**Version 1.0**

**Prepared by**

**Терновському Денису Сергійовичу**

**ХНУРЕ**

**16.04.2025**

**Зміст (Table of Contents)**

[Історія ревізій 67](#_Toc200617158)

[**1. ВИМОГИ ДО БІЗНЕСУ** 68](#_Toc200617159)

[1.1. Передумови 68](#_Toc200617160)

[1.2. Можливості для бізнесу 68](#_Toc200617161)

[1.3. Бізнес-цілі та критерії успіху 69](#_Toc200617162)

[1.4. Потреби клієнтів або ринку 70](#_Toc200617163)

[1.5. Бізнес-ризики 72](#_Toc200617164)

[**2. ОБРАЗ РІШЕННЯ** 73](#_Toc200617165)

[2.1. Судження про бачення проекту 73](#_Toc200617166)

[2.2. Основний функціонал 73](#_Toc200617167)

[2.3. Припущення та залежності 74](#_Toc200617168)

[**3. ОБСЯГ І ОБМЕЖЕННЯ** 75](#_Toc200617169)

[3.1. Обсяг першого випуску 75](#_Toc200617170)

[3.2. Обсяг подальших випусків 75](#_Toc200617171)

[3.3. Обмеження та виключення 75](#_Toc200617172)

[**4. БІЗНЕС-КОНТЕКСТ** 77](#_Toc200617173)

[4.1. Бізнес-цілі та критерії успіху 77](#_Toc200617174)

[4.2. Робоче середовище 77](#_Toc200617175)

#### Історія ревізій

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Date** | **Reason For Chages** | **Version** |
| Терновський Д. С. | 16.04.2025 | Розробка документу | Version 1.0 |

1. **ВИМОГИ ДО БІЗНЕСУ**
   1. Передумови (Background)

Розробка генератора REST API стала відповіддю на потребу в прискоренні початкових етапів бекенд-розробки у проєктах з використанням реляційних баз даних. Більшість сучасних CRUD-сервісів мають повторювану структуру, яку можна автоматизувати на основі вже існуючої схеми БД. Водночас підтримка документації (OpenAPI), налаштування авторизації й відповідність стандартам REST – залишаються рутинними завданнями.

Передумови для розробки:

* зростання попиту на швидке створення backend-рішень у стартапах і малих командах;
* уніфікованість CRUD-логіки при взаємодії з базами даних;
* потреба в автоматичному оновленні документації після зміни структури БД;
* необхідність інтеграції з frontend через стандартизоване REST API.
  1. Можливості для бізнесу (Business Opportunity)

Розроблена система має потенціал бути впровадженою у широкому спектрі проєктів, пов’язаних із швидким створенням backend-сервісів, та пропонує ряд переваг.

* + 1. Зростання попиту на REST API у малих і середніх командах

Більшість стартапів та малих ІТ-команд працюють у стислі терміни та з обмеженими ресурсами. Генератор REST API дозволяє значно скоротити час розробки та уникнути ручної реалізації стандартних CRUD-операцій, що є актуальним при створенні MVP і прототипів.

* + 1. Автоматизація документації

Система автоматично формує документацію до API на основі OpenAPI-специфікації. Це знижує навантаження на розробника, полегшує підтримку проєкту й дає змогу одразу тестувати запити через Faker UI.

* + 1. Інтеграція з різними СУБД

Підтримка як PostgreSQL, так і MongoDB забезпечує більшу гнучкість у впровадженні генератора в існуючі системи. Це відкриває можливості для використання як в навчальних, так і в комерційних проєктах.

* + 1. Можливість кастомізації

Архітектура системи дозволяє розширення функціоналу за рахунок шаблонів генерації та конфігураційних параметрів. Це дає змогу налаштовувати ендпоінти під потреби конкретного проєкту без втручання в основний код генератора.

* + 1. Перспективи для open-source і SaaS-моделі

Генератор може бути відкритим проєктом для спільноти з можливістю розширення й адаптації, або ж стати основою для SaaS-рішення, яке дозволить створювати API через інтерфейс без написання коду.

* + 1. Освітній потенціал

Простота використання генератора робить його ідеальним інструментом для студентів, які можуть використовувати його як практичну базу для вивчення REST API, баз даних і архітектури серверних додатків.

* 1. Бізнес-цілі та критерії успіху (Business Objectives and Success Criteria)

Бізнес-цілі:

BO1. Розробка генератора REST API з CLI-інтерфейсом та підтримкою Faker-документації:

* Мета: Створення гнучкого інструменту, який дозволяє автоматично створювати ендпоінти REST API та відповідну документацію на основі структури бази даних.
* SC1.1: Успішна генерація API для 100% таблиць БД.
* SC1.2: Сумісність із PostgreSQL і MongoDB.
* BO2. Випуск MVP-версії як open-source рішення:
* Мета: Надання першої стабільної версії генератора, яка буде доступною для спільноти та придатною до використання в навчальних і дослідницьких цілях.
* SC2.1: Завантаження перших 50 користувачами через 1 місяць після релізу.
* SC2.2: Позитивний відгук 80% користувачів на Github.
* BO3. Динамічна підтримка змін у структурі бази даних:
* Мета: Реалізація механізмів, які дозволяють адаптувати API до змін у БД без необхідності ручного втручання.
* SC3.1: При зміні структури таблиць відповідні ендпоінти оновлюються автоматично.

BO4. Реалізація системи автентифікації:

* Мета: Забезпечення базового рівня захисту доступу до згенерованого API через сучасні методи автентифікації.
* SC4.1: Підтримка JWT або basic-token авторизації на рівні API.
  1. Потреби клієнтів або ринку (Customer or Market Needs)

Основні потреби клієнтів включають:

* + 1. Автоматизація рутинних операцій:
* Потреба: Розробники прагнуть уникати ручної реалізації однотипних CRUD-операцій для кожної таблиці бази даних.
* Вимога: Генератор повинен забезпечувати повну автоматизацію базових операцій та їх адаптацію до змін у структурі БД.
  + 1. Зрозумілий і гнучкий інтерфейс використання:
* Потреба: Користувачі очікують мінімального порогу входу у використання інструменту.
* Вимога: Інтерфейс CLI повинен бути інтуїтивним, з прикладами, шаблонами та можливістю інтеграції в CI/CD процеси.
  + 1. Надійність і підтримка:
* Потреба: Користувачі хочуть бути впевнені у стабільності згенерованого API та мати можливість швидко вирішувати помилки.
* Вимога: Повинна бути доступна документація, автоматизовані тести та система звітів про помилки.
  + 1. Гнучкість у налаштуванні:
* Потреба: Проєкти мають різні вимоги до авторизації, структури маршрутів, моделей тощо.
* Вимога: Інструмент має дозволяти кастомізацію шаблонів генерації, форматів відповіді та обробки помилок.
  + 1. Безпека доступу до API:
* Потреба: Підприємства вимагають захисту своїх сервісів від несанкціонованого доступу.
* Вимога: Підтримка базових механізмів авторизації (JWT, токени), а також захисту від поширених атак (CORS, CSRF, rate-limiting).

Основні потреби ринку включають:

* + 1. Прискорення розробки ПЗ:
    - Потреба: Компанії шукають рішення, які зменшують витрати на розробку MVP і скорочують time-to-market.
    - Вимога: Інструмент повинен дозволяти створювати REST API за хвилини, а не дні.
    1. Сумісність із сучасною інфраструктурою:
  + Потреба: Системи мають бути адаптовані до хмарних рішень, CI/CD-пайплайнів, DevOps середовищ.
  + Вимога: Генератор повинен мати прості опції для інтеграції з Docker, GitHub Actions, Vercel, Heroku тощо.
    1. Стандартизація та масштабованість:
* Потреба: Корпоративні клієнти прагнуть мати уніфіковані API між різними командами.
* Вимога: Згенеровані API мають відповідати стандартам REST/JSON:API та підтримувати версіонування.
  + 1. Відкритість коду та доступність:
* Потреба: Ринок надає перевагу open-source рішенням з активною спільнотою.
* Вимога: Інструмент має мати відкритий код, систему pull requestів та публічну документацію.
  1. Бізнес-ризики (Business Risks)

Розробка та впровадження генератора REST API пов’язані з певними ризиками, які можуть вплинути на досягнення бізнес-цілей. Основні ризики включають:

* Технічні ризики. При реалізації підтримки різних СУБД можливі проблеми сумісності та некоректна генерація ендпоінтів для нетипових схем таблиць.
* Безпекові ризики. Недостатньо налаштовані механізми автентифікації можуть призвести до несанкціонованого доступу до даних через згенероване API.
* Ризики масштабування. У разі активного використання інструменту у великих проектах без оптимізації шаблонів можливо зниження продуктивності або виникнення складнощів при оновленні структури БД.
* Ризики впровадження. Команди, що звикли до ручної розробки API, можуть відмовитись від переходу на автоматизоване рішення через недовіру або складність адаптації.
* Залежність від сторонніх бібліотек. Такі компоненти, як Sequelize або Faker, можуть мати власні обмеження, помилки чи оновлення, які впливають на стабільність генератора.
* Ризики конкуренції. На ринку вже існують популярні рішення (наприклад, Supabase, PostgREST), які можуть мати ширший функціонал або спільноту.
* Мінімізація цих ризиків передбачає розробку модульної архітектури, покриття основних сценаріїв юніт-тестами, ведення документації та регулярну перевірку залежностей. Важливою також є можливість кастомізації під конкретні потреби користувача.

1. **ОБРАЗ РІШЕННЯ**
   1. Судження про бачення проекту (Vision Statement)

Проєкт має на меті створення універсального генератора REST API, який дозволить розробникам швидко створювати серверну логіку для взаємодії з реляційними базами даних. Система орієнтована на малий бізнес, стартапи, освітні цілі та окремих розробників, які хочуть отримати готову серверну частину за кілька хвилин.

Основні аспекти бачення проєкту включають:

* Інноваційність: Автоматизація рутинної розробки API та документації з використанням сучасних засобів OpenAPI, Sequelize та [Node.js](http://node.js).
* Надійність і точність: Генератор формує стабільні, предикативні маршрути, пов’язані з існуючими таблицями бази даних, і оновлює їх у разі змін у схемі.
* Зручність: Простий CLI-інтерфейс та можливість налаштування шаблонів генерації забезпечують адаптивність під різні проєкти.
* Безпека: Інтеграція з базовими механізмами автентифікації (JWT, токени), захистом CORS та перевіркою запитів.
* Масштабованість: Можливість інтеграції у великі проєкти завдяки розширюваній структурі й шаблонізованому підходу.
* Інтеграція: Сумісність із сучасними хмарними середовищами (Heroku, Railway, Vercel) та CI/CD процесами.
  1. Основний функціонал (Major Features)

Програмна система включає наступний функціонал:

* зчитування структури бази даних через INFORMATION\_SCHEMA або pg\_catalog;
* генерація CRUD-ендпоінтів для кожної таблиці бази даних;
* створення моделей Sequelize;
* створення Faker-документації на основі отриманої структури API;
* опціональне додавання автентифікації на основі JWT;
* конфігурація параметрів генерації (префікси маршрутів, назви моделей, тип відповіді тощо);
* інтеграція в CI/CD процеси через CLI;
* кастомізація шаблонів відповіді і логіки обробки запитів.
  1. Припущення та залежності (Assumptions and Dependencies)

Для коректної роботи системи необхідно дотримання наступних умов:

Технології розробки:

* + 1. Node.js v18+ – середовище виконання JavaScript;
    2. Express.js – HTTP-фреймворк для побудови маршрутизації;
    3. Sequelize – ORM для роботи з PostgreSQL та MongoDB;
    4. Faker.js – для генерації та виводу документації;
    5. JSON Web Token (JWT) – для автентифікації користувачів;
    6. PostgreSQL / MongoDB – як джерело даних для аналізу структури;
    7. Dotenv – для зберігання конфігурацій у змінних середовища;
    8. Commander / Yargs – для побудови CLI-інтерфейсу.

Безпека:

* Застосовується валідація запитів, обмеження доступу до маршрутів, захист CORS;
* Всі секретні ключі зберігаються в .env файлі, недоступному для зовнішнього середовища.

Припущення:

* Користувач має базові знання командного рядка;
* Користувач має доступ до робочої реляційної БД;
* Структура БД не містить складних кастомних типів даних, які не підтримуються Sequelize;
* Документація Faker буде використовуватися для перевірки запитів під час розробки, а не у продакшн-середовищі без додаткової авторизації.

1. **ОБСЯГ І ОБМЕЖЕННЯ**
   1. Обсяг першого випуску (Scope of Initial Release**)**

Перший випуск генератора REST API включає наступний функціонал:

* автоматичне зчитування структури бази даних (PostgreSQL / MongoDB);
* генерація моделей Sequelize на основі таблиць БД;
* створення стандартних CRUD-ендпоінтів для кожної таблиці;
* автогенерація документації за стандартом Faker (OpenAPI);
* можливість виведення документації через Faker UI;
* CLI-інтерфейс для запуску генератора та управління параметрами;
* базова інтеграція автентифікації з використанням JWT;
* кастомізація префіксів маршрутів, назв моделей, структури відповіді API.
  1. Обсяг подальших випусків (Scope of Subsequent Release)

У наступних версіях проєкту планується реалізація:

* генерація GraphQL-схеми поряд з REST API для універсальності вибору;
* створення інтерфейсу для інтерактивного налаштування генерації API через веб-UI;
* підтримка додаткових баз даних (наприклад, SQLite, MSSQL);
* генерація тестів для згенерованих ендпоінтів (unit та integration);
* покращена система авторизації з ролями (RBAC);
* можливість генерації PWA-адмінки на основі структури БД;
* інтеграція з CI/CD інструментами (GitHub Actions, GitLab Pipelines);
* створення шаблонів для автогенерації користувацьких хуків та middleware;
* додаткові типи запитів: фільтрація, сортування, пагінація та складні зв’язки між таблицями.
  1. Обмеження та виключення (Limitations and Exclusions)
* генератор не модифікує саму базу даних (не створює / не редагує таблиці);
* не підтримуються надто складні типи даних або користувацькі функції у БД;
* Faker-документація не включає опис бізнес-логіки, лише структуру запитів;
* автентифікація реалізована на базовому рівні (JWT або токен), без зовнішніх провайдерів;
* CLI не підтримує інтерактивного режиму (лише передача параметрів);
* система не містить UI з візуалізацією API — лише через Faker UI або Postman.

1. **БІЗНЕС-КОНТЕКСТ**
   1. Бізнес-цілі та критерії успіху (Business Objectives and Success Criteria)

Таблиця 4.1.1 Профілі зацікавлених сторін

|  |  |  |  |
| --- | --- | --- | --- |
| **Зацікавлена сторона** | **Основна цінність** | **Основні інтереси** | **Обмеження** |
| Індивідуальні розробники | Швидке створення готового бекенду | Автоматизація CRUD, генерація документації, просте налаштування | Обмежені технічні знання, потреба в документації |
| Стартап-команди | Скорочення time-to-market | Швидке створення MVP, підтримка змін у БД, масштабування API | Обмеження по ресурсах та часі |
| Освітні заклади | Навчання принципам REST, баз даних і безпеки | Демонстрація архітектури серверної частини, автоматичне оновлення API | Відсутність складної авторизації або гнучкого налаштування |
| Команди з підтримки проектів | Уніфікований бекенд для багатьох застосунків | Легка підтримка API, стандартизовані підходи до документації та автентифікації | Висока потреба в кастомізації та інтеграції |

* 1. Робоче середовище (Operating Environment)

Система розгортається у середовищі з доступом до інтернету, що включає серверні ресурси для хостингу backend-частини, бази даних, а також веб-клієнта. Генератор REST API працює в середовищі Node.js 18+ та потребує встановлених модулів (Sequelize, Express, Faker) для забезпечення роботи серверної логіки та автогенерації документації.