**МІНІСТЕРСТВО ОСВІТИ ТА НАУКИ УКРАЇНИ**

**НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ «КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ ІМЕНІ ІГОРЯ СІКОРСЬКОГО»**

**НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ АТОМНОЇ ТА ТЕПЛОВОЇ ЕНЕРГЕТИКИ**

**КАФЕДРА ІНЖЕНЕРІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ В ЕНЕРГЕТИЦІ**

ПУНКТИ 5-8 КУРСОВОЇ РОБОТИ

**з дисципліни:** ««Основи Веб-програмування»

**на тему:** «Додаток, який створює сильні та безпечні паролі»

**Виконав**:

Семенов Максим Михайлович

Студент 2 курсу

Групи ТВ-32

Посилання на **GitHub** репозиторій: <https://github.com/max-s-m/web-kursova5-8>

**Київ - 2025**

**Розділ 5. Створення діаграм компонентів, взаємодії та класів веб-додатку.**

Взаємодія клієнтської частини "Генератора Паролів" із сервером відбувається через HTTP API для виконання операцій зі сховищем паролів.

1. Валідація майстер-ключа / Розблокування сховища:
   1. Після введення користувачем майстер-ключа та натискання кнопки "Розблокувати / Створити сховище", клієнт виконує POST-запит на сервер за шляхом /api/validate-key.
   2. У тілі запиту передається об'єкт { masterKey: "введений\_ключ" }.
   3. Обробка відповіді:
      1. Статус 200 OK: Майстер-ключ валідний (або нове сховище створено). Клієнт зберігає майстер-ключ локально для поточної сесії, відображає повідомлення про успіх та переходить до Головного Меню.
      2. Статус 401 Unauthorized: Неправильний майстер-ключ. Клієнт відображає повідомлення про помилку, отримане з тіла відповіді сервера (поле message).
      3. Статус 400 Bad Request: Не надано майстер-ключ. Клієнт відображає відповідне повідомлення.
      4. Статус 5xx Server Error: Клієнт відображає стандартне повідомлення "Серверна помилка, спробуйте ще раз".
2. Отримання списку збережених паролів:
   1. Після вибору пункту "Переглянути паролі" в Головному Меню, клієнт виконує GET-запит на сервер за шляхом /api/passwords.
   2. Обробка відповіді:
      1. Статус 200 OK: Сервер повертає масив об'єктів { id, encrypted\_login, encrypted\_password }. Клієнт дешифрує encrypted\_login та encrypted\_password за допомогою збереженого майстер-ключа та відображає список користувачеві.
      2. Статус 5xx Server Error: Клієнт відображає повідомлення "Помилка завантаження паролів".
3. Додавання нового запису пароля:
   1. Після заповнення користувачем даних для нового пароля (логін, пароль) та підтвердження, клієнт спочатку шифрує логін та пароль локально за допомогою майстер-ключа.
   2. Потім виконується POST-запит на сервер за шляхом /api/passwords.
   3. У тілі запиту передається об'єкт { encryptedLogin: "зашифрований\_логін", encryptedPassword: "зашифрований\_пароль" }.
   4. Обробка відповіді:
      1. Статус 201 Created: Пароль успішно додано. Клієнт відображає повідомлення про успіх.
      2. Статус 400 Bad Request: Некоректні вхідні дані. Клієнт відображає повідомлення про помилку з тіла відповіді.
      3. Статус 409 Conflict: Запис з таким (зашифрованим) логіном вже існує. Клієнт відображає відповідне повідомлення.
      4. Статус 5xx Server Error: Клієнт відображає стандартне повідомлення про помилку.
4. Видалення запису пароля:
   1. Після введення користувачем логіна для видалення та підтвердження, клієнт спочатку шифрує цей логін локально за допомогою майстер-ключа.
   2. Потім виконується DELETE-запит на сервер за шляхом /api/passwords.
   3. У тілі запиту передається об'єкт { encryptedLogin: "зашифрований\_логін\_для\_видалення" }.
   4. Обробка відповіді:
      1. Статус 200 OK (або 204 No Content): Пароль успішно видалено. Клієнт відображає повідомлення про успіх.
      2. Статус 400 Bad Request: Некоректні вхідні дані. Клієнт відображає повідомлення про помилку.
      3. Статус 404 Not Found: Пароль з таким (зашифрованим) логіном не знайдено. Клієнт відображає відповідне повідомлення.
      4. Статус 5xx Server Error: Клієнт відображає стандартне повідомлення про помилку.

**Розділ 6. Розробка бази даних для збереження та отримання інформації**

1. Обґрунтування вибору СКБД: Було обрано реляційну систему керування базами даних (СКБД) MySQL. Вибір обґрунтований її надійністю, широкою розповсюдженістю, доброю підтримкою спільнотою та достатньою продуктивністю для потреб "Генератора Паролів". Структура даних програми є простою та чітко визначеною, що добре узгоджується з реляційною моделлю.
2. Схема бази даних складається з двох таблиць: passwords та validation\_marker.

Таблиця passwords:

* id (INT, PK, AI)
* encrypted\_login (TEXT, NN, UQ)
* encrypted\_password (TEXT, NN)

Таблиця validation\_marker:

* id (INT, PK, AI)
* marker\_key (VARCHAR(50), UQ, NN)
* encrypted\_value (TEXT, NN)

(Де PK - Primary Key, AI - Auto Increment, NN - Not Null, UQ - Unique)

A black and white image of a person

AI-generated content may be incorrect.

Рисунок 6 - Схема бази даних

(На рисунку зображено дві таблиці без прямих зв'язків зовнішніми ключами між ними).

1. Створені індекси:
   1. Таблиця passwords, індекс за полем encrypted\_login (тип UNIQUE INDEX). Використовується для забезпечення унікальності зашифрованих логінів та для прискорення операцій пошуку записів (наприклад, при видаленні або перевірці на дублікати перед вставкою).
   2. Таблиця validation\_marker, індекс за полем marker\_key (тип UNIQUE INDEX). Використовується для швидкого доступу до єдиного запису маркера валідації та забезпечення його унікальності.

(Первинні ключі id в обох таблицях автоматично індексуються СКБД MySQL).

**Розділ 7. Інтеграція сторонніх сервісів в систему**

У програмі "Генератор Паролів" не передбачено інтеграції зі сторонніми сервісам (система не розгортається, шифрування оригінальне, а база даних локальна), через що, як вказано у інструкції, ця частина не виконується.

**Розділ 8. Створення серверної архітектури додатку**

Для розробки серверної частини програми було обрано монолітну архітектуру. Такий підхід був обраний через відносно невеликий та чітко визначений набір функціональних вимог, що дозволяє спростити розробку серверного додатку. Всі серверні компоненти працюють в межах одного процесу Node.js.

Серверний додаток реалізований з використанням фреймворку Express.js і відповідає за наступні основні функції:

1. Обробка HTTP-запитів від клієнтської частини.
2. Валідація майстер-ключа через взаємодію з маркером у базі даних.
3. Збереження, отримання та видалення зашифрованих записів паролів з бази даних MySQL.

Основні модулі/компоненти серверної частини:

1. Головний файл сервера (server.js): Ініціалізація Express-додатку, налаштування проміжного програмного забезпечення (middleware) для обробки запитів (наприклад, cors, express.json), визначення маршрутів API, запуск HTTP-сервера.
2. Модуль маршрутизації API (вбудований в server.js): Визначення ендпоінтів (/api/validate-key, /api/passwords) та прив'язка їх до відповідних функцій-обробників.
3. Функції-обробники API запитів (в server.js): Обробка логіки для кожного ендпоінта. Включає отримання даних із запиту, взаємодію з модулем доступу до БД та модулем криптографії (для маркера), формування та відправлення HTTP-відповіді клієнту.
4. Модуль доступу до бази даних (server/db.js): Налаштування з'єднання з MySQL (створення пулу з'єднань), надання функцій для виконання SQL-запитів (INSERT, SELECT, DELETE), ініціалізація схеми бази даних (створення таблиць, якщо вони не існують).
5. Модуль криптографії (server/cryptoService.js): На сервері використовується для шифрування/дешифрування маркера валідації майстер-ключа. Логіка ідентична клієнтському крипто-сервісу.

Схема основних компонентів серверної частини:

A group of white rectangular labels

AI-generated content may be incorrect.

Рисунок 8 - Діаграма компонентів серверної частини