1 ВСТУП

1.1 Огляд продукту

Розроблювана система призначена для автоматичної генерацiї вебзастосунків на основi побудованих користувачем ER-дiаграм. Програмний продукт включає засоби візуального проектування структури бази даних, конфігурації зовнішнього вигляду, логіки валідації форм, а також інструменти для експорту повнофункціонального вебзастосунку з клієнтською і серверною частинами.

1.2 Мета

Метою продукту є спрощення процесу створення вебзастосунків для користувачів без глибоких технічних знань через автоматизацію генерації структури проєкту за ER-діаграмою.

1.3 Межі

Серверна частина:

– розроблена з використанням мови програмування C# та фреймворку ASP.NET Core;

– реалізоване REST API для обміну даними між клієнтською частиною та сервером;

– використовується MS SQL Server для зберігання структурованих даних (користувачі, проєкти, таблиці, колонки);

– передбачено збереження конфігурацій та ER-діаграм у MongoDB як гнучкі JSON-структури;

– реалізована система автентифікації та авторизації користувачів із підтримкою JWT і ролей (адміністратор, користувач);

– підтримується модульна генерація серверної логіки, DTO, моделей та контролерів з CRUD-функціональністю;

– забезпечено автоматичну генерацію архіву готового застосунку у форматі ZIP;

Клієнтська частина:

– реалізована за допомогою React.js та TypeScript;

– надає графічний редактор ER-діаграм із підтримкою drag-and-dro;

1.4 Посилання

– <https://learn.microsoft.com/aspnet/core/>

– <https://react.dev/>

– <https://www.mongodb.com/docs/>

– <https://learn.microsoft.com/ef/core/>

1.5 Означення та абревіатури

Користувач – особа, яка взаємодіє з програмною системою через веб-інтерфейс або пристрої введення даних (сенсори). Може мати одну з ролей: адміністратор, технічний персонал або звичайний користувач.

Сервер – програмна система, що обробляє запити, виконує бізнес-логіку системи та зберігає інформацію у базі даних.

API (Application Programming Interface) – інтерфейс програмування застосунків, через який інші системи та пристрої взаємодіють із сервером.

CRUD (Create, Read, Update, Delete) – базові операції для створення, зчитування, оновлення та видалення даних у системі.

REST (Representational State Transfer) – архітектурний стиль для розробки веб-сервісів, використаний у створенні серверного API.

ER (Entity-Relationship) – діаграма — графічне зображення даних та їхніх взаємозв'язків в базі даних.

DTO (Data Transfer Object) — об'єкт, який використовується для передачі даних між різними частинами програми або різними системами.

ORM (Object-Relational Mapping) — технологія програмування, яка зв'язує бази даних з концепціями об'єктно-орієнтованих мов програмування, створюючи віртуальну об'єктну базу даних.

JWT (JSON Web Token) — відкритий стандарт для безпечної передачі інформації між сторонами у вигляді об'єкта JSON.

DB (Database) – база даних PostgreSQL, яка зберігає всю інформацію про користувачів, рослини, сенсори та зчитування.

2 ЗАГАЛЬНИЙ ОПИС

2.1 Перспективи продукту

Основні перспективи продукту:

– розширення підтримки нових баз даних (наприклад, PostgreSQL, Firebase);

– додавання нових шаблонів інтерфейсу та тем оформлення;

– інтеграція з хмарними сервісами;

– створення спільноти користувачів для обміну шаблонами, конфігураціями та діаграмами;

2.2 Функції продукту

Користувачі матимуть такі функції:

– реєстрація;

– аутентифікація;

– створення та редагування проєктів;

– побудова ER-діаграм за допомогою графічного редактора;

– конфігурація структури таблиць, включаючи фільтрацію, сортування, назви полів;

– додавання правил валідації до форм введення;

– вибір шаблону інтерфейсу для згенерованого застосунку;

– генерація повноцінного вебзастосунку у вигляді архіву з кодом;

– завантаження та збереження проєктів у систему;

– перегляд і керування попередніми конфігураціями;

2.3 Характеристики користувачів

2.3.1 Адміністратори

– відповідають за управління користувачами та ролями;

– добре орієнтуються в принципах розгортання вебзастосунків та безпеці;

– працюють із файлами конфігурацій та мають розширені технічні знання;

– використовують платформу для створення шаблонів, навчальних матеріалів або демо-проєктів;

2.3.2 Технічний персонал

– використовують платформу як генератор для початкових шаблонів проєктів;

– розширюють згенерований код вручну або інтегрують його у великі рішення;

– очікують зручного експорту, гнучкої конфігурації та прозорої логіки коду;

– мають базові знання JavaScript, C# або SQL.

2.3.3 Кінцеві користувачі (агровиробники або фермери)

– користуються платформою для створення MVP або навчальних прототипів;

– не завжди мають технічні навички, але хочуть отримати робочий вебзастосунок;

– очікують простий інтерфейс, візуальні підказки, автоматизацію процесів.

2.4 Загальні обмеження

На даному етапі присутні такі обмеження:

– реалізований лише веб-інтерфейс, мобільний застосунок наразі не передбачено;

– генерація застосунку підтримується тільки для CRUD-логіки, без складних бізнес-процесів;

– користувач має мати обліковий запис для доступу до функцій генерації.

2.5 Припущення й залежності

На даному етапі припущення та залежності такі:

– користувач працює з ПК або ноутбука з сучасним браузером;

– клієнтська частина залежить від стабільного підключення до інтернету для взаємодії з API;

– серверна частина має бути розгорнута у середовищі, сумісному з .NET Core 6+ та MS SQL Server;

– збереження файлів (архівів) залежить від файлової системи, на якій розгорнуто сервер.

3 КОНКРЕТНІ ВИМОГИ

3.1 Вимоги до зовнішніх інтерфейсів

3.1.1 Інтерфейс користувача

Користувач має такий інтерфейс:

– веб-інтерфейс, розроблений з використанням React.js (SPA);

– інтеграція з REST API для взаємодії із сервером;

– оступні ресурси: створення проєктів, побудова ER-діаграм, генерація застосунку, керування обліковим записом.

3.1.2 Апаратний інтерфейс

Вимоги до апаратного інтерфейсу:

– будь-який комп’ютер або ноутбук із підтримкою сучасного веббраузера;

– мінімум 4 ГБ оперативної пам’яті;

– 1 ГБ вільного місця для збереження згенерованих архівів та проєктів.

3.1.3 Програмний інтерфейс

Серверна частина:

– реалізована на базі ASP.NET Core, C#;

– RESTful API для взаємодії з клієнтом;

– зберігання даних у MS SQL Server та MongoDB.

3.1.4 Комунікаційний протокол

Для безпечної передачі даних між сенсорами, клієнтською та серверною частинами використовуватиметься протокол HTTPS.

Для управління сесіями користувачів та забезпечення контролю доступу використовується JWT. Це дозволяє реалізувати безпечну авторизацію з можливістю масштабування.

3.1.5 Обмеження пам’яті

Для коректної роботи програмного забезпечення діють такі обмеження:

– серверна частина потребує приблизно 1 ГБ вільного місця на диску;

– усі збережені конфігурації та ER-діаграми зберігаються як JSON-об’єкти у MongoDB.

3.1.6 Операції

Основні операції програмної системи:

– реєстрація/вхід користувача;

– створення, збереження, редагування проєктів;

– побудова ER-діаграми;

– конфігурація таблиць, полів, зв’язків;

– налаштування валідації та авторизації;

– генерація застосунку;

– завантаження архіву з кодом.

3.1.7 Функції продукту

Програмне забезпечення має такі функції:

– візуальне моделювання структури бази даних;

– визначення логіки валідації форм;

– формування архіву з вебзастосунком;

– керування користувачами (реєстрація, авторизація);

– перегляд і управління створеними проєктами.

3.1.8 Припущення й залежності

Припущення:

– користувач має базові навички роботи з вебінтерфейсами;

– користувач має доступ до стабільного інтернет-з’єднання;

– середовище для розгортання підтримує .NET Core і Node.js;

Залежності:

– залежність від стабільної роботи серверної частини для генерації проєктів;

– залежність від сумісності з браузером користувача (Chrome, Firefox тощо);

– залежність від коректної роботи MongoDB та MS SQL Server;

– залежність від актуальності сторонніх бібліотек та фреймворків (React, ASP.NET Core, EF).

3.2 Властивості програмного продукту

Програмний продукт забезпечує генерацію повноцінного вебзастосунку на основі побудованої користувачем ER-діаграми з можливістю налаштування структури, вигляду та логіки роботи інтерфейс.

Основні властивості продукту:

– користувачі можуть створювати й редагувати структуру таблиць, визначати типи полів і зв’язки між сутностями;

– платформа підтримує конфігурацію правил валідації (наприклад, обов’язковість, формат введення);

– є можливість налаштовувати зовнішній вигляд форм введення;

– застосунок генерується у вигляді архіву з готовим кодом (клієнт+сервер);

– користувач може зберігати проєкти, повторно завантажувати й редагувати їх;

– забезпечено захист доступу до функціоналу через JWT та HTTPS;

– реалізована двомовність інтерфейсу (українська / англійська).

3.3 Атрибути програмного продукту

3.3.1 Надійність

Ціль – система має стабільно виконувати генерацію коду та обробку конфігурацій без помилок.

Метрика – не більше 0.1% збоїв на місяць при типовому навантаженні.

3.3.2 Доступність

Ціль – платформа повинна бути доступною в будь-який момент для зареєстрованих користувачів.

Метрики:

– доступність 99.9% на місяць;

– підтримка роботи з будь-якого сучасного браузера.

3.3.3 Безпека

Ціль – забезпечення конфіденційності та цілісності проєктів і облікових записів.

Метрики:

– паролі зберігаються у хешованому вигляді;

– автентифікація відбувається з використанням JWT;

– взаємодія з API лише через HTTPS.

3.3.4 Супроводжуваність

Ціль – система повинна бути легкою для оновлення, тестування і масштабування.

Метрики:

– модульна архітектура;

– використання відкритих стандартів (React, .NET, REST API).

3.3.5 Переносимість

Ціль – можливість розгортання серверної частини на різних середовищах (локальних або хмарних).

Метрики:

– використання кросплатформеного .NET Core;

– конфігурації через змінні середовища;

3.3.6 Продуктивність

Ціль – генерація проєкту має виконуватись швидко і без затримок.

Метрики:

– середній час генерації — до 5 секунд на типовий проєкт;

– підтримка роботи одночасно для щонайменше 100 користувачів;

3.4 Вимоги бази даних

Тип баз даних:

– **MS SQL Server** – для зберігання інформації про користувачів, проєкти, таблиці та поля;

– **MongoDB** – для збереження JSON-конфігурацій ER-діаграм, шаблонів і налаштувань генерації.

Призначення – зберігання структурованих даних системи, зокрема:

– облікові записи користувачів, їхні налаштування;

– дані про структуру бази для генерації застосунку;

– історія редагувань проєктів.

3.5 Інші вимоги

Інші вимоги включають:

– документація API за допомогою Swagger;

– підтримка темної/світлої теми оформлення;