**Software Requirements Specification for**

**<** **Веб-застосунок для генерації коду програмної системи на підставі файлу розмітки «Salmon Dreams»>**

**Version 1.0 approved**

**Prepared by <Vyshniak Viktoriia, Zhuravel Yevhenii>**

**<PZPI-21-5>**

**<04.06.2025>**

**ЗМІСТ**

[1 ВСТУП](#__RefHeading___Toc3267_16749813) 3

[1.1 Огляд продукту](#__RefHeading___Toc3269_16749813) 3

[1.2 Мета 4](#__RefHeading___Toc3271_16749813)

[1.3 Межі 4](#__RefHeading___Toc3273_16749813)

[1.4 Посилання 5](#__RefHeading___Toc3275_16749813)

[1.5 Означення та абревіатури 5](#__RefHeading___Toc3277_16749813)

[2 ЗАГАЛЬНИЙ ОПИС 6](#__RefHeading___Toc3279_16749813)

[2.1 Перспективи продукту 6](#__RefHeading___Toc3281_16749813)

[2.2 Функції продукту 6](#__RefHeading___Toc3283_16749813)

[2.3 Характеристики користувачів 8](#__RefHeading___Toc3285_16749813)

[2.4 Загальні обмеження 9](#__RefHeading___Toc3287_16749813)

[2.5 Припущення й залежності 9](#__RefHeading___Toc3289_16749813)

[3 КОНКРЕТНІ ВИМОГИ 11](#__RefHeading___Toc3291_16749813)

[3.1 Вимоги до зовнішніх інтерфейсів 11](#__RefHeading___Toc3293_16749813)

[3.1.1 Інтерфейс користувача 11](#__RefHeading___Toc3295_16749813)

[3.1.2 Апаратний інтерфейс 11](#__RefHeading___Toc3297_16749813)

[3.1.3 Програмний інтерфейс 12](#__RefHeading___Toc3299_16749813)

[3.1.4 Комунікаційний протокол 12](#__RefHeading___Toc3301_16749813)

[3.1.5 Обмеження пам’яті 12](#__RefHeading___Toc3516_16749813)

[3.1.6 Операції 13](#__RefHeading___Toc3518_16749813)

[3.2 Властивості програмного продукту 13](#__RefHeading___Toc3520_16749813)

[3.3 Атрибути програмного продукту 14](#__RefHeading___Toc3522_16749813)

[3.3.1 Надійність 14](#__RefHeading___Toc3524_16749813)

[3.3.2 Масштабованість 14](#__RefHeading___Toc3526_16749813)

[3.3.3 Відкрита до відлагодження 14](#__RefHeading___Toc3528_16749813)

[3.3.4 Адаптивність 14](#__RefHeading___Toc3530_16749813)

[3.3.5 Швидкодія 14](#__RefHeading___Toc3532_16749813)

[3.3.6 Підтримка багатьох браузерів](#__RefHeading___Toc3534_16749813) 14

[3.3.7 Безпека даних](#__RefHeading___Toc3536_16749813) 15

[3.3.8 Робота з одночасним доступом](#__RefHeading___Toc3538_16749813) 15

[3.3.9 Захист від помилок](#__RefHeading___Toc3540_16749813) 15

[3.4 Вимоги бази даних 15](#__RefHeading___Toc3542_16749813)

**1 ВСТУП**

* 1. Огляд продукту

У сучасному світі розробка програмного забезпечення є складним та багаторівневим процесом, що вимагає значних ресурсів, часу та координації. Команди розробників часто стикаються з необхідністю створення типових компонентів, таких як модулі авторизації, управління користувачами, інтеграційні сервіси тощо. Це призводить до повторюваної роботи та зниження ефективності.

Наявні рішення, такі як фреймворки, бібліотеки або платформи з низьким кодом (low-code/no-code), частково вирішують ці проблеми. Проте вони часто обмежені у гнучкості та потребують додаткової адаптації для реалізації специфічних вимог бізнесу. Це може ускладнити інтеграцію та збільшити витрати на розробку.

Для вирішення цієї проблеми виникла необхідність у створенні інструменту, який би дозволяв автоматизувати процес генерації програмного забезпечення на ранніх етапах розробки, а саме в побудові системи, що на основі вхідного файлу розмітки (який містить опис сутностей, їх взаємозв’язків, правил авторизації та основної логіки) автоматично формує архів з готовим проєктом. Користувач описуватиме ключові сутності, їх взаємозв’язки, правила авторизації та інші параметри у зручному форматі. На основі цієї інформації система автоматично генеруватиме архів із повноцінним проєктом, що включатиме всі необхідні компоненти. Гнучка архітектура дозволятиме розробникам обирати та налаштовувати модулі відповідно до своїх потреб, забезпечуючи швидке отримання робочого прототипу.

Як результат, реалізація цього проєкту забезпечить розробників інструментом для швидкої генерації програмних систем. Це дозволить значно скоротити час на розробку стандартних функцій, покращити якість коду завдяки використанню перевірених модулів та підвищити продуктивність роботи команд. Крім того, система відкриває можливості для спільноти розробників, де кожен може запропонувати власні системні модулі та сприяти розвитку платформи. Такий підхід сприятиме зниженню витрат на розробку, зменшенню кількості помилок під час написання коду та прискоренню виходу продуктів на ринок.

Галуззю застосування розробленої системи є автоматизація процесу створення MVP-продуктів та програмних проєктів із базовою функціональністю.

* 1. Мета

Метою проєкту є створення веб-застосунку для генерації коду програмної системи на підставі файлу розмітки, що спростить та прискорить процес розробки типових застосунків. Система має забезпечити стандартизований підхід до створення структури програмних рішень, зменшити обсяг рутинної роботи та підвищити ефективність команд розробників.

Система орієнтована на широке коло користувачів – від технічних спеціалістів до представників бізнесу – і дозволить зосередитись на реалізації бізнес-логіки, мінімізуючи витрати на проєктування базових компонентів. Передбачається, що рішення матиме гнучку архітектуру з можливістю масштабування та подальшого розширення, що дозволить адаптувати його під потреби різних проєктів.

Загальна концепція системи передбачає наявність окремих компонентів для обробки вхідних даних і формування програмного коду, що дозволить ефективно розділити відповідальність між етапами генерації та забезпечити стабільність і гнучкість у процесі подальшого розвитку платформи.

* 1. Межі

У рамках цієї роботи буде реалізовано клієнт-серверний застосунок, що складається з двох основних частин: серверної (бекенд) та клієнтської (фронтенд). Основна функціональність, пов’язана з обробкою вхідних даних, логікою генерації програмного коду та керуванням користувачами, зосереджена на стороні сервера. Клієнтська частина відповідає за взаємодію з користувачем та передачу даних до сервера.

Серверна частина буде реалізована як Web API-застосунок на базі ASP.NET Core з використанням мови програмування C# та платформи .NET 8. Для забезпечення авторизації буде використовуватись механізм JWT (JSON Web Token) із симетричним шифруванням. Обробка запитів до бази даних здійснюватиметься за допомогою ORM Entity Framework Core, а в якості системи керування базами даних обрана MS SQL Server.

Клієнтська частина реалізується на Angular 18 із використанням HTML, SCSS та TypeScript. Вона забезпечуватиме інтерфейс користувача для опису вхідних параметрів, взаємодії з системою та перегляду результатів генерації.

* 1. Посилання

Документ посилається на стандарт ДСТУ 3008:2015 «Документація. Звіти у сфері наукових досліджень та розробок. Структура і правила оформлення». Текст має офіційний стиль, IEEE Std 830-1993 «Рекомендована практика IEEE для специфікацій вимог до програмного забезпечення».

* 1. Означення та абревіатури

DI – Dependency Injection

ПЗ – програмне забезпечення

DTO – Data Transfer Object

ППЗ – Проміжне Програмне Забезпечення

БД – база даних

СУБД – система управління базами даних

ORM – Object Relational Mapping

1. **ЗАГАЛЬНИЙ ОПИС**
   1. Перспективи продукту

Розроблена система має значний потенціал для подальшого розвитку та комерційного впровадження. Вона може стати незамінним інструментом для стартапів, яким необхідно швидко протестувати бізнес-ідею або вивести мінімально життєздатний продукт (MVP) на ринок без значних витрат та залучення великої команди розробників.

Крім того, система ефективно застосовується в освітньому процесі для наочної демонстрації базових принципів побудови архітектури програмних систем та впровадження шаблонів проєктування. Це робить її цінною для викладачів і студентів у сфері комп’ютерних наук та програмної інженерії.

У корпоративному середовищі система може бути використана для стандартизації та уніфікації початкової структури програмних проєктів, що сприятиме підвищенню ефективності командної розробки та підтримці єдиних підходів до архітектури.

Монетизація продукту передбачена через впровадження системи підписок, які надаватимуть користувачам доступ до генерації проєктів та додаткових функцій. У перспективі планується розширення функціональності платформи: буде збільшено кількість підтримуваних технологічних стеків, архітектурних шаблонів, а також додано підтримку нових фреймворків, мов програмування та систем управління базами даних.

* 1. Функції продукту

Бізнес-логіка «SalmonDreams» ґрунтується на ряді операцій, що включають в себе функціонал для користувачів, які бажають згенерувати готове програмне рішення, та адміністраторів системи.

Функціонал серверної частини:

* робота з сесіями користувачів;
* генерація архіву програми;
* генерація проєкту за допомогою мови розмітки (YAML, JSON);
* генерація проєкту за допомогою темплейтів;
* генерація проєкту за допомогою білдера;
* можливість переглядати дашборди з основними метриками системи;
* генерацію ASP.NET Web API-застосунку на мові програмування C# з використанням проміжного програмного забезпечення та сучасних DI-контейнерів;
* генерацію сутностей бази даних;
* побудову логічних зв’язків між сутностями («один-до-одного», «один-до-багатьох» та «багато-до-багатьох») з відповідним налаштуванням ORM;
* генерацію коду доступу до бази даних на основі ORM (Entity Framework) з використанням СУБД (MS SQL Server);
* генерацію бізнес-логіки з реалізацією CRUD-операцій для кожної сутності;
* генерацію DTO;
* генерацію API-контролерів;
* формування OpenAPI-специфікації з використанням Swagger UI для візуалізації;
* генерацію механізмів автентифікації та авторизації з використанням JWT;
* генерацію та реєстрацію Automapper-профілів, необхідних для коректного перетворення між сутностями та DTO.

Функціонал клієнтської частини:

* можливість генерувати програмну систему за допомогою мови розмітки (YAML, JSON);
* можливість генерувати програмну систему за допомогою темплейтів;
* можливість генерувати програмну систему за допомогою білдера запитів.

У майбутніх релізах буде додано підтримку інших мов програмування, СУБД, інших типів архітектури, створення повноцінного клієнт-серверного застосунку, створення користувацьких шаблонів, підтримки більшої кількості механізмів автентифікації та авторизації, підтримки CI/CD.

* 1. Характеристики користувачів

Таблиця 2.1 – Користувачі системи

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Користувач** | **Головна цінність** | **Ставлення** | **Головний інтерес** | **Обмеження** |
| Розробник ПЗ | Автоматизація рутинних завдань, швидке створення каркасу проєкту | Позитивне. Очікується скорочення часу на розробку та зменшення помилок | Генерація якісного та підтримуваного коду | Потрібне правильне налаштування шаблонів та підтримка актуальних технологій |
| Технічний лідер/архітектор | Впровадження стандартизованого підходу до розробки проєктів | Позитивне. Очікується зниження технічного боргу та спрощення онбордингу нових розробників | Генерація коду відповідно до архітектурних принципів | Обмежена гнучкість у випадку нестандартних вимог |
| Проєктний менеджер | Скорочення часу на старт нових проєктів | Позитивне. Очікується підвищення продуктивності команди | Зниження витрат часу на початкову розробку | Необхідність навчання команди користуванню системою |
| DevOps інженер | Стабільність та відповідність коду CI/CD процесам | Нейтральне. Очікується відповідність стандартам розгортання | Інтеграція з існуючими інструментами CI/CD | Потрібна сумісність із поточним стеком технологій |
| Клієнт (замовник ПЗ) | Зменшення витрат на розробку MVP | Позитивне. Очікується швидке отримання першої версії продукту | Прискорення процесу розробки | Обмежені можливості кастомізації без втручання розробника |

* 1. Загальні обмеження
* створюється лише новий проєкт, без змін у вже написаному коді;
* система працює лише з файлами у певному форматі (JSON або YAML);
* генерація коду можлива лише для стандартних CRUD-операцій, без підтримки складної бізнес-логіки;
* система орієнтована виключно на серверний код та базу даних.
* система не включає налаштування деплою чи збірки коду.
  1. Припущення й залежності

У межах розробки системи автоматизованої генерації коду було прийнято такі припущення:

* користувачі мають базові знання про розмітку JSON- та YAML-структуру програмних систем;
* система підтримує інтеграцію з популярною мовою програмування C#;
* користувачі мають доступ до інтернету для завантаження модулів із маркетплейсу та інтеграції з зовнішніми сервісами;
* користувачі мають необхідні права доступу для використання зовнішніх API та сервісів;
* система підтримує масштабованість для роботи з великими проєктами та командною розробкою;
* користувач має базові знання синтаксису щодо обраної мови програмування для іменування проєкту та сутностей системи, що генерується;
* користувач має базові технічні знання та здатний самостійно інтегрувати або модифікувати згенерований код у межах власного проєкту.

Щодо залежностей, можна виділити наступні:

* система залежить від доступності зовнішніх бібліотек та фреймворків для генерації коду (наприклад, ASP.NET, Entity Framework тощо);
* система потребує налаштування для підтримки багатьох мов програмування та відповідних інструментів для їх генерації;
* наявність середовища виконання, необхідного для компіляції та запуску створених проєктів;
* наявність завантаженого SDK та СУБД для обраних технологій.

1. **КОНКРЕТНІ ВИМОГИ**
   1. Вимоги до зовнішніх інтерфейсів

3.1.1 Інтерфейс користувача

Інтерфейс програмної системи реалізовано як односторінковий вебзастосунок з використанням Angular. Структура проєкту побудована на принципі розділення компонентів за сторінками: кожна сторінка має власні компоненти, що відповідають за конкретну функціональність. Це забезпечує зручну підтримку та розширюваність коду, а також дозволяє точно контролювати функціональне наповнення кожного розділу.

Для реалізації візуальних елементів використовувалась бібліотека PrimeNG, яка надала готові компоненти, такі як кнопки, таблиці, діалогові вікна та форми. Усі повторно використовувані елементи були винесені до окремого модуля shared, що дозволило уникнути дублювання та забезпечити централізоване оновлення компонентів при зміні вимог до інтерфейсу.

Під час розробки основна увага приділялась реалізації повного функціоналу для досвідчених користувачів, які можуть самостійно формувати розмітку проєктів. Було реалізовано три режими роботи: введення текстової або файл-розмітки, розширений конструктор з модульним деревом і автоматичним формуванням вкладених елементів, а також Template builder – окрема частина інтерфейсу для менш досвідчених користувачів, яка дозволяє генерувати проєкти шляхом заповнення простих форм. Інтерфейс створювався з увагою до адаптивності, але основними пристроями для тестування були комп’ютери з широкоформатними моніторами.

3.1.2 Апаратний інтерфейс

Мінімальні вимоги до апаратного інтерфейсу:

* доступ до мережі Інтернет (wifi або мобільний інтернет);
* операційна система Windows 10 та вище або MacOS для ноутбуків та IOS або Android для смарфонів;
* актуальний браузер (Google Chrome, Opera, Safari, Firefox, Microsoft Edge).

3.1.3 Програмний інтерфейс

Програмний інтерфейс системи реалізований у вигляді REST API, що забезпечує обмін даними між клієнтською та серверною частинами. Користувач може надсилати конфігурації у форматах JSON або YAML для генерації проєктів, використовуючи шаблони, темплейти або білдер-запити. Серверна частина обробляє запити, динамічно створює необхідні сутності, сервіси та контролери, формує структуру застосунку й повертає архів з готовим кодом. API захищене за допомогою JWT-автентифікації. Доступ до даних реалізовано через ORM Entity Framework Core, що забезпечує зручну взаємодію з реляційною СУБД. У перспективі передбачена підтримка користувацьких модулів для розширення можливостей системи.

3.1.4 Комунікаційний протокол

Комунікація між різними компонентами системи здійснюватиметься через протоколи HTTP/HTTPS. Архітектура побудована за клієнт-серверним принципом, де сервер виконує основні функції з управління ресурсами та надання послуг, які використовуються клієнтською частиною.

3.1.5 Обмеження пам’яті

Система не потребує значних обсягів оперативної пам’яті для своєї роботи, проте для стабільної генерації проєктів середньої складності рекомендується мінімум 4 ГБ оперативної пам’яті на сервері. У випадку обробки великих конфігурацій або паралельної роботи декількох користувачів можливе тимчасове збільшення споживання пам’яті до 8 ГБ і більше.

Клієнтська частина є вебзастосунком і не накладає суттєвих вимог до пам’яті на стороні користувача, окрім стандартних вимог сучасного браузера. Для забезпечення швидкодії і уникнення перевантаження пам’яті система використовує механізми оптимізації, такі як очищення тимчасових файлів після генерації, обмеження розміру конфігурацій, а також потокову обробку даних.

3.1.6 Операції

Взаємодія між клієнтом і серверною частиною системи здійснюється через HTTP-протокол. Клієнт надсилає запит до API, після чого отримує відповідь у форматі, що відповідає запиту (наприклад, JSON або YAML). Для доступу до ресурсів використовуються URL-адреси, які слугують точками входу до відповідних функціональних можливостей системи.

Кожен URL-ресурс супроводжується HTTP-методом, який визначає тип операції, що виконується над ресурсом. Основні операції:

* GET – використовується для отримання інформації або даних з сервера;
* POST – дозволяє створювати нові ресурси;
* PUT – забезпечує повне оновлення наявного ресурсу;
* PATCH – виконує часткове оновлення даних;
* DELETE – видаляє вказаний ресурс із системи.

3.2 Властивості програмного продукту

Програмний продукт побудований із використанням сучасного технологічного стеку, що забезпечує швидкість розробки, масштабованість і зручність супроводу. Основний акцент зроблено на модульність і гнучкість: система підтримує генерацію програмних проєктів на основі шаблонів або розмітки, динамічне створення компонентів (моделі, сервіси, контролери), а також можливість розширення за рахунок підключення нових фреймворків. Архітектура побудована за принципами чистого коду та з використанням REST API, що гарантує простоту інтеграції з іншими системами. Продукт орієнтований як на індивідуальне використання, так і на командну розробку, з перспективою подальшого розширення функціональності.

3.3 Атрибути програмного продукту

3.3.1 Надійність

Програмна система повинна мати можливість відновлення з негативного стану та бути доступною для користувачів 99% часу.

3.3.2 Масштабованість

Програмна система повинна мати відкриті точки розширення для подальшого легкого розширення. Точки розширення мають дозволяти створення атомарних одиниць розширення для мінімальної зміни вже існуючої системи.

3.3.3 Відкрита до відлагодження

Система повинна мати можливість збору даних про свій стан для подальшого їх аналізу розробниками або аналітиками. При помилці, система повинна надавати вичерпний опис помилки з даними, яких повинно вистачити для ідентифікації такої самої помилки в майбутньому та її відтворення для подальшого вирішення.

3.3.4 Адаптивність

Інтерфейс системи повинен бути спроєктований для досягнення адаптивності на різних екранах, а саме на мобільних пристроях, планшетах та комп’ютерах.

3.3.5 Швидкодія

Система повинна підтримувати механізми збереження даних для подальшого їх перевикористання при потребі здобуття цих даних шляхом високонавантажених дій.

3.3.6 Підтримка багатьох браузерів

Програмна система має бути доступною та працювати однаковим чином на всіх сучасних браузерах.

3.3.7 Безпека даних

Програмна система повинна зберігати лише необхідну для ідентифікації користувача приватну інформацію, мінімізуючи обсяг персональних даних. Уся критична інформація зберігається в зашифрованому або захешованому вигляді. Для шифрування даних використовується стійкий алгоритм Blowfish, який забезпечує високий рівень безпеки при зберіганні конфіденційної інформації. Для автентифікації та авторизації користувачів застосовується JWT (JSON Web Token), що дозволяє реалізувати безпечну безстейтову взаємодію між клієнтом і сервером, а також ефективно контролювати доступ до ресурсів системи.

3.3.8 Робота з одночасним доступом

Програмна система повинна мати можливість обробки паралельних запитів без впливу запитів на результат один одного, якщо це не пряма дія спрямована на зміну стану системи.

3.3.9 Захист від помилок

Частини інтерфейсу програмної системи повинні бути доступними лише при достатній кількості введених даних для запобігання подальших помилок при обробці цих даних.

3.4 Вимоги бази даних

У процесі розробки системи було обрано систему управління базами даних Microsoft SQL Server, оскільки вона забезпечує високу продуктивність, добру масштабованість, а також має відмінну сумісність із середовищем розробки. Крім того, ця СУБД гарантує надійне та безпечне зберігання даних.

Для реалізації моделі даних у межах проєкту було використано реляційний підхід, що передбачає представлення об'єктів предметної області у вигляді таблиць з можливістю встановлення між ними зв’язків через унікальні ідентифікатори — первинні та зовнішні ключі. Така структура дозволяє ефективно зберігати та обробляти структуровану інформацію, що позитивно впливає на загальну продуктивність системи.

Для взаємодії з базою даних було обрано ORM-фреймворк Entity Framework Core. Його застосування спрощує доступ до даних за рахунок об’єктно-орієнтованого підходу, забезпечує захист від SQL-ін’єкцій та знижує ймовірність помилок завдяки використанню дерев виразів при формуванні запитів.