Специфікація програмного забезпечення

«Програмної системи для зберігання, спільного доступу та перевірки якості програмного коду»

Автори:

студенти гр. ПЗПІ-21-4

Щоголєв С. А.

Павленко П. О.

Білодід М. О.

Харків 2025

1 Вступ

1.1 Огляд проєкту

У сучасній сфері розробки програмного забезпечення команди розробників щодня стикаються з проблемою ефективного зберігання, пошуку, повторного використання та перевірки якості програмного коду. Зростання складності проєктів, збільшення кількості мікросервісів, розподілених команд і часта зміна складу фахівців призводять до того, що значна частина напрацьованого коду або втрачається, або не використовується повторно, а команда витрачає багато часу на вирішення вже знайдених кимось раніше задач. Традиційний підхід до організації кодових активів за допомогою особистих сховищ, неповних репозиторіїв чи локальних файлів не забезпечує централізованого доступу, не дозволяє швидко знаходити потрібний фрагмент і ускладнює колективну роботу. Часто навіть наявність Git-репозиторіїв не вирішує проблеми пошуку фрагментів, оскільки відсутня гнучка система каталогізації, тегування, аналізу і контролю якості. Як наслідок, розробники вимушені багаторазово писати однотипний код, дублювати функціонал, а це призводить до втрати часу, накопичення технічного боргу і зниження продуктивності команди.

У таких умовах виникає нагальна потреба у створенні програмної системи, яка забезпечить централізоване зберігання, зручний пошук, спільний доступ та автоматизовану перевірку якості програмного коду. Така система повинна допомагати як окремим розробникам, так і організаціям зберігати напрацьовані рішення, уникати дублювання логіки, пришвидшувати онбординг нових членів команди, формувати єдиний стандарт корпоративної розробки, а також підвищувати рівень безпеки та захисту внутрішньої інтелектуальної власності компанії. Особливої актуальності набуває підтримка спільної роботи над кодом із гнучким керуванням правами доступу, можливістю створення командних і корпоративних колекцій фрагментів, організацією комунікації та обміну знаннями всередині компанії.

Водночас, сучасні технології відкривають нові можливості для вирішення цих завдань. Використання інструментів штучного інтелекту, таких як великі мовні моделі (LLM) і embedding-механізми, дозволяє реалізувати інтелектуальний пошук не лише за ключовими словами, а й за семантикою та логічною схожістю коду. Автоматизований аналіз якості фрагментів на відповідність принципам DRY, SOLID, контроль наявності дублікатів, рекомендації щодо покращення структури — усе це дозволяє команді підтримувати високий рівень якості продукту, зменшувати ризики помилок і оптимізувати процес розробки.

Програмна система, яку передбачається створити, орієнтована на широкий спектр користувачів — від індивідуальних розробників і студентів до команд великих організацій, що працюють у корпоративному або аутсорсинговому форматі. Вона дозволяє організувати повний цикл роботи з фрагментами коду: додавання, зберігання, каталогізація, семантичний пошук, автоматична перевірка якості, спільне використання та генерація нових проєктів на основі вже перевірених рішень із можливістю експорту структури у вигляді архіву. Враховуючи потреби сучасних компаній, система також інтегрується з зовнішніми сервісами — OpenAI API, SMTP для email-повідомлень, платформами DevOps для автоматичного тестування і розгортання, підтримує багаторівневу архітектуру з хмарним зберіганням, масштабуванням і сучасними вимогами до інформаційної безпеки.

Розробка такої системи стане дієвим інструментом підвищення ефективності роботи ІТ-команд, дозволить зберігати й розвивати корпоративні знання, мінімізувати дублювання зусиль і підвищити стандарти якості програмного забезпечення, що випускається.

1.2 Мета

Метою створення програмної системи для зберігання, спільного доступу та перевірки якості програмного коду є підвищення ефективності, продуктивності й якості розробки програмного забезпечення. Система має забезпечити розробникам централізоване зберігання та швидкий пошук фрагментів коду, зручний механізм їх повторного використання, а також безпечний спільний доступ у команді або організації. Інтеграція автоматизованої перевірки якості дозволить виявляти дублювання та порушення стандартів коду, підвищити підтримуваність проєктів і скоротити час на рев’ю.

Важливим завданням є створення платформи для накопичення, організації та передачі знань, що дозволяє зберігати напрацьовані рішення, полегшувати старт нових проєктів і спрощувати адаптацію нових членів команди. Упровадження сучасних технологій — семантичного пошуку, елементів штучного інтелекту й інтеграцій із хмарними сервісами — сприятиме автоматизації рутинних процесів і мінімізації технічного боргу.

Кінцева мета — надати надійний інструмент для підтримки високої якості коду, організації командної роботи та підвищення конкурентоспроможності розробницьких команд і організацій.

1.3 Межі

Межі проєкту визначають функціональні можливості системи, її обмеження, а також сфери застосування. Програмна система призначена виключно для роботи з фрагментами програмного коду та пов’язаними з ними метаданими, такими як мова програмування, теги, опис і контекст використання. Система підтримує створення, редагування, пошук, аналіз і спільний доступ до кодових фрагментів, однак не призначена для зберігання або обробки бінарних файлів, графіки чи великих даних, що не є програмним кодом.

В межах проєкту реалізується підтримка основних популярних мов програмування, але розширення цього переліку або інтеграція з екзотичними чи корпоративними мовами не входить до початкового функціоналу. Генерація нових програмних проєктів здійснюється на основі наявних у системі фрагментів, а не шляхом повністю автоматизованого створення унікальних рішень “з нуля”. Робота з кодом та управління доступом до нього здійснюється лише для зареєстрованих користувачів і команд у межах хмарної інфраструктури системи.

Система не охоплює повноцінного редагування коду на рівні IDE, розгортання зовнішніх сервісів або інтеграції з внутрішніми корпоративними системами поза зазначеною інфраструктурою. Крім того, автоматизований аналіз якості коду та семантичний пошук не гарантують повної відсутності помилок чи вразливостей — відповідальність за фінальну перевірку і використання результатів залишається за користувачем.

Таким чином, програмна система має чіткі межі застосування і забезпечує стабільну, безпечну та ефективну роботу саме у сфері організації, зберігання, пошуку та перевірки якості програмного коду для індивідуальних розробників, команд і організацій.

1.4 Посилання

У процесі розробки проєкту використовуються такі джерела та нормативні документи:

* Відомча і міжнародна технічна документація з організації DevOps, хмарної інфраструктури, інструментів зберігання та аналізу коду (GitHub, Azure, OpenAI API);
* Матеріали офіційної документації щодо REST API, React, .NET, CosmosDB, Azure DevOps, а також OpenAI Embeddings;
* Актуальні наукові та технічні статті, присвячені автоматизації пошуку та перевірки коду, впровадженню штучного інтелекту у сфері розробки ПЗ;
* Практичний досвід та best practices, описані у відкритих джерелах, блогах і на професійних форумах для розробників.

2 Загальний опис

2.1 Перспективи програмного забезпечення

Розробка програмної системи для зберігання, спільного доступу та перевірки якості програмного коду відкриває нові можливості для організації ефективної командної роботи й накопичення внутрішніх знань у сфері розробки програмного забезпечення. Використання такого інструменту дозволяє компаніям та розробникам створити централізовану, структуровану й доступну бібліотеку кодових фрагментів, що підвищує швидкість впровадження нових проєктів, знижує ризик дублювання коду та втрати напрацьованих рішень.

Система надає можливість інтегрувати найкращі практики та стандарти програмування, впроваджувати автоматизований контроль якості коду та рекомендації щодо його покращення, що позитивно впливає на підтримуваність, масштабованість і безпеку програмних продуктів. Розширення функціоналу за рахунок впровадження штучного інтелекту та семантичного пошуку відкриває новий рівень інтелектуалізації процесів аналізу, підбору та повторного використання коду. Це дозволяє командам мінімізувати витрати часу на пошук рішень та сконцентруватися на розв’язанні складних і творчих задач.

Завдяки хмарній архітектурі та сучасним засобам автоматизації (DevOps, CI/CD) система здатна обслуговувати як малі, так і великі організації, забезпечуючи надійність, гнучкість і високу доступність сервісу для користувачів з будь-якої точки світу. Впровадження такої програмної системи у повсякденну практику командної розробки підвищить конкурентоспроможність компанії, скоротить цикл розробки нових продуктів та сприятиме розвитку внутрішньої корпоративної культури щодо обміну знаннями.

У перспективі система може стати основою для побудови власних внутрішніх стандартів, формування спільнот розробників та розвитку корпоративних баз знань, інтегруючись із зовнішніми сервісами, інструментами аналітики, автоматизованого тестування та керування проєктами. Такий підхід забезпечує сталість розвитку та актуальність системи в умовах постійних змін і зростаючих вимог до сучасних IT-продуктів.

2.2 Функції продукту

* Додавання, редагування та зберігання фрагментів програмного коду з супровідною інформацією (опис, теги, мова програмування, контекст використання).
* Каталогізація та організація фрагментів у особисті, командні та корпоративні колекції.
* Гнучкий пошук фрагментів за ключовими словами, тегами та семантичними критеріями (інтеграція з AI).
* Надання спільного доступу до фрагментів із налаштуванням ролей та прав користувачів.
* Автоматизована перевірка якості коду (аналіз на дублювання, відповідність принципам DRY, SOLID тощо).
* Рекомендації щодо покращення якості, структури та стилю програмного коду.
* Генерація початкових проєктів на основі текстового опису задачі.
* Створення та експорт повної структури проєкту у вигляді архіву для швидкого старту розробки.
* Ведення історії змін і аудит дій з кодовими фрагментами.
* Інтеграція з зовнішніми сервісами: OpenAI API для аналізу та генерації, SMTP для відправки повідомлень, CI/CD для автоматичного тестування й розгортання.
* Захист і безпека даних, аутентифікація користувачів, аудит доступу та захист від несанкціонованих дій.
* Адаптивний, сучасний і зручний веб-інтерфейс для роботи з фрагментами на різних пристроях.

2.3 Характеристики користувачів

Користувачами програмної системи є як окремі спеціалісти, так і цілі команди, що працюють у сфері розробки програмного забезпечення. Основну цільову аудиторію становлять професійні розробники, які мають базові навички роботи з сучасними мовами програмування та розуміють принципи організації коду і проектної роботи. Для таких користувачів система є інструментом для зручного зберігання, швидкого пошуку та повторного використання власних і командних фрагментів коду.

Важливою категорією є технічні керівники, тімліди й архітектори, які відповідають за стандартизацію розробки в організації, контроль якості коду, впровадження найкращих практик і організацію внутрішньої бази знань. Вони активно використовують механізми контролю доступу, управління ролями, аудит змін і перевірку якості фрагментів, забезпечуючи узгодженість і спадковість корпоративних рішень.

Система також орієнтована на командну роботу в малих і середніх організаціях або стартапах, де важливо забезпечити швидке навчання нових учасників, доступ до типових рішень, прикладів та шаблонів. Для студентів, молодих спеціалістів і початківців система виконує освітню функцію, дозволяючи вивчати типові структури коду, аналізувати якість рішень і порівнювати власні напрацювання з перевіреними фрагментами.

Крім того, користувачами можуть бути DevOps-інженери, тестувальники й аналітики, які беруть участь у процесі автоматизації тестування, контролю якості й організації інфраструктури CI/CD, використовуючи відповідний функціонал для інтеграції з іншими корпоративними сервісами.

Для успішного використання продукту користувачам достатньо мати базове розуміння принципів роботи з веб-інтерфейсами, навички реєстрації та аутентифікації у веб-додатках, а також загальні уявлення про структуру програмних проєктів. Платформа не вимагає глибоких знань адміністрування або специфічних технологій, що робить її доступною для широкого кола фахівців у галузі розробки ПЗ.

2.4 Загальні обмеження

Обмеження та вийнятки-1. Система працює лише з текстовими фрагментами програмного коду та супровідною інформацією (опис, теги, мова програмування); зберігання бінарних файлів, зображень чи медіаданих не підтримується.

Обмеження та вийнятки-2. Підтримка мов програмування обмежена переліком, затвердженим при розгортанні системи; розширення цього списку потребує додаткової розробки.

Обмеження та вийнятки-3. Система не забезпечує повноцінної інтеграції з зовнішніми корпоративними чи локальними системами поза визначеною хмарною інфраструктурою (Azure, SMTP тощо).

Обмеження та вийнятки-4. Автоматизована перевірка якості коду та семантичний пошук ґрунтуються на застосованих алгоритмах і моделях, тому не гарантують повної відсутності логічних або архітектурних помилок у фрагментах.

Обмеження та вийнятки-5. Система розгортається на сучасних веб-платформах і потребує для роботи підтримки актуальних версій браузерів і наявності інтернет-з’єднання.

Обмеження та вийнятки-6. Генерація нових проєктів здійснюється лише на основі наявних у системі фрагментів коду; створення унікальних рішень "з нуля" не передбачене.

2.5 Припущення й залежності

Припущення-1. Користувачі мають базові навички роботи з сучасними мовами програмування та розуміють структуру програмних проєктів.

Припущення-2. Користувачі знайомі з принципами командної роботи, ролями та спільного доступу до інформації в ІТ-системах.

Припущення-3. Для реєстрації та використання системи користувачі мають доступ до електронної пошти й можуть підтвердити свою особу.

Припущення-4. Користувачі працюють із актуальними версіями браузерів і мають стабільне підключення до Інтернету.

Залежність-1. Якість результатів автоматичного аналізу коду та семантичного пошуку залежить від налаштувань моделей штучного інтелекту та їхньої актуальності.

Залежність-2. Продуктивність і стабільність системи залежать від обраної хмарної інфраструктури, навантаження на сервер і швидкості Інтернет-з’єднання користувачів.

Залежність-3. Коректність перевірки коду та його повторного використання залежить від якості й повноти введених метаданих та тегів для фрагментів.

Залежність-4. Функціонал інтеграції із зовнішніми сервісами (OpenAI, SMTP, CI/CD) працює коректно лише за умови доступності відповідних API та актуальності ключів доступу.

3 Конкретні вимоги

3.1 Вимоги до зовнішніх інтерфейсів

3.1.1 Інтерфейс користувача

Інтерфейс користувача програмної системи розроблено як сучасний, інтуїтивно зрозумілий веб-додаток із підтримкою адаптивного дизайну. Основна мета — забезпечити швидкий доступ до ключових функцій: додавання, редагування, перегляду та пошуку фрагментів коду, організації спільної роботи, перевірки якості й експорту проєктів. Інтерфейс реалізовано на базі React, що дозволяє динамічно оновлювати вміст без перезавантаження сторінки, а також гарантує високу швидкодію та зручність навігації.

На головній сторінці системи користувач бачить структурований список доступних фрагментів коду з коротким описом, тегами, мовою програмування й іншою супровідною інформацією. Передбачена система фільтрації та пошуку за ключовими словами, тегами, а також можливість виконувати семантичний пошук із використанням елементів штучного інтелекту. Для кожного фрагмента доступний перегляд повного коду, супровідної інформації, а також історії змін і коментарів.

Особистий кабінет користувача дозволяє переглядати й організовувати власні фрагменти, редагувати їх, додавати нові або видаляти неактуальні. Окремий розділ призначений для спільної роботи в команді чи організації — з можливістю керування корпоративними колекціями, ролями, доступом і запрошенням нових учасників. Передбачена система сповіщень про зміни, нові фрагменти та важливі події.

Для зручності користувачів у систему інтегровано редактор коду з підсвічуванням синтаксису, автодоповненням і перевіркою на основні помилки, що значно полегшує внесення змін безпосередньо в інтерфейсі. Інтуїтивна форма додавання нового фрагмента передбачає вказання назви, опису, вибір мови програмування та додавання тегів.

Всі елементи інтерфейсу розташовані логічно та структуровано, що забезпечує просту навігацію й швидкий доступ до необхідного функціоналу. Дизайн інтерфейсу виконано у сучасному стилі з підтримкою світлої та темної тем, адаптацією під різні розміри екрану та пристрої. Система передбачає захист від несанкціонованого доступу, авторизацію користувача та можливість відновлення пароля через електронну пошту.

Загалом, інтерфейс користувача орієнтований на зручність, інтуїтивність і швидкість виконання типових операцій, що дозволяє ефективно використовувати систему як окремим розробникам, так і цілим командам.

3.1.2 Апаратний інтерфейс

Мінімальні вимоги до апаратного інтерфейсу для використання програмної системи включають:

* Операційна система: сучасні дистрибутиви Linux, Windows 10 або вище, або MacOS.
* Доступ до мережі Інтернет для коректної передачі та синхронізації даних з сервером.
* Не менше 4 ГБ оперативної пам’яті для стабільної та плавної роботи веб-інтерфейсу та виконання основних операцій.
* Актуальна версія одного з підтримуваних веб-браузерів: Google Chrome, Mozilla Firefox, Microsoft Edge, Opera або Safari.

Забезпечення цих вимог дозволяє користувачам працювати із системою на більшості сучасних персональних комп’ютерів і ноутбуків без необхідності додаткового встановлення спеціалізованого програмного забезпечення.

3.1.3 Програмний інтерфейс

Програмна система реалізована із використанням сучасного та гнучкого технологічного стеку, що забезпечує високу продуктивність, масштабованість і легку інтеграцію з іншими інструментами. Серверна частина побудована на базі C#, .NET 8 та ASP.NET Core Web API, що дозволяє створювати модульні, стабільні та надійні серверні сервіси з підтримкою REST API для інтеграції з клієнтською частиною й зовнішніми системами. Для зберігання даних застосовується хмарна NoSQL база даних CosmosDB, що підтримує роботу з напівструктурованими та векторними даними, необхідними для семантичного пошуку фрагментів коду.

Клієнтська частина системи реалізована як односторінковий веб-додаток на основі React.js, із використанням бібліотек React Query для асинхронної взаємодії з API, React Hook Form для роботи з формами, Styled Components для організації стилів і Monaco Editor для роботи з кодом прямо в браузері. Інтерфейс адаптований для роботи на різних пристроях та підтримує сучасний підхід до UI/UX.

Для автоматизованого аналізу якості коду й семантичного пошуку застосовується інтеграція з OpenAI API (GPT та Embedding-моделі), що дозволяє реалізувати інтелектуальні функції перевірки, рекомендацій і підбору фрагментів. Повідомлення й підтвердження для користувачів здійснюються через SMTP, з інтеграцією поштових серверів для верифікації акаунтів і сповіщень.

Впроваджено CI/CD на базі Azure DevOps: кожна зміна проходить автоматичне тестування (Unit та Integration tests за допомогою nUnit, Moq, Vitest, Testing Library), перевірку якості й розгортання на різних середовищах. Для зберігання та контролю версій використовується система Git із центральним репозиторієм на GitHub, що дозволяє організувати колективну розробку, проводити код-рев’ю, підтримувати історію змін і інтеграцію з іншими інструментами.

Архітектура системи логічно поділена на незалежні компоненти, що забезпечує гнучкість розширення й модифікації. Документовані REST API-ендпоінти дозволяють легко інтегрувати систему з іншими сервісами, зовнішніми додатками чи внутрішніми корпоративними рішеннями.

Такий комплексний підхід до побудови програмного інтерфейсу забезпечує надійність, сумісність та актуальність системи для сучасної командної розробки.

3.1.4 Комунікаційний протокол

Комунікація між клієнтською та серверною частинами програмної системи здійснюється через сучасний протокол передачі даних HTTPS (HyperText Transfer Protocol Secure). Всі запити від користувача, включаючи авторизацію, додавання, пошук, редагування та видалення фрагментів коду, відправляються з клієнтського веб-додатку до серверної частини через RESTful API. Використання HTTPS забезпечує захищену передачу інформації, шифрування даних та захист від несанкціонованого доступу.

Для інтеграції із зовнішніми сервісами (OpenAI API, SMTP, Azure DevOps) також використовуються стандартизовані протоколи передачі даних, такі як HTTPS та SMTP, із застосуванням авторизаційних токенів та ключів доступу для забезпечення безпеки обміну інформацією. Запити до зовнішніх API виконуються у захищених сесіях з перевіркою валідності відповідей та обробкою помилок.

Таким чином, система гарантує цілісність, конфіденційність та достовірність переданих даних як у межах локальної інфраструктури, так і під час інтеграції з зовнішніми хмарними або сервісними платформами.

3.1.5 Обмеження пам’яті

Для забезпечення стабільної роботи веб-інтерфейсу та виконання основних операцій із фрагментами програмного коду мінімально рекомендований обсяг оперативної пам’яті на користувацькому пристрої становить 4 ГБ. Це дозволяє підтримувати комфортну роботу з кількома відкритими вкладками браузера, виконувати основні дії у системі, а також взаємодіяти з редактором коду та іншими елементами інтерфейсу.

Для коректної роботи клієнтської частини у браузері та зберігання кешованих даних, історії пошуку, тимчасових файлів і локальних копій фрагментів коду рекомендується мати не менше 500 МБ вільного простору на диску користувача. Додатково, для завантаження архівів згенерованих проектів або експортованих даних потрібно мати відповідний вільний простір для розміщення цих файлів.

На серверній стороні обмеження по пам’яті визначаються інфраструктурою хмарного середовища (Azure), однак система оптимізована для роботи з великими обсягами кодових фрагментів та метаданих, а також для паралельного обслуговування багатьох користувачів. Використання сучасних механізмів кешування та асинхронної обробки запитів дозволяє мінімізувати споживання пам’яті та забезпечувати високу продуктивність навіть при зростанні навантаження.

3.1.6 Операції

Комунікація між клієнтською та серверною частинами системи здійснюється через HTTP(S)-запити у форматі REST API. Основними операціями, які підтримує система, є:

* Надсилання запитів на отримання списку фрагментів коду, їх перегляд, фільтрацію та пошук за ключовими словами або тегами.
* Додавання нового фрагмента коду із зазначенням супровідної інформації (назва, опис, мова програмування, теги).
* Редагування та оновлення існуючих фрагментів коду й пов’язаних з ними даних.
* Видалення фрагментів із особистої або командної колекції.
* Надсилання запитів на аналіз якості коду, перевірку на дублювання та відповідність стандартам програмування.
* Формування та експорт проекту на основі вибраних фрагментів у вигляді архіву для подальшої роботи.
* Управління користувачами, включаючи реєстрацію, аутентифікацію, зміну пароля та відновлення доступу.
* Керування доступом до командних або корпоративних колекцій, призначення ролей і запрошення нових учасників.
* Отримання та відправлення сповіщень щодо змін у спільних колекціях або виконання важливих операцій.

Всі дані передаються у форматі JSON, що забезпечує сумісність і простоту обробки з боку клієнта та сервера. Для захисту даних усі операції виконуються в захищеній сесії з використанням HTTPS, а доступ до приватної інформації обмежується механізмами авторизації та автентифікації.

3.2 Властивості програмного продукту

Система створена на основі сучасних технологій та актуальних методологій розробки для забезпечення стабільної та якісної роботи. Основна обробка даних, включаючи зберігання, аналіз якості коду, пошук і формування проектів, виконується на серверній частині системи, що дозволяє досягти високої продуктивності, оптимізувати використання ресурсів і забезпечити масштабованість рішення.

Такий підхід зменшує навантаження на клієнтські пристрої, оскільки основні обчислення та зберігання інформації відбуваються на сервері у хмарній інфраструктурі. Це дозволяє користувачам працювати із системою навіть на пристроях із невеликими апаратними ресурсами, забезпечуючи при цьому швидкий відгук та високу доступність сервісу.

Система підтримує безпечний доступ через захищений протокол, має механізми аутентифікації та контролю ролей, що гарантує конфіденційність і цілісність даних користувачів. Завдяки модульній архітектурі та використанню сучасних бібліотек та фреймворків, система легко розширюється і супроводжується, що дозволяє швидко впроваджувати новий функціонал відповідно до вимог ринку й користувачів.

3.3 Атрибути програмного продукту

3.3.1 Надійність

Система розроблена для забезпечення стабільної роботи при одночасному доступі значної кількості користувачів. Архітектура сервісу оптимізована для роботи у хмарному середовищі, що дозволяє обробляти численні паралельні запити до сховища фрагментів коду та до інструментів перевірки якості без помітного зниження продуктивності. Всі ключові компоненти автоматично масштабуються в залежності від навантаження, що забезпечує безперервність сервісу навіть у пікові періоди. У разі оновлення або розгортання нових версій можливі короткочасні затримки, проте вся інформація користувачів зберігається у хмарній базі даних, а автоматизовані бекапи дозволяють швидко відновити працездатність у разі аварійних ситуацій.

3.3.2 Доступність

Система орієнтована на цілодобову доступність для користувачів незалежно від їхнього розташування. Завдяки розміщенню в сучасній хмарній інфраструктурі (наприклад, Microsoft Azure), доступ до функціоналу забезпечується 24/7 з будь-якої точки світу, де є підключення до Інтернету. Використання відмовостійких серверів і балансувальників навантаження дозволяє уникнути простоїв навіть при виникненні технічних несправностей на окремих вузлах.

3.3.3 Безпека

Всі дані, що передаються між клієнтською та серверною частинами, шифруються за допомогою протоколу HTTPS, що гарантує захист від перехоплення і несанкціонованого доступу. Для автентифікації та авторизації користувачів застосовуються сучасні механізми з використанням токенів, а також двофакторна аутентифікація для корпоративних акаунтів. Контроль ролей і доступу до фрагментів коду дозволяє розмежовувати права між звичайними користувачами, членами команди та адміністраторами. Регулярно здійснюється аудит дій та моніторинг підозрілої активності.

3.3.4 Супроводжуваність

Програмна система побудована за принципами модульної архітектури й відповідає сучасним стандартам розробки, що спрощує внесення змін, додавання нового функціоналу та усунення помилок. Код системи детально документований, використовується система контролю версій Git, а також автоматичне тестування (unit, integration tests), що забезпечує стабільність і високу якість під час оновлень.

3.3.5 Переносимість

Клієнтська частина системи є кросплатформенною — для роботи з нею достатньо будь-якого сучасного браузера. Серверна частина розгортається у хмарі та не залежить від конкретної операційної системи користувача. Це дозволяє використовувати систему на пристроях із різними ОС (Windows, Linux, MacOS) без необхідності встановлення додаткового програмного забезпечення.

3.3.6 Продуктивність

Система оптимізована для обробки великої кількості запитів і роботи з великим обсягом даних без значних затримок. Використання асинхронних обробників запитів, кешування, а також масштабування серверних ресурсів дозволяє забезпечувати швидкий відгук інтерфейсу й мінімізувати час виконання основних операцій навіть при зростанні кількості користувачів.

3.4 Вимоги бази даних

Для зберігання основних даних, таких як фрагменти програмного коду, метадані, інформація про користувачів, команди та історію змін, система використовує сучасну розподілену NoSQL базу даних CosmosDB, розгорнуту у хмарній інфраструктурі. Вибір CosmosDB зумовлений потребою в масштабованості, високій доступності, швидкому виконанні операцій та підтримці напівструктурованих і векторних даних для семантичного пошуку.

База даних повинна забезпечувати надійне зберігання як текстових фрагментів коду, так і пов’язаних з ними структур — тегів, ієрархії папок, записів про зміни, а також прав доступу та ролей користувачів. Необхідно підтримувати можливість зберігання великої кількості документів, швидке виконання операцій пошуку, фільтрації й сортування, а також гнучке масштабування у разі зростання навантаження.

Для забезпечення захисту інформації й безперервної роботи система повинна автоматично створювати резервні копії даних, а також підтримувати механізми відновлення у разі аварійних ситуацій. Обмін даними між серверною частиною і базою відбувається через захищені з’єднання, що гарантує цілісність та конфіденційність інформації.

Всі вимоги до бази даних орієнтовані на забезпечення швидкого доступу до даних, стабільної роботи сервісу та можливості подальшого розширення функціоналу системи без необхідності суттєвих змін у структурі зберігання.