Форма № 24

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»

Факультет комп'ютерних наук і технологій
(повне найменування факультету)

Кафедра програмних засобів
(повне найменування кафедри)

Пояснювальна записка

до дипломного проєкту (роботи)
бакалавр
(ступінь вищої освіти)

на тему ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ ДЛЯ УПРАВЛІННЯ БІБЛІОТЕКОЮ ПРОГРАМНИХ ФРАГМЕНТІВ
SOFTWARE FOR MANAGING A LIBRARY OF CODE SNIPPETS

Виконав(ла): студент(ка) 4 курсу, групи КНТ-112
Спеціальності 121 Інженерія програмного
(код і найменування спеціальності)
забезпечення
Освітня програма (спеціалізація)
Інженерія програмного забезпечення

МІКУЛІН М.Д.
(ПРІЗВИЩЕ та ініціали)
Керівник КОЦУР М.І.
(ПРІЗВИЩЕ та ініціали)
Рецензент СКРУПСЬКИЙ С.Ю.
(ПРІЗВИЩЕ та ініціали)

2026

Форма № 25

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»

Факультет КНТ
Кафедра програмних засобів
Ступінь вищої освіти бакалавр
Спеціальність 121 Інженерія програмного забезпечення
(код і найменування)

Освітня програма (спеціалізація) Інженерія програмного забезпечення
(назва освітньої програми (спеціалізації)

ЗАТВЕРДЖУЮ

Завідувач кафедри ПЗ, д.т.н, проф.
_____Сергій СУББОТІН
“____” _____2026 року

З А В Д А Н Н Я

НА ДИПЛОМНИЙ ПРОЄКТ (РОБОТУ) СТУДЕНТА(КИ)

МІКУЛІНА Михайла Денисовича
(ПРІЗВИЩЕ, ім’я, по батькові)

1. Тема проєкту (роботи) Програмне забезпечення для управління бібліотекою програмних фрагментів. Software for Managing a Library of Code Snippets

керівник проєкту (роботи) к.т.н., доцент, КОЦУР Михайло Ігорович ,
(науковий ступінь, вчене звання, ПРІЗВИЩЕ, ім’я, по батькові)
затверджені наказом закладу вищої освіти від “ 7 ” квітня 2026 року № 139
2. Строк подання студентом проєкту (роботи) 03 червня 2026 року
3. Вихідні дані до проєкту (роботи) рекомендована література, технічне завдання

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити) 1. Аналіз предметної області. 2. Матеріали і методи. 3. Опис програми. 4. Експлуатація, тестування та експериментальне дослідження програми.

5. Перелік графічного матеріалу (з точним зазначенням обов’язкових креслень, кількість слайдів, плакатів)
Слайди презентації

6. Консультанти розділів проєкту (роботи)

Розділ	ПРІЗВИЩЕ, ініціали та посада консультанта	Підпис, дата	
		завдання видав	прийняв виконане завдання
1-4 Основна частина	КОЦУР М.І., доцент		
Нормоконтроль	ДЕЙНЕГА Л.Ю., ст. викладач		

7. Дата видачі завдання “ 7 ” квітня 2026 року.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломного проєкту (роботи)	Строк виконання етапів проєкту (роботи)	Примітка
1	Постановка завдання роботи.	1 тиждень	Завдання, ТЗ
2	Аналіз предметної області.	1 тиждень	Розділ 1
3	Вибір мови програмування та інших технологій	2 тиждень	Розділ 2
	розробки.		
4	Розробка структури програми.	2 тиждень	Розділ 3
5	Розробка програми.	3-4 тижні	Розділи 3, 4
6	Тестування та експериментальне дослідження	5 тиждень	Розділ 4
	програмного забезпечення.		
7	Оформлення пояснювальної записки та документів	6 тиждень	Додатки
	до неї.		
8	Нормоконтроль та рецензування.	7 тиждень	
9	Захист роботи.	8 тиждень	

Студент(ка) _____ Михайло МІКУЛІН
(підпис) (Імя ПРІЗВИЩЕ)

Керівник проєкту (роботи) _____ Михайло КОЦУР
(підпис) (Імя ПРІЗВИЩЕ)

РЕФЕРАТ

Пояснювальна записка до дипломної кваліфікаційної роботи бакалавра: 89 с., 3 табл., 31 рис., 3 дод., 16 джерел.

C#, VISUAL STUDIO, МОВА ПРОГРАМУВАННЯ, ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ, ПРОГРАМНИЙ КОД.

Об’єкт дослідження – процес розробки програмного забезпечення для управління бібліотекою програмних фрагментів.

Предмет дослідження – програмні засоби для управління бібліотекою програмних фрагментів.

Мета роботи – розробка програмного забезпечення для управління бібліотекою програмних фрагментів, що забезпечує зменшення часу пошуку та повторного написання коду.

Матеріали, методи та технічні засоби: мова програмування C#, середовище розробки Visual Studio.

Результати. Розроблено проєктні рішення для створення програмного забезпечення для управління бібліотекою програмних фрагментів. Створено програмне забезпечення для управління бібліотекою програмних фрагментів. Проведено тестування розробленої програмної системи для управління бібліотекою програмних фрагментів.

Висновки. Мету роботи досягнуто. Розроблено програмне забезпечення для управління бібліотекою програмних фрагментів за допомогою мови програмування C# та середовища розробки Visual Studio.

Галузь використання – розробка програмного забезпечення.

ABSTRACT

Explanatory note to the diploma qualifying work of the bachelor: 89 pages, 3 tables, 31 figures, 3 appendixes, 16 sources.

C#, VISUAL STUDIO, PROGRAMMING LANGUAGE, SOFTWARE, SOFTWARE CODE.

The object of research is the process of developing software for managing a library of program fragments.

The subject of the research is software for managing a library of program fragments.

The purpose of this work is to develop the software for managing a library of program fragments, which provides a reduction in the time of searching and rewriting the code.

Materials, methods and technical tools: C# programming language, Visual Studio development environment.

Results. Project solutions for the creation of software for managing a library of program fragments has been designed. The software for managing a library of program fragments has been developed. The developed software system for managing a library of program fragments has been tested.

Conclusions. The goal of the work has been achieved. The software tools for managing a library of program fragments using the C# programming language and the Visual Studio development environment have been developed.

Scope of use – software development.

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ ТА УМОВНИХ ПОЗНАК

API – Application Programming Interface;

MVC – Model-View-Controller;

БД – база даних;

ПЗ – програмне забезпечення.

ВСТУП

Актуальність програмного забезпечення для управління бібліотекою програмних фрагментів визначається сучасними потребами розробників у швидкому та ефективному доступі до готових компонентів коду. Використання такого програмного забезпечення (ПЗ) дозволяє оптимізувати процеси розробки, скорочуючи час на написання повторюваних або стандартних елементів програм, що особливо важливо у великих проектах із складною архітектурою. Крім того, системи керування бібліотекою програмного коду сприяють підвищенню якості програмних продуктів шляхом стандартизації та централізації зберігання фрагментів, забезпечуючи контроль версій, документацію та можливість повторного використання перевірених рішень [1], [2].

Актуальність таких програмних систем також зростає через необхідність співпраці між кількома розробниками, коли централізоване управління фрагментами зменшує ризик конфліктів, дублювання коду та помилок. У контексті сучасних технологій, де застосовуються мікросервіси, модульна архітектура та швидкі цикли релізів, програмне забезпечення для управління бібліотекою фрагментів стає невід’ємним інструментом, який підвищує продуктивність, забезпечує ефективне використання ресурсів і підтримує стандартизацію коду, що сприяє стабільності та масштабованості програмних рішень [3], [4].

Зважаючи на постійне зростання обсягів програмних проєктів і складність їх структури, роль таких систем продовжує збільшуватися, стаючи ключовим елементом сучасної інженерії програмного забезпечення [4]-[6].

Проте деякі програми для управління бібліотекою програмних фрагментів можуть бути досить складними для інтеграції у вже існуючі проєкти, оскільки часто виникає потреба у налаштуванні специфічних робочих процесів, сумісності з різними середовищами розробки та мовами програмування. Крім того, використання деяких програмних засобів для

централізованого зберігання коду створює ризики безпеки та доступності, оскільки помилки або збої у програмній системі можуть впливати на роботу всього колективу розробників. Важливою проблемою виступає й навчання користувачів, оскільки ефективне використання бібліотеки потребує певного рівня знань і дисципліни у підтриманні структури та стандартів. У підсумку, незважаючи на очевидні переваги, програмне забезпечення для управління бібліотекою програмних фрагментів залишається не позбавленим обмежень, що вимагає ретельного планування та регулярного контролю його експлуатації [1]-[6].

Тому актуальною є розробка програмного забезпечення для управління бібліотекою програмних фрагментів. У дипломній кваліфікаційній роботі бакалавра розв’язується актуальне завдання розробки програмного забезпечення для управління бібліотекою програмних фрагментів, що забезпечує зменшення часу пошуку та повторного написання коду.

Для досягнення поставленої мети у кваліфікаційній роботі бакалавра необхідно розв’язати такі задачі:

– виконати аналіз предметної області та програмних засобів для управління бібліотекою програмних фрагментів;

– здійснити проєктування програмного забезпечення для управління бібліотекою програмних фрагментів;

– створити програмне забезпечення для управління бібліотекою програмних фрагментів;

– виконати тестування розробленого програмного забезпечення для управління бібліотекою програмних фрагментів.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Програмне забезпечення для управління бібліотекою програмних фрагментів

Програмне забезпечення для управління бібліотекою програмних фрагментів є класом спеціалізованих інформаційних систем, призначених для централізованого зберігання, впорядкування та повторного використання типових елементів програмного коду. Такі програмні системи забезпечують формування єдиного структурованого сховища фрагментів, що можуть застосовуватися під час розробки програмного забезпечення незалежно від конкретного проєкту або середовища виконання. Їх використання орієнтоване на підтримку систематичної роботи з програмним кодом, зменшення дублювання рішень і підвищення узгодженості програмних компонентів [1].

У межах таких програмних засобів створюються умови для логічної класифікації фрагментів за мовами програмування, призначенням, тематикою або іншими ознаками, що полегшує навігацію та пошук необхідної інформації. Передбачається можливість збереження як простих прикладів синтаксису, так і більш складних шаблонів, які можуть бути адаптовані до різних контекстів використання. Це сприяє скороченню часу на розробку та зменшенню ймовірності помилок, пов’язаних із повторним написанням стандартних конструкцій [1].

Особливістю програмного забезпечення для управління бібліотекою програмних фрагментів є орієнтація на підтримку знань і накопичення практичного досвіду розробки. Такі системи виступають своєрідною базою знань, у якій фрагменти коду зберігаються разом із поясненнями, коментарями або контекстом застосування. Це дозволяє забезпечити спадковість рішень, полегшити навчання нових учасників команди та підтримувати єдині підходи до програмування [1].

Переваги використання зазначених програмних засобів проявляються у підвищенні продуктивності роботи розробників, оптимізації процесів

створення програмного забезпечення та покращенні якості кінцевого продукту. Завдяки централізованому доступу до перевірених фрагментів зменшується залежність від індивідуальної пам’яті розробника, а також спрощується стандартизація коду в межах організації або проєкту. Крім того, такі системи можуть підтримувати спільну роботу, що сприяє обміну напрацюваннями та формуванню єдиного інформаційного простору для команди [1].

Загалом програмне забезпечення для управління бібліотекою програмних фрагментів розглядається як допоміжний інструмент інженерії програмного забезпечення, що забезпечує раціональне використання наявних рішень, сприяє підвищенню ефективності розробки та підтримує впровадження кращих практик програмування в повсякденну діяльність [1].

Додатково програмне забезпечення для управління бібліотекою програмних фрагментів може розглядатися як інструмент, що відіграє важливу роль у процесах стандартизації та уніфікації програмного коду. За його допомогою забезпечується формування єдиних підходів до реалізації типових задач, що позитивно впливає на підтримуваність програмних продуктів і спрощує подальший супровід систем. Використання узгоджених фрагментів коду сприяє зменшенню розбіжностей у стилі програмування та підвищенню читабельності вихідних текстів [2].

Таке програмне забезпечення також може виконувати функцію засобу контролю якості, оскільки до бібліотеки, як правило, потрапляють перевірені та апробовані фрагменти, що вже були використані на практиці. Це дозволяє знизити ризик появи помилок і вразливостей у нових проєктах, а також забезпечити повторне застосування надійних рішень. У цьому контексті бібліотека програмних фрагментів може розглядатися як накопичувальний ресурс технічного досвіду, який з часом набуває особливої цінності [2].

Важливим аспектом є інтеграція таких програмних систем із середовищами розробки та іншими інструментами життєвого циклу програмного забезпечення. Завдяки цьому створюються умови для швидкого

доступу до фрагментів безпосередньо під час написання коду, що мінімізує переривання робочого процесу. Крім того, можливість синхронізації з системами керування версіями або засобами командної роботи розширює функціональні можливості та підвищує зручність використання [2].

Програмне забезпечення для управління бібліотекою програмних фрагментів може застосовуватися не лише у професійній розробці, а й у навчальному процесі. Воно сприяє накопиченню прикладів реалізації різних алгоритмів і конструкцій, що полегшує засвоєння матеріалу та формування практичних навичок програмування. У такому випадку бібліотека фрагментів виступає як навчальний ресурс, орієнтований на багаторазове використання та поступове розширення [3].

Таким чином, програмне забезпечення для управління бібліотекою програмних фрагментів може розглядатися як багатофункціональний інструмент, що поєднує в собі засоби зберігання, організації та поширення програмних рішень. Їх використання сприяє підвищенню ефективності розробки, забезпечує збереження знань і підтримує безперервність процесів створення та супроводу програмного забезпечення [3].

Процес розробки програмного забезпечення для управління бібліотекою програмних фрагментів зазвичай розпочинається з формування узагальненого бачення майбутньої системи та визначення її призначення у межах діяльності розробників або команд. На цьому етапі уточнюються основні сценарії використання, характер взаємодії користувачів із системою та очікувані результати її впровадження. Особлива увага приділяється тому, яким чином програмні фрагменти будуть накопичуватися, зберігатися та повторно використовуватися у різних проєктах [3], [4].

Подальший етап пов’язується з аналізом функціональних і нефункціональних вимог, що дає змогу визначити необхідний обсяг можливостей системи, рівень продуктивності, вимоги до безпеки та масштабованості. У межах цього процесу формується уявлення про архітектуру програмного забезпечення, обираються принципи організації

даних і визначаються технологічні рішення, здатні забезпечити стабільну роботу системи в умовах зростання обсягу бібліотеки фрагментів [3], [4].

Після цього здійснюється проєктування внутрішньої структури програмного забезпечення, під час якого моделюються основні компоненти, їх взаємодія та логіка обробки інформації. На цьому етапі закладаються механізми керування доступом, пошуку та класифікації програмних фрагментів, а також визначаються підходи до інтеграції з іншими інструментами розробки. Проєктні рішення мають забезпечувати гнучкість системи та можливість її подальшого розширення без суттєвих змін базової логіки [3], [4].

Безпосередня реалізація програмного забезпечення для управління бібліотекою програмних фрагментів полягає у поетапному створенні функціональних модулів відповідно до затвердженої архітектури. У ході розробки особлива увага приділяється якості коду, узгодженості стилю програмування та повторному використанню внутрішніх компонентів, що відповідає самій ідеї бібліотеки фрагментів. Паралельно здійснюється перевірка працездатності окремих частин системи та їх коректної взаємодії між собою [3], [4].

Наступним важливим етапом є тестування, у межах якого оцінюється відповідність програмного забезпечення для управління бібліотекою програмних фрагментів визначеним вимогам і стабільність його функціонування в різних умовах використання. Перевіряється правильність збереження та обробки даних, ефективність механізмів пошуку, а також надійність системи керування доступом. Результати тестування використовуються для усунення виявлених недоліків і оптимізації роботи системи [3], [4].

Завершальним етапом процесу розробки стає підготовка програмного забезпечення для управління бібліотекою програмних фрагментів до впровадження та подальшого супроводу. На цьому етапі забезпечується готовність системи до реальної експлуатації, формуються рекомендації щодо її

використання та визначаються підходи до оновлення і розвитку. У результаті формується програмний продукт, здатний ефективно підтримувати процес накопичення та повторного використання програмних фрагментів, а також адаптуватися до змін потреб користувачів і середовища розробки [3], [4].

Фрагменти програмного коду розглядаються як ефективний засіб оптимізації процесу розробки, що дозволяє скоротити кількість повторюваних дій і забезпечити єдність реалізації типових рішень. Замість постійного створення однакових елементів з початкового рівня, з’являється можливість скористатися вже підготовленими заготовками, які відповідають потрібному контексту та значно зменшують імовірність виникнення помилок під час ручного введення. Такий підхід спрощує подальший супровід програмного забезпечення та полегшує виявлення проблем у процесі налагодження [5].

Під фрагментами коду розуміється сукупність програмних конструкцій, призначених для багаторазового застосування. Такі блоки зберігаються з метою подальшого обліку та швидкого доступу, що робить їх зручними для повторного використання у майбутньому. За своєю природою вони нагадують добре відомі та легко відтворювані елементи тексту, які можна без зусиль інтегрувати у новий контекст. Збережені фрагменти дають змогу підтримувати єдиний стиль програмування в межах команди та використовувати перевірені рішення без необхідності повторного введення коду, що вже довів свою працездатність [5].

Зазвичай фрагменти коду зберігаються у текстових редакторах, інтегрованих середовищах розробки або спеціалізованих системах керування такими елементами. Це забезпечує швидке вставлення потрібних конструкцій у проєкти за допомогою мінімальної кількості дій. Наприклад, введення короткого умовного позначення може автоматично перетворюватися на повноцінну структуру коду, необхідну для створення окремого компонента чи початкової частини програми [5], [6].

Практика використання фрагментів коду є корисною не лише для досвідчених фахівців, а й для всіх, хто має справу з програмуванням.

Збереження та повторне застосування типових рішень сприяє економії часу та підвищенню продуктивності, особливо в умовах обмежених термінів виконання завдань. У сучасному середовищі розробки володіння навичками створення таких заготовок поступово переходить із додаткової можливості у необхідну складову ефективної роботи [5], [6].

Розуміння процесу формування фрагментів програмного коду дозволяє перетворювати об’ємні та часто повторювані ділянки програмного тексту на компактні скорочення, доступ до яких здійснюється максимально швидко. Для цього доцільно аналізувати власний стиль програмування та звертати увагу на ті елементи, що регулярно використовуються у повсякденній роботі. Повторювані частини, незалежно від того, стосуються вони інтерфейсу, серверної логіки чи перевірки даних, можуть значно уповільнювати розробку за умови постійного переписування. Їх виділення та збереження у вигляді фрагментів створює більш впорядкований і продуктивний робочий процес, що позитивно впливає на загальну ефективність команди [5], [6].

Перед тим як перетворювати певний фрагмент програмного коду на багаторазово використовуваний елемент, доцільним вважається його попереднє доопрацювання. Код має бути очищений від зайвих конструкцій, дублювань і випадкових рішень, а також приведений до єдиного стилю оформлення. Важливо переконатися, що він коректно працює та не містить помилок, оскільки фрагменти призначені для прямого застосування в різних проєктах без необхідності постійного доопрацювання. Якісний фрагмент повинен бути готовим до використання одразу після вставлення, не вимагаючи суттєвих змін [5], [6].

Для підвищення універсальності фрагментів програмного коду широко застосовується механізм змінних елементів або спеціальних маркерів, які дозволяють адаптувати код до конкретного випадку. Такі маркери розміщуються в тих частинах програмного тексту, що найчастіше змінюються, наприклад у назвах методів, класів або адресах ресурсів. Після вставлення фрагмента курсор автоматично переміщується між цими позиціями, що значно

прискорює введення індивідуальних значень і позбавляє потреби редагувати код вручну [5], [6].

Збереження фрагментів коду тісно пов’язане з вибором відповідного формату та інструментів. Різні середовища розробки пропонують власні підходи до організації та зберігання таких заготовок, тому рекомендовано дотримуватися усталених стандартів, які забезпечують зручну структуру та сумісність. Використання офіційно підтримуваних форматів спрощує керування фрагментами та робить їх доступними у межах конкретного середовища або на різних пристроях [5], [6].

Важливу роль відіграє також коректне іменування фрагментів і визначення ключового слова для їх швидкого виклику. Чітка та змістовна назва полегшує орієнтацію в сховищі фрагментів, а короткий і продуманий тригер дозволяє миттєво розгорнути потрібний код. Уникнення конфліктів із загальновживаними словами зменшує ризик помилкового спрацювання та робить роботу з фрагментами більш передбачуваною [5], [6].

Після збереження фрагмента необхідним етапом вважається його перевірка у тестовому середовищі. Введення тригера в порожньому редакторі дозволяє впевнитися, що код розгортається правильно, а форматування не порушується. Саме на цьому етапі доцільно внести фінальні корективи, щоб фрагмент був повністю готовим до практичного застосування [5], [6].

Зі збільшенням кількості фрагментів виникає потреба в їх впорядкуванні. Без належної організації сховище швидко стає складним для навігації, що знижує ефективність використання. Класифікація, логічне групування та використання зрозумілих позначень допомагають швидко знаходити потрібні елементи. Додатково важливо подбати про резервне копіювання та узгоджений формат зберігання, щоб уникнути втрати напрацьованих рішень [5], [6].

Незважаючи на очевидні переваги, неправильний підхід до створення фрагментів може призводити до зворотного ефекту. Однією з поширених проблем є надмірне використання жорстко закріплених значень, які

ускладнюють повторне застосування коду. Такі елементи значно обмежують гнучкість і вимагають ручного редагування при кожному використанні. Іншою типовою помилкою є ігнорування змінних маркерів, що змушує постійно переписувати фрагменти замість швидкої адаптації [5], [6].

Додаткові труднощі виникають у випадках, коли фрагменти не супроводжуються поясненнями або зрозумілим контекстом. За відсутності чітких описів зростає ймовірність неправильного використання, особливо в командному середовищі. Також негативно впливає прагнення зробити один фрагмент надмірно універсальним і об’ємним, оскільки це ускладнює його сприйняття та знижує зручність повторного застосування. Не менш важливим аспектом є вибір назв, адже заплутані або надто довгі тригери складно запам’ятовуються та уповільнюють робочий процес. Продуманий і стриманий підхід до іменування робить роботу з фрагментами більш інтуїтивною та ефективною [5], [6].

Визначено, що програмне забезпечення для управління бібліотекою програмних фрагментів є класом спеціалізованих інформаційних систем, призначених для централізованого зберігання, впорядкування та повторного використання типових елементів програмного коду. Такі програмні системи забезпечують формування єдиного структурованого сховища фрагментів, що можуть застосовуватися під час розробки програмного забезпечення незалежно від конкретного проєкту або середовища виконання. Їх використання орієнтоване на підтримку систематичної роботи з програмним кодом, зменшення дублювання рішень і підвищення узгодженості програмних компонентів.

1.2 Аналіз програмного забезпечення для управління бібліотекою програмних фрагментів

Проаналізуємо програмне забезпечення для управління бібліотеками програмних фрагментів [7]-[12].

Програмне забезпечення Masscode (рис. 1.1) призначене для організованого збереження, впорядкування та повторного використання програмних фрагментів у процесі розробки програмних продуктів. Його функціонування орієнтоване на підвищення ефективності роботи розробників шляхом централізованого накопичення коду та супровідної інформації, що дозволяє швидко знаходити потрібні рішення та адаптувати їх до поточних задач. У межах системи реалізовано підхід, за якого окремі фрагменти програмного коду розглядаються як самостійні об’єкти з власними характеристиками, описами та контекстом використання, що забезпечує зручність їх подальшого застосування [7], [9], [10].

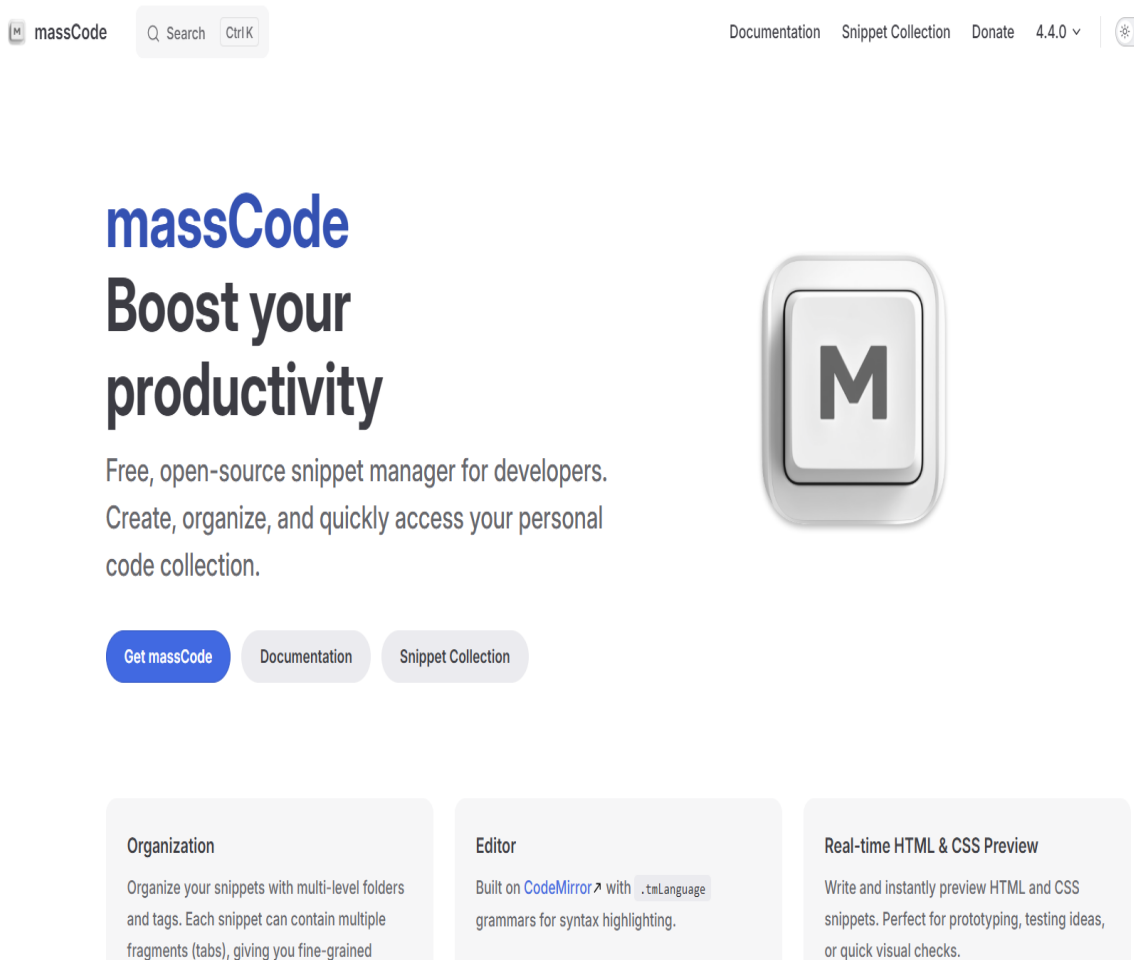

Рисунок 1.1 – Програмне забезпечення Masscode [10]

Програмне забезпечення Masscode забезпечує можливість зберігання програмних фрагментів різного призначення разом із текстовими

поясненнями, позначками та структурними атрибутами, які спрощують навігацію в бібліотеці. Взаємодія користувача з програмним забезпеченням будується таким чином, щоб мінімізувати часові витрати на пошук та аналіз уже створених рішень. Система підтримує гнучку організацію даних, що дозволяє групувати фрагменти за логічними ознаками, мовами програмування або сферою застосування, створюючи цілісне сховище знань, корисне як для індивідуальних розробників, так і для команд [7], [9], [10].

Важливою особливістю Masscode є орієнтація на зручність повсякденного використання у реальному процесі розробки. Програмне забезпечення надає інструменти для швидкого додавання нових фрагментів, редагування вже наявного коду та актуалізації супровідної інформації без порушення цілісності бібліотеки. При цьому зберігається логічна структура сховища, що сприяє підтриманню порядку навіть за умови значного обсягу даних. Такий підхід дозволяє уникати дублювання коду та сприяє формуванню єдиного стилю програмування [7], [9], [10].

Окрему увагу в Masscode приділено процесу пошуку та відбору фрагментів програмного коду. Реалізовані механізми дозволяють швидко знаходити необхідні елементи за змістом, описом або ключовими характеристиками, що особливо важливо при роботі з великою кількістю збережених рішень. Це перетворює бібліотеку програмних фрагментів на активний інструмент підтримки розробки, а не на пасивне сховище даних [7], [9], [10].

Загалом Masscode можна охарактеризувати як спеціалізоване програмне забезпечення, спрямоване на систематизацію та повторне використання програмного коду, що сприяє підвищенню продуктивності розробників і покращенню якості програмних продуктів. Його використання дозволяє ефективно накопичувати практичний досвід у вигляді коду та забезпечує зручний доступ до нього в процесі подальшої роботи [7], [9], [10].

Основними характеристиками програмного забезпечення Masscode є такі [7], [9], [10]:

– централізоване зберігання коду: програмне забезпечення Masscode підтримує накопичення фрагментів у єдиній бібліотеці з можливістю подальшого повторного використання та впорядкування [7], [9], [10];

– підтримка метаданих: кожен фрагмент програмного коду супроводжується описами, тегами та додатковими атрибутами, що спрощує його ідентифікацію та пошук у сховищі [7], [9], [10];

– зручний пошук і фільтрація: у програмному забезпеченні Masscode реалізовані механізми швидкого пошуку дозволяють знаходити фрагменти за змістом, ключовими словами або структурними ознаками [7], [9], [10];

– орієнтація на повторне використання: програмне забезпечення Masscode сприяє зменшенню дублювання коду та формуванню єдиної бази перевірених рішень для подальшого застосування [7], [9], [10];

– підтримка різних мов програмування: бібліотека може містити фрагменти, написані різними мовами, що робить програмне забезпечення універсальним інструментом для розробників [7], [9], [10];

– інтуїтивна організація бібліотеки: логічна структура зберігання фрагментів коду у програмному забезпеченні Masscode сприяє впорядкованому веденню коду навіть за значного обсягу даних [7], [9], [10];

– можливість редагування фрагментів: програмне забезпечення Masscode дозволяє оновлювати та уточнювати вміст бібліотеки без втрати її цілісності [7], [9], [10];

– підтримка командної роботи: Masscode може використовуватися як спільне сховище програмних рішень для групи розробників [7], [9], [10].

Серед недоліків програмного забезпечення Masscode можна відзначити обмежені можливості інтеграції з окремими середовищами розробки, а також потребу у додатковому налаштуванні для адаптації до специфічних робочих процесів окремих команд, що може ускладнювати його впровадження у складні або нестандартні проєкти [7], [9], [10].

Програмне забезпечення Snipit (рис. 1.2) орієнтоване на зручне керування бібліотекою програмних фрагментів безпосередньо у веббраузері та

позиціонується як легкий інструмент для щоденного використання розробниками. Його інтерфейс побудований у стилі сучасних середовищ програмування, що забезпечує швидку адаптацію користувачів і комфортну роботу з кодом. Архітектура рішення базується на сучасних вебтехнологіях, завдяки чому система демонструє високу швидкодію та стабільність під час редагування і збереження фрагментів програмного коду [7], [11].

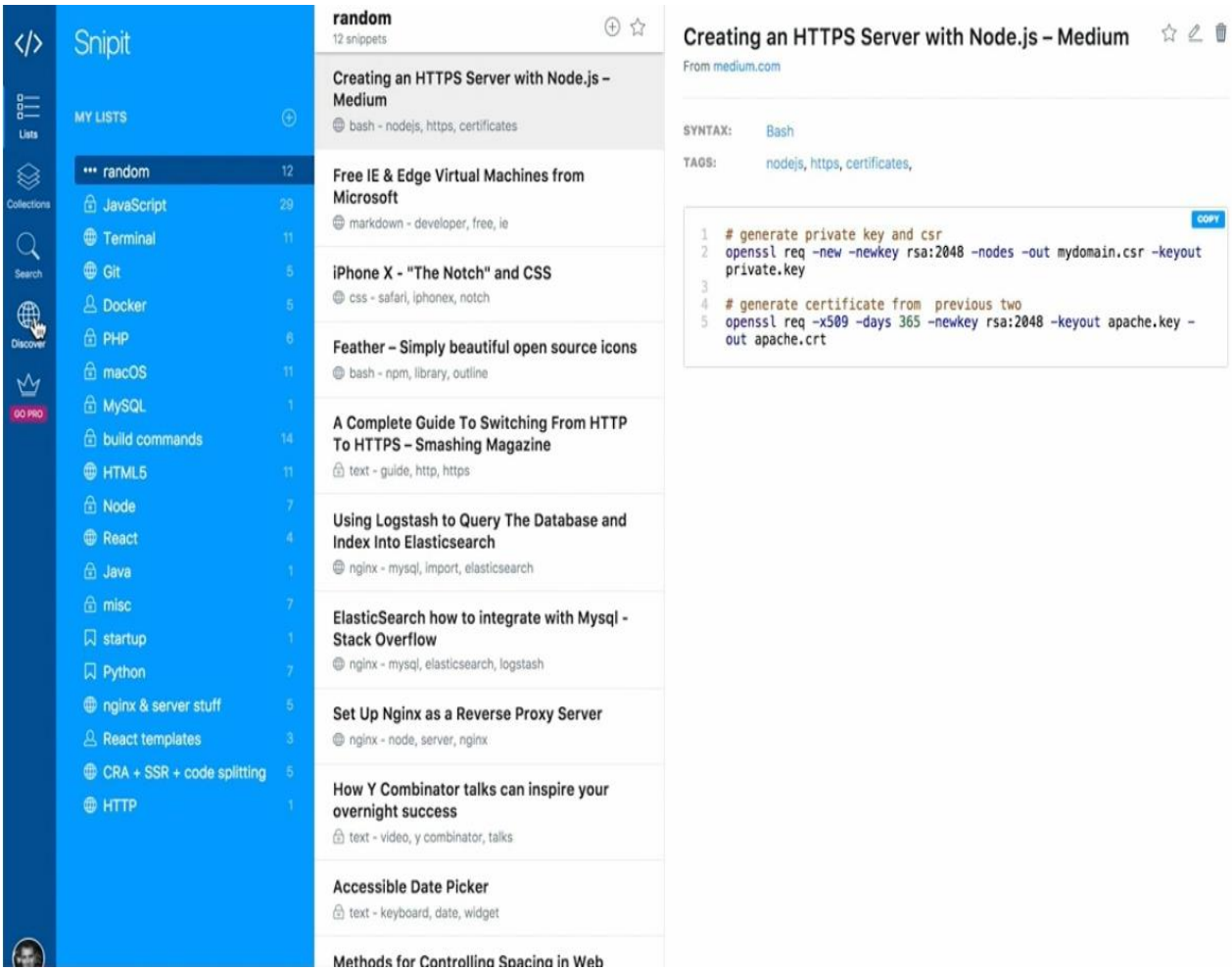

Рисунок 1.2 – Програмне забезпечення Snipit [11]

Функціонування Snipit зосереджене на можливості створення, редагування та організації фрагментів програмного коду без необхідності встановлення окремого настільного застосунку. Збереження даних здійснюється локально у середовищі браузера, що дозволяє працювати з бібліотекою автономно та забезпечує швидкий доступ до інформації.

Вбудований редактор коду підтримує підсвічування синтаксису для широкого кола мов програмування та за відчуттями наближений до популярних професійних редакторів, що підвищує зручність роботи та зменшує ймовірність помилок [7], [11].

Особливістю Snipit є гнучка організація бібліотеки програмних фрагментів, яка дозволяє структурувати код за колекціями, назвами та тематичними ознаками. Такий підхід спрощує навігацію у великому обсязі збережених даних і сприяє повторному використанню напрацювань у різних проєктах. Додатково реалізовано механізми швидкого створення нових фрагментів і використання комбінацій клавіш, що оптимізує робочий процес і скорочує час виконання типових дій [7], [11].

Snipit також надає можливості обміну програмними фрагментами шляхом експорту та імпорту, що робить можливим перенесення бібліотеки між середовищами або спільне використання коду з іншими розробниками. Окремі інструменти орієнтовані на роботу з кодом, знайденим на вебсторінках, що дозволяє швидко зберігати корисні приклади з зовнішніх джерел і включати їх до власної бібліотеки. Водночас зберігання даних у межах браузера передбачає відповідальність користувача за резервне копіювання, оскільки автоматична синхронізація з віддаленим сховищем може бути відсутньою [7], [11].

Програмне забезпечення Snipit має такі особливості [7], [11]:

– браузерна орієнтація: програмне забезпечення Snipit функціонує безпосередньо у веббраузері та не потребує інсталяції окремого настільного застосунку, що спрощує доступ до бібліотеки програмних фрагментів [7], [11];

– легковаговий інтерфейс: інтерфейс користувача побудований за принципами мінімалізму та нагадує сучасні редактори коду, що забезпечує швидке освоєння інструмента [7], [11];

– розвинений редактор коду: використання вбудованого редактора з підсвічуванням синтаксису дозволяє зручно працювати з фрагментами різними мовами програмування [7], [11];

– локальне зберігання даних: програмні фрагменти зберігаються у середовищі браузера, що забезпечує швидкий доступ до інформації без необхідності постійного підключення до мережі [7], [11];

– гнучка організація бібліотеки: програмне забезпечення Snipit підтримує структурування фрагментів за колекціями, назвами та тегами, що полегшує навігацію у сховищі коду [7], [11];

– розширені можливості пошуку: у програмному забезпеченні Snipit реалізовані механізми пошуку дозволяють швидко знаходити необхідні фрагменти за різними характеристиками [7], [11];

– підтримка швидких дій: у програмному забезпеченні Snipit підтримується наявність гарячих клавіш і швидкого створення нових фрагментів оптимізує робочий процес розробника [7], [11];

– можливість обміну даними: у програмному забезпеченні Snipit передбачені засоби експорту та імпорту програмних фрагментів для перенесення або спільного використання бібліотеки [7], [11].

Серед недоліків програмного забезпечення Snipit можна відзначити залежність від середовища браузера, обмежені можливості автоматичного резервного копіювання даних, а також недостатньо розкритий функціонал командної взаємодії, що може знижувати ефективність його використання у великих розподілених проєктах [7], [11].

Програмне забезпечення CodePen (рис. 1.3) являє собою веборієнтовану платформу, що широко використовується у середовищі розробки для створення, збереження та поширення програмних фрагментів. Його основне призначення полягає у наданні інтерактивного простору, де програмний код може редагуватися та виконуватися безпосередньо в браузері з миттєвим відображенням результатів. Такий підхід перетворює окремі приклади коду на наочні й зручні для повторного використання елементи, які можуть виконувати роль своєрідної бібліотеки програмних рішень [7], [12].

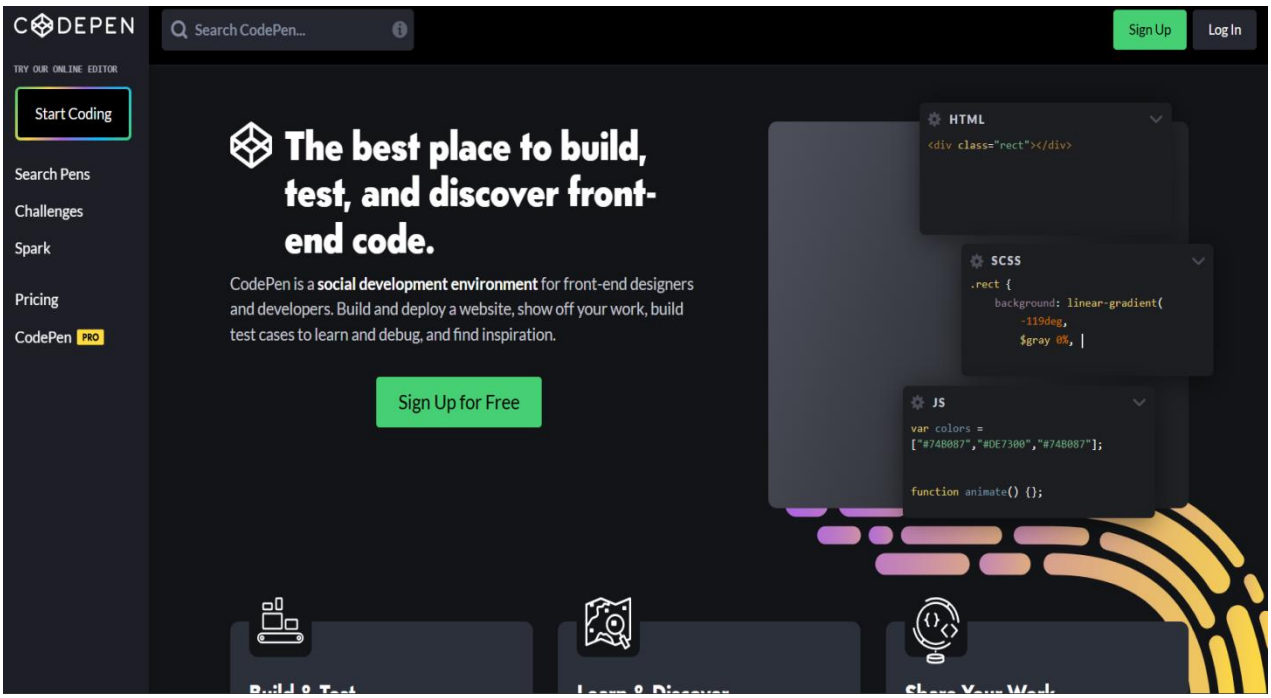

Рисунок 1.3 – Програмне забезпечення CodePen [12]

Функціонування програмного забезпечення CodePen ґрунтується на одночасній роботі з розміткою, стилями та сценаріями, що дозволяє комплексно демонструвати поведінку вебінтерфейсів. Користувачеві надається можливість експериментувати з кодом у реальному часі, одразу спостерігаючи зміни у візуальному представленні, що суттєво спрощує процес аналізу та вдосконалення фрагментів. Платформа також підтримує розширені інструменти для роботи з похідними синтаксисами та сучасними технологіями веброзробки, що робить її придатною для створення більш складних і гнучких рішень [7], [12].

Окремою рисою програмного забезпечення CodePen є орієнтація на обмін програмними напрацюваннями. Кожен створений фрагмент може бути представлений у вигляді самостійного прикладу, доступного для перегляду іншими користувачами або для інтеграції у зовнішні ресурси. Це сприяє активному поширенню знань і формуванню спільного середовища, у якому код виступає не лише інструментом розробки, а й засобом комунікації між фахівцями. Завдяки цьому бібліотека фрагментів постійно поповнюється новими ідеями та підходами [7], [12].

Програмне забезпечення CodePen також застосовується як середовище спільної роботи, у межах якого декілька учасників можуть взаємодіяти над одними й тими самими проєктами, узгоджуючи зміни та вдосконалюючи результати. Така модель використання особливо корисна для навчальних цілей, прототипування інтерфейсів і демонстрації концепцій. Водночас спрямованість платформи CodePen переважно на клієнтські вебтехнології зумовлює її обмежену придатність для задач, що виходять за межі фронтенд розробки [7], [12].

У цілому програмне забезпечення CodePen можна розглядати як універсальний онлайн інструмент для керування та презентації програмних фрагментів у сфері вебінтерфейсів, який поєднує функції редактора, сховища та соціальної платформи. Його використання сприяє швидкому обміну ідеями, навчанню на практичних прикладах і ефективному повторному застосуванню коду у процесі розробки [7], [12].

Програмне забезпечення CodePen має такі особливості [7], [12]:

– веборієнтоване середовище: програмне забезпечення CodePen функціонує повністю у браузері та не потребує встановлення локальних інструментів, що забезпечує швидкий доступ до бібліотеки програмних фрагментів [7], [12];

– інтерактивне редагування коду: програмне забезпечення CodePen підтримує одночасну роботу з розміткою, стилями та сценаріями з миттєвим відображенням результатів змін [7], [12];

– підтримка сучасних технологій: програмне забезпечення CodePen дозволяє використовувати розширені синтаксиси та інструменти, що застосовуються у сучасній веброзробці [7], [12];

– збереження та організація фрагментів: створені приклади коду можуть зберігатися як окремі елементи та використовуватися повторно у різних проєктах [7], [12];

– можливість вбудовування: програмні фрагменти легко інтегруються у зовнішні вебресурси для демонстрації або спільного доступу [7], [12];

– орієнтація на спільноту: програмне забезпечення CodePen об’єднує велику кількість розробників, що сприяє обміну досвідом, ідеями та готовими рішеннями [7], [12];

– підтримка колективної роботи: у програмному забезпеченні CodePen реалізовані інструменти для взаємодії кількох користувачів у межах спільних проєктів [7], [12];

– навчальний потенціал: програмне забезпечення CodePen широко використовується для демонстрації прикладів, експериментів та вивчення підходів до розробки інтерфейсів [7], [12].

Серед недоліків програмного забезпечення CodePen можна відзначити його обмежену придатність для роботи з серверними мовами програмування, залежність від постійного доступу до мережі, а також те, що функції збереження та керування великими приватними бібліотеками програмних фрагментів можуть бути менш зручними у порівнянні зі спеціалізованими менеджерами коду [7], [12].

Порівняльну характеристику програмного забезпечення для управління бібліотеками програмних фрагментів наведено у таблиці 1.1.

Таблиця 1.1 – Порівняння програмного забезпечення для управління бібліотеками програмних фрагментів

Критерій порівняння	Masscode [10]	Snipit [11]	CodePen [12]
підтримка різних мов програмування	+	+	+–
веборієнтований доступ	–	+	+
можливості структурування бібліотеки фрагментів коду	+	+	+–
підтримка командної роботи	+–	+–	+
інтеграція з процесом розробки	+	+–	+–
масштабованість бібліотеки	+	+–	+–

За результатами проведеного аналізу можна зробити висновок, що у наш час існує досить багато програмних засобів для управління бібліотеками програмних фрагментів. Проте деякі програмні засоби для управління бібліотеками програмних фрагментів можуть бути досить складними для інтеграції у вже існуючі проєкти, оскільки часто виникає потреба у налаштуванні специфічних робочих процесів, сумісності з різними середовищами розробки та мовами програмування. Крім того, використання деяких програмних засобів для централізованого зберігання коду створює ризики безпеки та доступності, оскільки помилки або збої у програмній системі можуть впливати на роботу всього колективу розробників. Важливою проблемою виступає й навчання користувачів, оскільки ефективне використання бібліотеки потребує певного рівня знань і дисципліни у підтриманні структури та стандартів. У підсумку, незважаючи на очевидні переваги, програмне забезпечення для управління бібліотекою програмних фрагментів залишається не позбавленим обмежень, що вимагає ретельного планування та регулярного контролю його експлуатації. Тому актуальною є розробка програмного забезпечення для управління бібліотекою програмних фрагментів.

При розробці програмного забезпечення для управління бібліотекою програмних фрагментів необхідно забезпечити такі функціональні вимоги:

– підтримка можливості збереження програмних фрагментів та пов’язаних з ними описів у базі даних системи;

– підтримка можливості редагування, оновлення та видалення програмних фрагментів і їхніх метаданих;

– можливість класифікації програмних фрагментів за мовами програмування, призначенням та іншими ознаками;

– можливість пошуку програмних фрагментів за ключовими словами, категоріями та змістом опису;

– можливість контролю доступу користувачів до операцій перегляду, редагування та адміністрування бібліотеки;

– забезпечення коректної обробки помилок і інформування користувача про результати виконання операцій у системі.

1.3 Висновки за розділом 1

Визначено, що програмне забезпечення для управління бібліотекою програмних фрагментів є класом спеціалізованих інформаційних систем, призначених для централізованого зберігання, впорядкування та повторного використання типових елементів програмного коду. Такі програмні системи забезпечують формування єдиного структурованого сховища фрагментів, що можуть застосовуватися під час розробки програмного забезпечення незалежно від конкретного проєкту або середовища виконання. Їх використання орієнтоване на підтримку систематичної роботи з програмним кодом, зменшення дублювання рішень і підвищення узгодженості програмних компонентів.

За результатами проведеного аналізу зроблено висновок, що у наш час існує досить багато програмних засобів для управління бібліотекою програмних фрагментів. Проте деякі програмні засоби для управління бібліотеками програмних фрагментів можуть бути досить складними для інтеграції у вже існуючі проєкти, оскільки часто виникає потреба у налаштуванні специфічних робочих процесів, сумісності з різними середовищами розробки та мовами програмування. Крім того, використання деяких програмних засобів для централізованого зберігання коду створює ризики безпеки та доступності, оскільки помилки або збої у програмній системі можуть впливати на роботу всього колективу розробників. Важливою проблемою виступає й навчання користувачів, оскільки ефективне використання бібліотеки потребує певного рівня знань і дисципліни у підтриманні структури та стандартів. У підсумку, незважаючи на очевидні переваги, програмне забезпечення для управління бібліотекою програмних фрагментів залишається не позбавленим обмежень, що вимагає ретельного

планування та регулярного контролю його експлуатації. Тому актуальною є розробка програмного забезпечення для управління бібліотекою програмних фрагментів.

Сформульовано функціональні вимоги до програмного забезпечення для управління бібліотекою програмних фрагментів.

2 МАТЕРІАЛИ І МЕТОДИ

2.1 Вибір мови програмування

При розробці програмного забезпечення для управління бібліотекою програмних фрагментів було використано мову програмування C# [13], [14].

Мова програмування C# є кросплатформенною мовою завдяки платформі .NET Core / .NET 5+, що дозволяє запускати програми на Windows, Linux і macOS без суттєвих змін у коді. Це робить мову гнучкою для розробки програмного забезпечення в різних середовищах. Мова C# активно підтримує сучасні парадигми програмування: об’єктно-орієнтоване, функціональне та компонентне програмування, що дозволяє створювати більш чистий, масштабований і підтримуваний код [13], [14].

Ще однією перевагою мова програмування C# є наявність потужної екосистеми програмних інструментів розробника. Visual Studio і Visual Studio Code забезпечують інтегроване середовище розробки з автодоповненням коду, відлагодженням, профілюванням та тестуванням, що значно пришвидшує розробку та підвищує якість програмних продуктів. C# підтримує роботу з базами даних через засоби Entity Framework, LINQ та інші технології, що спрощує обробку великих обсягів даних і інтеграцію з різними джерелами інформації [13], [14].

Крім цього, мова постійно оновлюється і розвивається, отримуючи нові можливості, такі як record-типи, асинхронні потоки даних, що відповідає вимогам сучасних програмних систем. C# також добре підходить для створення веб-застосунків через ASP.NET, а для графічних інтерфейсів доступні Windows Forms, WPF і MAUI, що дозволяє розробляти як класичні десктопні, так і мобільні програми [13], [14].

Таким чином, C# є сучасною, універсальною та потужною мовою, яка поєднує безпеку, стабільність і гнучкість, що робить її оптимальним вибором для широкого спектра завдань, включаючи розробку систем управління бібліотекою програмних фрагментів [13], [14].

Вибір мови програмування C# для розробки програмного забезпечення, призначеного для управління бібліотекою програмних фрагментів, є доцільним через кілька важливих аспектів, що впливають на ефективність створення, підтримки та масштабування системи [13], [14].

По-перше, мова програмування C# забезпечує високу інтеграцію з платформою .NET, що відкриває доступ до широкого спектра готових бібліотек і компонентів, спрощуючи реалізацію функціоналу роботи з базами даних, керування версіями, пошуку та сортування коду [13], [14].

По-друге, мова має строгий типізований синтаксис і потужну систему об’єктно-орієнтованого програмування, що дозволяє створювати чітко структуровані модулі та компоненти, які легко підтримувати і розширювати в майбутньому [13], [14].

Використання мови C# сприяє підвищенню стабільності та безпеки програмного забезпечення, оскільки платформа забезпечує автоматичне керування пам’яттю, обробку винятків та захист від багатьох поширених помилок програмування. Крім того, наявність інструментів для розробки графічних інтерфейсів, таких як Windows Forms або WPF, дозволяє створювати зручні та інтуїтивно зрозумілі інтерфейси для користувачів, що особливо важливо для систем керування бібліотекою фрагментів, де інтерфейс повинен бути максимально зрозумілим та функціональним. C# також підтримує сучасні підходи до розробки, включаючи багатопоточність, асинхронне виконання та роботу з вебсервісами, що дозволяє забезпечити швидкий доступ до великого обсягу даних і ефективну роботу у колективі розробників [13], [14].

Таким чином, обрання C# поєднує простоту використання та гнучкість із потужними можливостями платформи .NET, що робить цю мову оптимальним вибором для створення надійного, масштабованого і функціонально багатого програмного забезпечення для управління бібліотекою програмних фрагментів [13], [14].

Обґрунтування вибору мови програмування для розробки програмного забезпечення для управління бібліотекою програмних фрагментів наведено у таблиці 2.1.

Таблиця 2.1 – Обґрунтування вибору мови програмування для розробки програмного забезпечення для управління бібліотекою програмних фрагментів

Критерій порівняння мов програмування	Мова програмування		
	C#	Java	Python
Простота розробки	+	+–	+
Інтеграція з базами даних	+	+	+–
Підтримка графічного інтерфейсу	+	+–	+–
Масштабованість та модульність	+	+	+–
Безпека та контроль помилок	+	+	+–

Отже, для реалізації програмного забезпечення для управління бібліотекою програмних фрагментів обрано мову програмування C#, яка є сучасною, універсальною та потужною мовою, яка поєднує безпеку, стабільність і гнучкість, що робить її оптимальним вибором для широкого спектра завдань, включаючи розробку програмних систем для управління бібліотекою фрагментів коду. Крім того, мова C# забезпечує високу інтеграцію з платформою .NET, що відкриває доступ до широкого спектра готових бібліотек і компонентів, спрощуючи реалізацію функціоналу роботи з базами даних, керування версіями, пошуку та сортування коду. Наявність інструментів для розробки графічних інтерфейсів, таких як Windows Forms або WPF, дозволяє створювати зручні та інтуїтивно зрозумілі інтерфейси для користувачів, що особливо важливо для систем керування бібліотекою фрагментів, де інтерфейс повинен бути максимально зрозумілим та функціональним.

2.2 Вибір середовища розробки для створення програмного забезпечення для управління бібліотекою програмних фрагментів

В якості середовища розробки для створення програмного забезпечення для управління бібліотекою програмних фрагментів було обрано середовище Visual Studio [15], [16].

Visual Studio забезпечує багатофункціональне інтегроване середовище, яке поєднує підтримку багатьох мов програмування, серед яких C#, VB.NET, F#, C++ та інші, що дозволяє легко поєднувати різні компоненти та модулі в одному проекті. Середовище підтримує автоматичне форматування коду, рефакторинг та підказки IntelliSense, що значно підвищує продуктивність розробника та зменшує ймовірність помилок [15], [16].

Важливою особливістю є інтеграція з системами контролю версій, такими як Git і Azure DevOps, що спрощує спільну роботу команди, відстеження змін та ведення історії розробки. Visual Studio також пропонує інструменти для створення тестів, аналізу покриття коду та профілювання продуктивності, що забезпечує високу якість та надійність програмного забезпечення. Для створення користувацьких інтерфейсів доступні Windows Forms, WPF і новіший MAUI, які дозволяють будувати як класичні десктопні, так і кросплатформенні мобільні додатки [15], [16].

Додатково середовище підтримує широкий спектр розширень і плагінів, що дозволяє адаптувати його під специфіку проєкту, автоматизувати рутинні завдання, інтегрувати зовнішні сервіси та бібліотеки. Наявність детальної документації, навчальних матеріалів і активної спільноти робить Visual Studio дружнім для новачків і водночас потужним інструментом для професійних розробників [15], [16].

Отже Visual Studio забезпечує комплексний набір засобів для розробки, тестування та підтримки програмного забезпечення, що дозволяє ефективно реалізовувати проекти будь-якої складності, включаючи системи управління бібліотекою програмних фрагментів [15], [16].

Для розробки програмного забезпечення для управління бібліотекою програмних фрагментів доцільно обрати середовище розробки Visual Studio через комплексну підтримку процесу створення, налагодження та впровадження програмних систем [15], [16].

Visual Studio надає інтегроване середовище, що поєднує потужний редактор коду, інструменти для відлагодження, профілювання та тестування програм, що дозволяє значно скоротити час розробки і підвищити її ефективність [15], [16].

Підтримка C# та платформи .NET забезпечує безшовну інтеграцію з усіма необхідними бібліотеками, фреймворками та технологіями для роботи з базами даних, управління версіями коду та реалізації складної логіки пошуку та сортування фрагментів [15], [16].

Visual Studio також пропонує можливості для побудови графічних інтерфейсів через Windows Forms, WPF або MAUI, що дозволяє створювати зручні та інтуїтивно зрозумілі інтерфейси для користувачів бібліотеки програмних фрагментів. Вбудовані інструменти для контролю версій та співпраці команди, такі як інтеграція з Git, дозволяють ефективно координувати роботу кількох розробників, запобігаючи дублюванню коду та конфліктам між версіями. Крім того, Visual Studio підтримує розширення та плагіни, що забезпечує гнучкість і можливість адаптувати середовище під специфічні потреби проєкту [15], [16].

У результаті використання Visual Studio забезпечує комплексну підтримку на всіх етапах розробки програмного забезпечення для управління бібліотекою фрагментів, підвищує продуктивність розробників, покращує якість коду та зручність користування системою, роблячи її оптимальним вибором для такого типу проектів [15], [16].

Обґрунтування вибору середовища розробки для створення програмного забезпечення для управління бібліотекою програмних фрагментів наведено у таблиці 2.2.

Таблиця 2.2 – Обґрунтування вибору середовища розробки для створення програмного забезпечення для управління бібліотекою програмних фрагментів

Критерій порівняння середовищ розробки	Середовища розробки		
	Visual Studio	MonoDevelop	SharpDevelop
Потужні інструменти налагодження	+	+–	–
Засоби побудови графічного інтерфейсу	+	+–	–
Інструменти тестування та профілювання	+	+–	+–
Інтеграція з різними системами контролю версій	+	+–	+–
Підтримка розширень і плагінів	+	+–	–

Таким чином, для створення програмного забезпечення для управління бібліотекою програмних фрагментів обрано середовище розробки Visual Studio, яке поєднує потужний редактор коду, інструменти для відлагодження, профілювання та тестування програм, що дозволяє значно скоротити час розробки і підвищити її ефективність. Підтримка C# та платформи .NET забезпечує безшовну інтеграцію з усіма необхідними бібліотеками, фреймворками та технологіями для роботи з базами даних, управління версіями коду та реалізації складної логіки пошуку та сортування фрагментів.

2.3 Висновки за розділом 2

Для реалізації програмного забезпечення для управління бібліотекою програмних фрагментів обрано мову програмування C#, яка є сучасною,

універсальною та потужною мовою, яка поєднує безпеку, стабільність і гнучкість, що робить її оптимальним вибором для широкого спектра завдань, включаючи розробку програмних систем для управління бібліотекою фрагментів коду. Крім того, мова C# забезпечує високу інтеграцію з платформою .NET, що відкриває доступ до широкого спектра готових бібліотек і компонентів, спрощуючи реалізацію функціоналу роботи з базами даних, керування версіями, пошуку та сортування коду.

Для створення програмного забезпечення для управління бібліотекою програмних фрагментів обрано середовище розробки Visual Studio, яке поєднує потужний редактор коду, інструменти для відлагодження, профілювання та тестування програм, що дозволяє значно скоротити час розробки і підвищити її ефективність. Підтримка C# та платформи .NET забезпечує безшовну інтеграцію з усіма необхідними бібліотеками, фреймворками та технологіями для роботи з базами даних, управління версіями коду та реалізації складної логіки пошуку та сортування фрагментів.

3 ОПИС ПРОГРАМИ

3.1 Структура програмного забезпечення для управління бібліотекою програмних фрагментів

Структура програмного забезпечення для управління бібліотекою програмних фрагментів побудована за принципами чіткого розмежування відповідальності між основними логічними та функціональними компонентами системи, що забезпечує зручність розробки, супроводу та масштабування вебзастосунку. В основу архітектури покладено підхід, за якого окремі частини системи відповідають за представлення даних, обробку бізнес-логіки та взаємодію з користувачем. Такий підхід дозволяє зменшити зв’язність між компонентами та підвищити стабільність роботи програмного забезпечення.

Структуру програмного забезпечення для управління бібліотекою програмних фрагментів наведено на рис. 3.1.

Файлова структура проєкту включає спеціалізований розділ, у якому зосереджені статичні ресурси, необхідні для коректного відображення та функціонування вебінтерфейсу. У цьому розділі розміщуються стилі оформлення, що відповідають за візуальне представлення елементів інтерфейсу та підсвічування синтаксису програмного коду, а також шрифти й графічні матеріали, які використовуються під час побудови сторінок системи. Наявність окремого сховища для таких ресурсів сприяє впорядкованості проєкту та спрощує внесення змін у зовнішній вигляд застосунку.

Логіка обробки запитів користувача реалізується через контролери, які виконують роль посередників між інтерфейсом і внутрішніми моделями даних. Саме на цьому рівні здійснюється приймання запитів, їх аналіз та передавання відповідних даних для подальшої обробки або відображення.

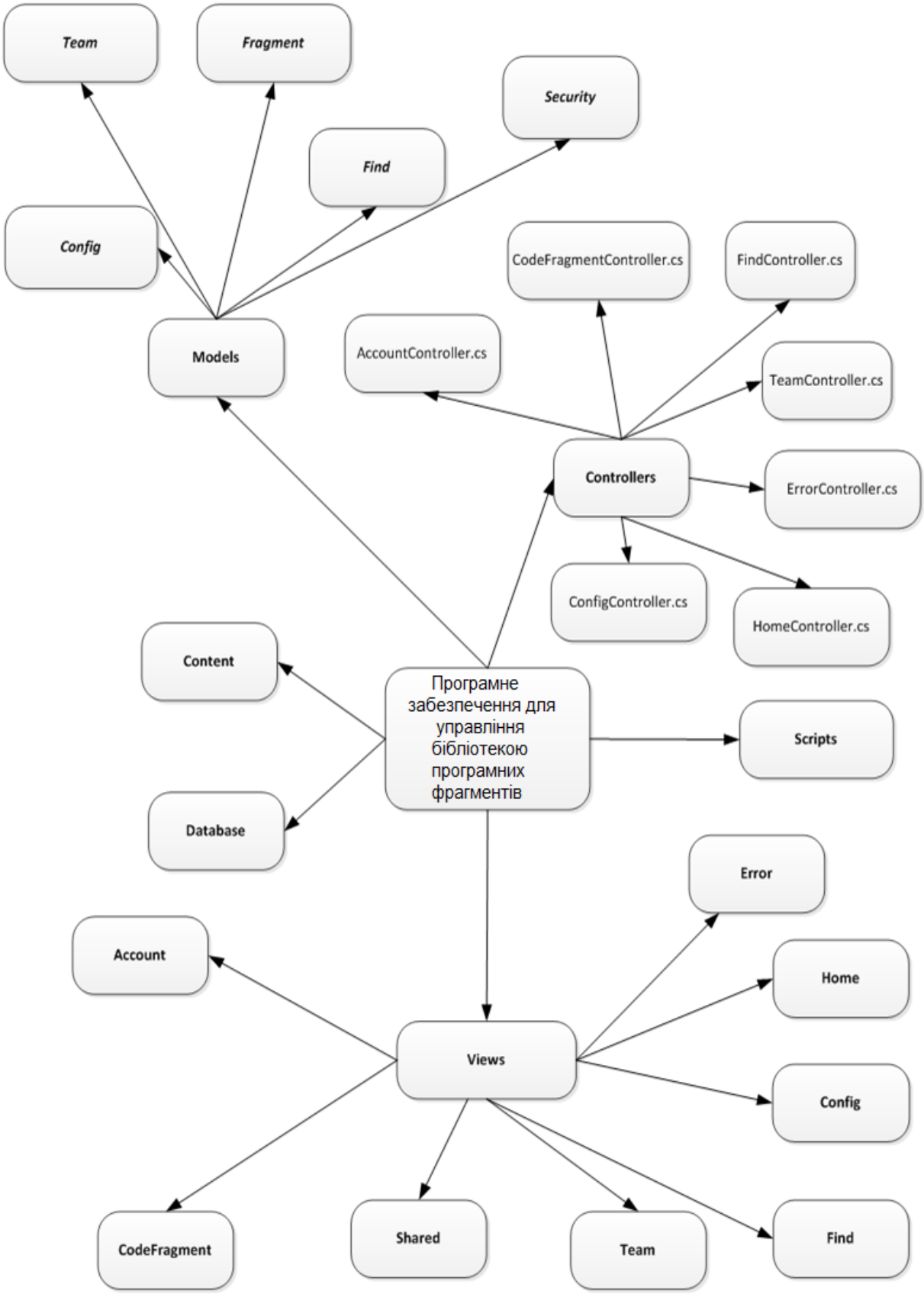

Рисунок 3.1 – Структура програмного забезпечення для управління бібліотекою програмних фрагментів

Контролери ПЗ для управління бібліотекою програмних фрагментів відповідають за роботу з обліковими записами користувачів, управління фрагментами програмного коду, реалізацію механізмів пошуку, адміністрування команд розробників, обробку помилок, зміну конфігураційних параметрів і загальну навігацію в межах вебзастосунку. Такий поділ функцій дозволяє зберігати логіку керування системою впорядкованою та зрозумілою.

Моделі вебзастосунку для управління бібліотекою програмних фрагментів містять опис предметної області та визначають правила взаємодії зі сховищем даних. У цьому шарі зосереджено класи, що відповідають за представлення програмних фрагментів, організацію командної роботи, реалізацію пошукових механізмів, управління налаштуваннями та забезпечення безпеки. Моделі інкапсулюють бізнес-логіку системи, забезпечуючи цілісність даних і коректність операцій незалежно від способу їх виклику. Такий підхід сприяє повторному використанню коду та спрощує його тестування.

Шар представлень програмного забезпечення для управління бібліотекою програмних фрагментів відповідає за формування інтерфейсу користувача та відображення інформації, отриманої від контролерів. Структура представлень організована відповідно до логіки контролерів, що забезпечує узгодженість між обробкою запитів і відображенням результатів. Окремо виділяються спільні елементи інтерфейсу, які використовуються в різних частинах системи, що дозволяє уникнути дублювання коду та підтримувати єдиний стиль оформлення.

Додаткову функціональність вебзастосунку для управління бібліотекою програмних фрагментів забезпечують клієнтські скрипти, які відповідають за динамічну взаємодію з користувачем, перевірку введених даних та підвищення зручності роботи із системою. Їх винесення в окремий структурний компонент дозволяє чітко відокремити клієнтську логіку від серверної частини.

Таким чином, запропонована структура програмного забезпечення для управління бібліотекою програмних фрагментів забезпечує логічну впорядкованість, гнучкість і масштабованість системи управління бібліотекою програмних фрагментів. Чіткий поділ на функціональні компоненти сприяє підвищенню якості коду, спрощує процес супроводу та створює передумови для подальшого розвитку і розширення функціональних можливостей вебзастосунку.

3.2 Функціонування програмного забезпечення для управління бібліотекою програмних фрагментів

Функціонування програмного забезпечення для управління бібліотекою програмних фрагментів реалізовано у вигляді вебзастосунку, взаємодія з яким здійснюється безпосередньо через інтерфейс користувача. Такий підхід дозволяє наочно продемонструвати принципи роботи системи накопичення та повторного використання програмного коду, а також забезпечує універсальний доступ до бібліотеки незалежно від робочого середовища розробника. Аналогічна логіка використовується і в разі інтеграції з середовищем розробки, де функціональність системи застосовується безпосередньо під час створення програмного забезпечення, однак у вебваріанті всі операції зосереджені у самостійному програмному продукті.

Початковим етапом роботи з вебзастосунком є взаємодія з механізмами керування обліковими записами, що забезпечують ідентифікацію користувачів та контроль доступу до функцій системи. Після входу в систему користувач отримує доступ до персоналізованого інтерфейсу, в межах якого відображаються доступні можливості відповідно до його ролі та налаштувань. У процесі експлуатації програмного забезпечення забезпечується збереження індивідуальних параметрів роботи, що дозволяє адаптувати систему до потреб конкретного користувача без необхідності повторної конфігурації.

Функціонування програмного забезпечення для управління бібліотекою програмних фрагментів подамо за допомогою схеми, зображеної на рис. 3.2.

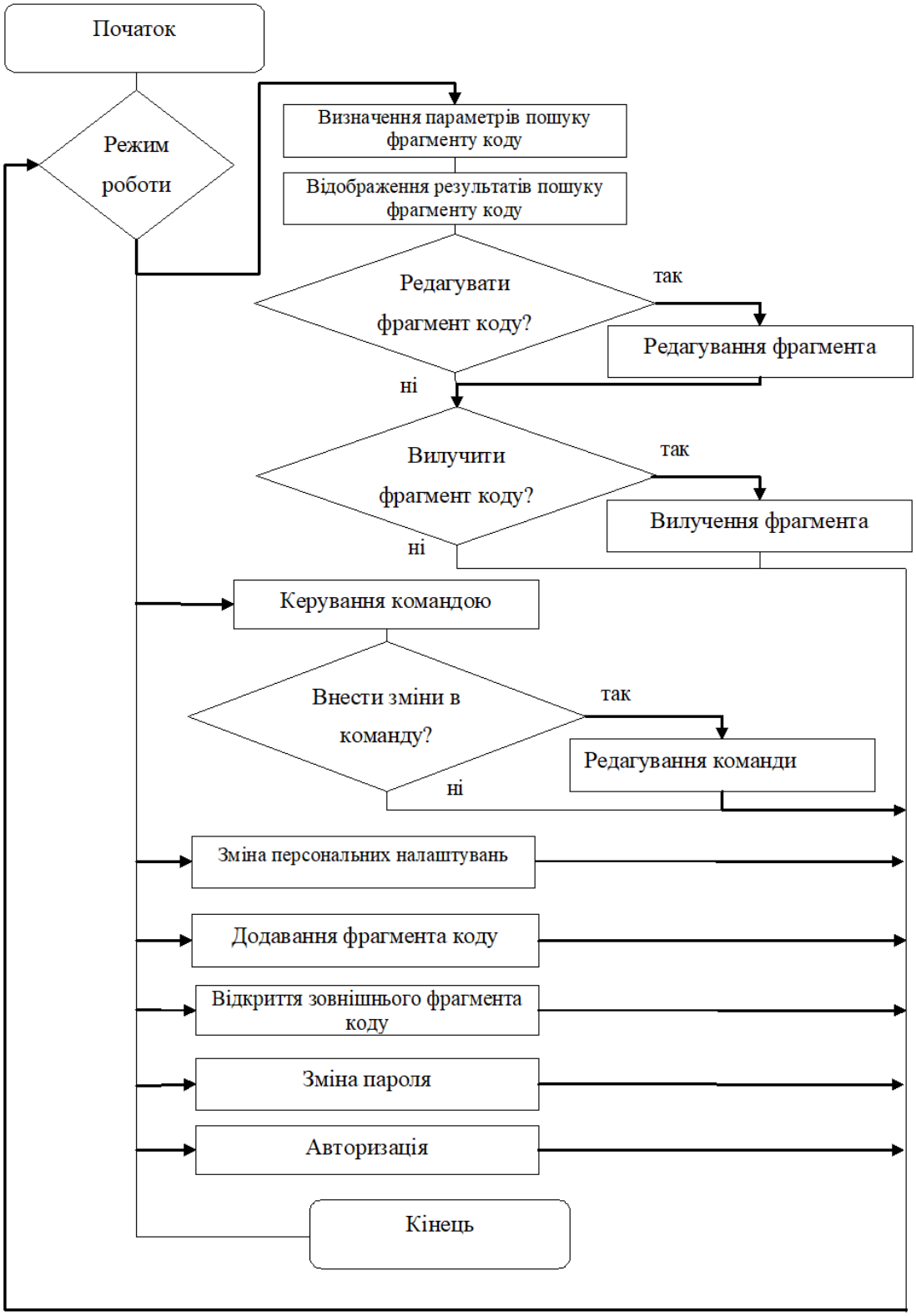

Рисунок 3.2 – Схема функціонування програмного забезпечення для управління бібліотекою програмних фрагментів

Значну роль у функціонуванні програмного забезпечення відіграють механізми організації командної роботи. Через відповідні елементи інтерфейсу реалізується створення команд, керування складом учасників та розмежування доступу до фрагментів програмного коду. Такий підхід забезпечує впорядковане зберігання фрагментів і сприяє ефективній співпраці між розробниками, зменшуючи ризик дублювання або некоректного використання програмних компонентів.

Центральним елементом роботи системи є управління фрагментами програмного коду. Програмне забезпечення надає можливість додавання нових фрагментів із зазначенням необхідних атрибутів, що полегшують подальший пошук та повторне використання. Збережені фрагменти стають доступними для перегляду, редагування або видалення відповідно до встановлених прав доступу. Окрема увага приділяється роботі з зовнішніми фрагментами, що дозволяє інтегрувати у бібліотеку програмний код, створений поза межами системи, з подальшим його впорядкуванням та використанням у спільних проєктах.

Важливою складовою функціонування вебзастосунку є реалізація пошукових механізмів, які забезпечують швидкий доступ до необхідних програмних фрагментів. Користувач має можливість задавати параметри пошуку, після чого система формує перелік результатів, з якими можна виконувати подальші дії. Такий підхід значно скорочує час на пошук потрібного коду та підвищує ефективність роботи з бібліотекою. Окремі функції управління безпекою, зокрема зміна пароля та персональні налаштування, забезпечують захист облікових записів і стабільність роботи системи.

Отже, функціонування програмного забезпечення для управління бібліотекою програмних фрагментів базується на інтеграції механізмів автентифікації, командної взаємодії, зберігання та пошуку програмного коду в єдиному вебсередовищі. Реалізований підхід забезпечує зручність використання, підвищує продуктивність розробників і створює умови для

ефективного повторного застосування програмних рішень у процесі розробки програмного забезпечення.

3.3 Розробка бази даних програмного забезпечення для управління бібліотекою програмних фрагментів

Схему бази даних програмного забезпечення для управління бібліотекою програмних фрагментів наведено на рис. 3.3.

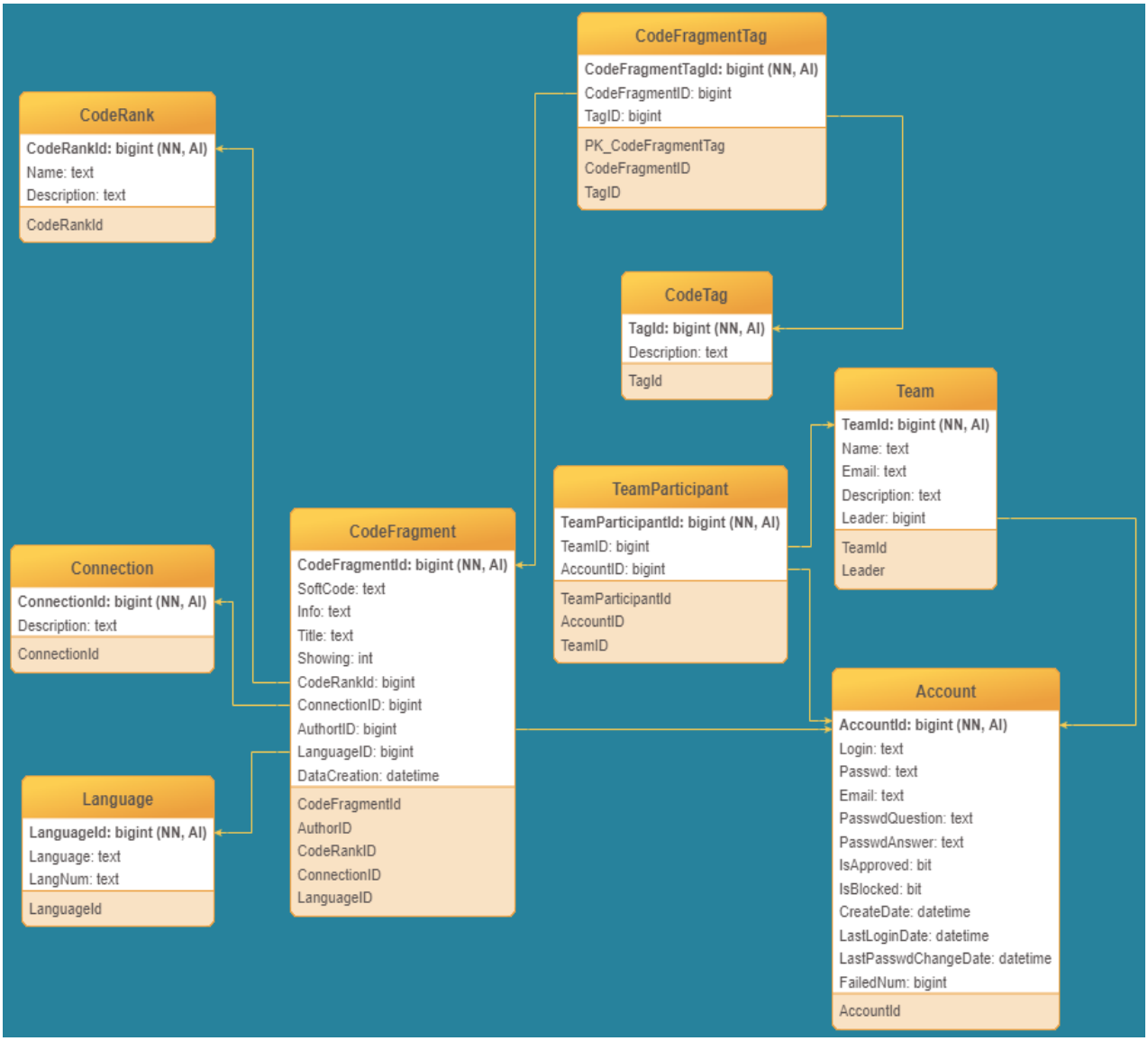

Рисунок 3.3 – Схема бази даних програмного забезпечення для управління бібліотекою програмних фрагментів

База даних програмного забезпечення для управління бібліотекою програмних фрагментів спроєктована з урахуванням необхідності централізованого зберігання програмного коду, метаданих про нього, а також інформації, що забезпечує організацію спільної роботи розробників. Вона виконує роль ядра системи, оскільки саме через неї реалізується збереження, пошук, класифікація та повторне використання фрагментів програмного коду. Структура бази даних орієнтована на підтримку цілісності даних, логічну впорядкованість зв’язків між сутностями та можливість подальшого розширення функціональних можливостей програмного забезпечення.

Центральне місце у структурі бази даних для управління бібліотекою програмних фрагментів займає сутність, що відповідає за зберігання фрагментів програмного коду. У ній зосереджено не лише текст коду, а й описова інформація, яка дозволяє ідентифікувати призначення фрагмента, автора, мову програмування, категорію та рівень доступності. Такий підхід забезпечує комплексне представлення кожного фрагмента як самостійного інформаційного об’єкта, придатного для аналізу, пошуку та повторного використання. Додатково зберігається інформація про час створення, що дозволяє відстежувати історію наповнення бібліотеки та аналізувати динаміку її розвитку.

Для забезпечення впорядкованої класифікації програмного коду у базі даних для управління бібліотекою програмних фрагментів передбачено окремі сутності, які описують мови програмування та категорії фрагментів. Зв’язок між фрагментами та мовами програмних фрагментів реалізовано таким чином, що кожен фрагмент асоціюється лише з однією мовою програмування, тоді як одна мова може відповідати значній кількості фрагментів. Аналогічний підхід застосовано і до категорій, що дозволяє групувати код за функціональним призначенням або рівнем складності, зберігаючи при цьому простоту логічної моделі даних.

Окремий механізм групування реалізується через сутність, яка відповідає за об’єднання взаємопов’язаних фрагментів програмного коду. Така

організація дає змогу формувати логічні набори фрагментів, які доцільно використовувати разом, що підвищує зручність повторного застосування готових рішень. Кожен фрагмент може бути включений лише до однієї такої групи, що спрощує інтерпретацію зв’язків і забезпечує коректність виведення пов’язаних елементів.

Значна увага у структурі бази даних для управління бібліотекою програмних фрагментів приділена зберіганню інформації про користувачів і команди розробників. Дані про облікові записи включають як ідентифікаційну та контактну інформацію, так і атрибути, пов’язані з безпекою, автентифікацією та контролем доступу. Це дозволяє реалізувати механізми авторства фрагментів, обмеження доступу та персоналізації роботи з бібліотекою. Взаємодія між користувачами та командами організована таким чином, що один користувач може керувати кількома командами, а також одночасно входити до складу різних груп розробників. Для цього у базі даних передбачено проміжну сутність, яка забезпечує коректну реалізацію складних зв’язків між користувачами та командами.

Для підвищення ефективності пошуку та гнучкості опису програмних фрагментів у базі даних для управління бібліотекою програмних фрагментів використовується механізм тегів. Один фрагмент може бути охарактеризований кількома тегами, а один тег, у свою чергу, може застосовуватися до різних фрагментів. Така багатозначна семантична прив’язка реалізується через окрему зв’язувальну сутність, що дозволяє зберігати структуру даних у нормалізованому вигляді та уникати дублювання інформації.

Отже, база даних програмного забезпечення для управління бібліотекою програмних фрагментів є логічно структурованою та функціонально насиченою системою зберігання даних. Її архітектура забезпечує цілісність, гнучкість і масштабованість, створюючи надійне підґрунтя для ефективного накопичення, класифікації та повторного

використання програмного коду в умовах індивідуальної та командної розробки.

3.4 Проєктування інтерфейсу програмного забезпечення для управління бібліотекою програмних фрагментів

Проєктування інтерфейсу взаємодії користувача з програмним забезпеченням для управління бібліотекою програмних фрагментів виконано з урахуванням вимог технічного завдання. При розробленні програмного забезпечення для управління бібліотекою програмних фрагментів враховані функціональні вимоги до програми:

– підтримка можливості збереження програмних фрагментів та пов’язаних з ними описів у базі даних системи;

– підтримка можливості редагування, оновлення та видалення програмних фрагментів і їхніх метаданих;

– можливість класифікації програмних фрагментів за мовами програмування, призначенням та іншими ознаками;

– можливість пошуку програмних фрагментів за ключовими словами, категоріями та змістом опису;

– можливість контролю доступу користувачів до операцій перегляду, редагування та адміністрування бібліотеки;

– забезпечення коректної обробки помилок і інформування користувача про результати виконання операцій у системі.

Інтерфейс форми пошуку програмних фрагментів у внутрішній базі даних наведено на рис. 3.4.

Інтерфейс сторінки відображення результатів пошуку програмних фрагментів наведено на рис. 3.5.

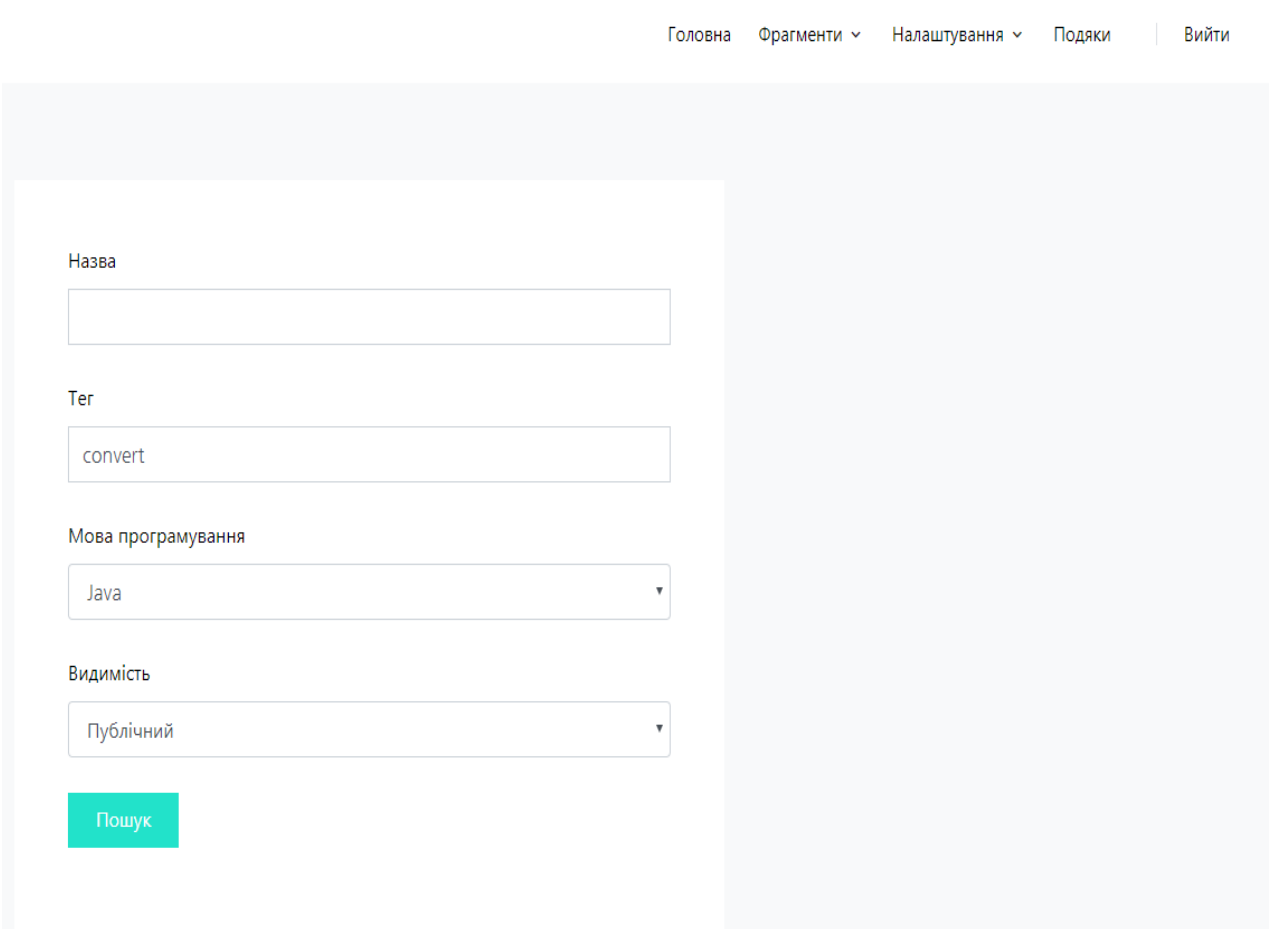

Рисунок 3.4 – Інтерфейс форми пошуку програмних фрагментів у внутрішній базі даних

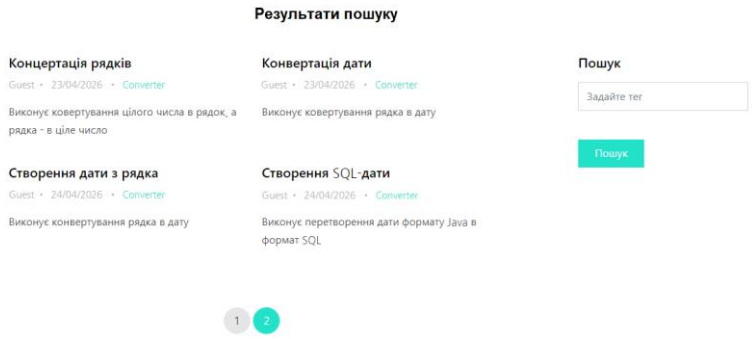

Рисунок 3.5 – Інтерфейс сторінки відображення результатів пошуку програмних фрагментів

Сторінку перегляду та редагування програмного фрагменту з внутрішньої бази наведено на рис. 3.6.

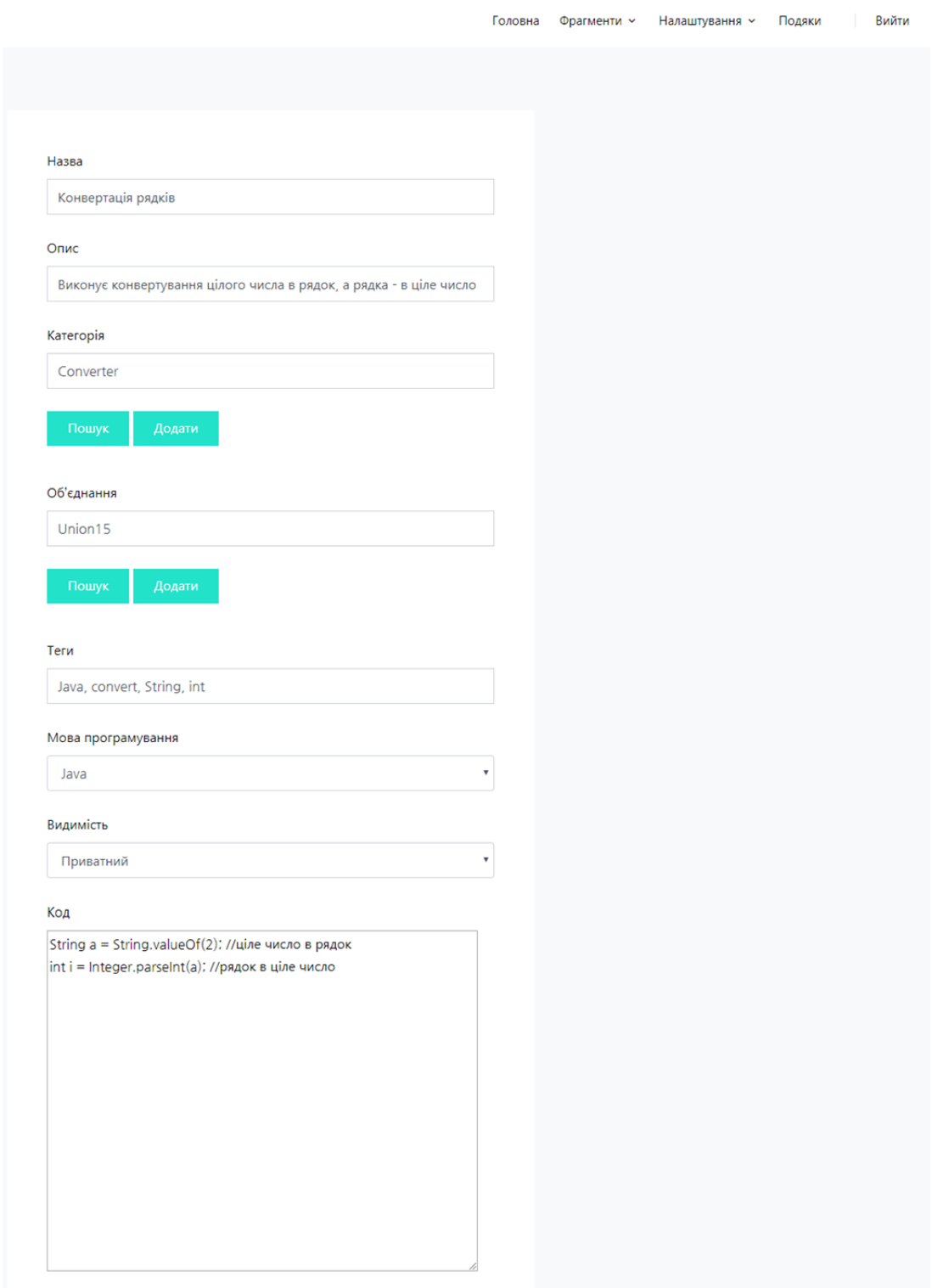

Рисунок 3.6 – Сторінка перегляду та редагування програмного фрагменту з внутрішньої бази

Форму авторизації наведено на рис. 3.7.

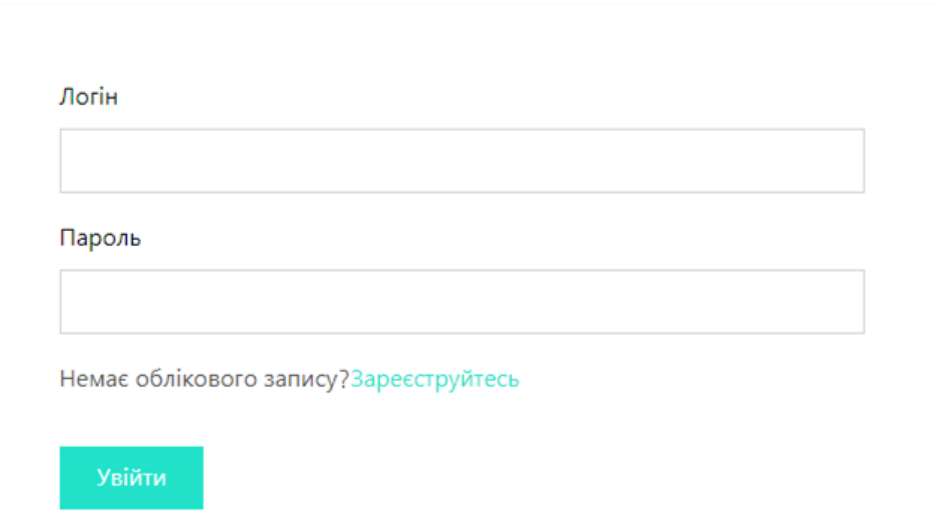

Рисунок 3.7 – Форма авторизації

Форму відкриття програмного фрагменту з зовнішнього джерела наведено на рис. 3.8.

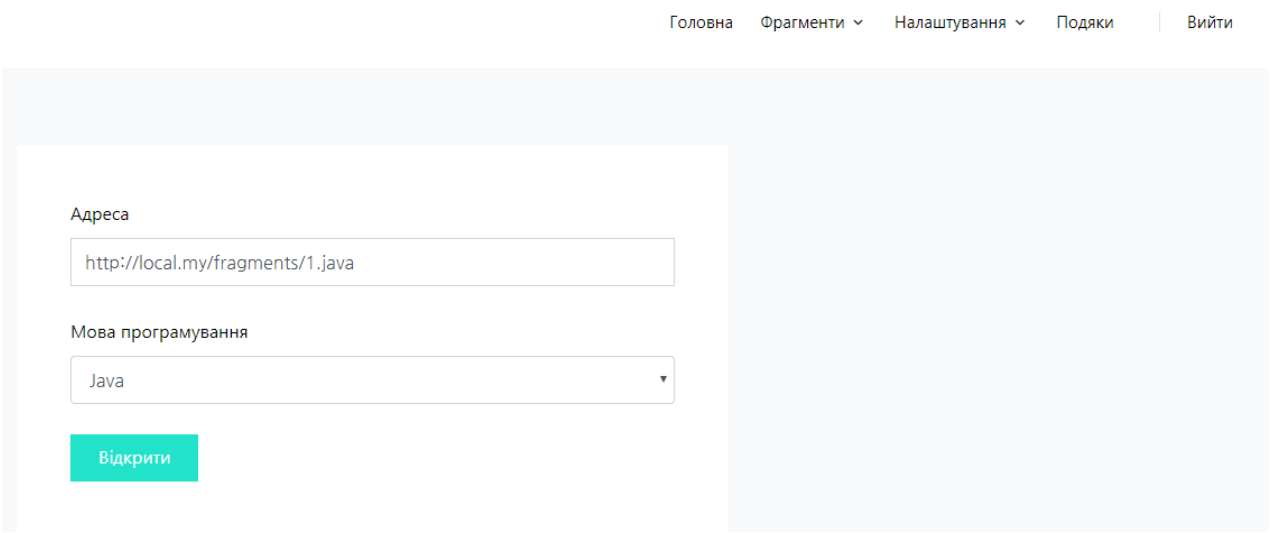

Рисунок 3.8 – Форма відкриття програмного фрагменту з зовнішнього джерела

3.5 Висновки за розділом 3

Розроблено програмне забезпечення для управління бібліотекою програмних фрагментів.

Запропоновано структуру програмного забезпечення для управління бібліотекою програмних фрагментів, яка побудована за принципами чіткого

розмежування відповідальності між основними логічними та функціональними компонентами системи, що забезпечує зручність розробки, супроводу та масштабування вебзастосунку. В основу архітектури покладено підхід, за якого окремі частини системи відповідають за представлення даних, обробку бізнес-логіки та взаємодію з користувачем. Такий підхід дозволяє зменшити зв’язність між компонентами та підвищити стабільність роботи програмного забезпечення.

Описано функціонування програмного забезпечення для управління бібліотекою програмних фрагментів. Визначено, що функціонування програмного забезпечення для управління бібліотекою програмних фрагментів базується на інтеграції механізмів автентифікації, командної взаємодії, зберігання та пошуку програмного коду в єдиному вебсередовищі. Реалізований підхід забезпечує зручність використання, підвищує продуктивність розробників і створює умови для ефективного повторного застосування програмних рішень у процесі розробки програмного забезпечення.

Розроблено базу даних програмного забезпечення для управління бібліотекою програмних фрагментів, яка спроєктована з урахуванням необхідності централізованого зберігання програмного коду, метаданих про нього, а також інформації, що забезпечує організацію спільної роботи розробників. Вона виконує роль ядра системи, оскільки саме через неї реалізується збереження, пошук, класифікація та повторне використання фрагментів програмного коду. Структура бази даних орієнтована на підтримку цілісності даних, логічну впорядкованість зв’язків між сутностями та можливість подальшого розширення функціональних можливостей програмного забезпечення.

Виконано проєктування інтерфейсу взаємодії користувача з програмним забезпеченням для управління бібліотекою програмних фрагментів.

4 ЕКСПЛУАТАЦІЯ, ТЕСТУВАННЯ ТА ЕКСПЕРИМЕНТАЛЬНЕ ДОСЛІДЖЕННЯ ПРОГРАМИ

4.1 Призначення й умови застосування програми

Програма призначена для управління бібліотекою програмних фрагментів. Програма написана на мові C#, яка є сучасною, універсальною та потужною мовою, яка поєднує безпеку, стабільність і гнучкість, що робить її оптимальним вибором для широкого спектра завдань, включаючи розробку програмних систем для управління бібліотекою фрагментів коду. В якості середовища розробки для створення програмного забезпечення для управління бібліотекою програмних фрагментів обрано Visual Studio, яке поєднує потужний редактор коду, інструменти для відлагодження, профілювання та тестування програм, що дозволяє значно скоротити час розробки і підвищити її ефективність.

Програмне забезпечення для управління бібліотекою програмних фрагментів забезпечує виконання таких функцій:

– підтримка можливості збереження програмних фрагментів та пов’язаних з ними описів у базі даних системи;

– підтримка можливості редагування, оновлення та видалення програмних фрагментів і їхніх метаданих;

– можливість класифікації програмних фрагментів за мовами програмування, призначенням та іншими ознаками;

– можливість пошуку програмних фрагментів за ключовими словами, категоріями та змістом опису;

– можливість контролю доступу користувачів до операцій перегляду, редагування та адміністрування бібліотеки;

– забезпечення коректної обробки помилок і інформування користувача про результати виконання операцій у системі.

Для роботи програмного забезпечення для управління бібліотекою програмних фрагментів потрібні такі технічні та програмні ресурси:

обчислювальна система з процесором загального призначення з тактовою частотою не нижче середнього сучасного рівня, що забезпечує стабільне виконання серверної та клієнтської логіки веб-додатка, а також оперативна пам’ять обсягом не менше кількох гігабайтів, достатня для одночасної роботи вебсервера, системи керування базами даних і браузера користувача. Для зберігання програмних фрагментів, метаданих та службової інформації необхідна наявність вільного дискового простору, обсяг якого визначається інтенсивністю використання системи та кількістю збереженого коду, при цьому рекомендується застосування швидких накопичувачів для підвищення продуктивності доступу до даних.

З програмного боку функціонування системи потребує встановленої операційної системи, сумісної з платформою .NET, що забезпечує виконання серверної частини додатка та підтримку сучасних мережевих протоколів. Для коректної роботи вебзастосунку необхідне наявне середовище виконання .NET відповідної версії, а також вебсервер, здатний обробляти HTTP-запити та забезпечувати взаємодію клієнтської і серверної частин програми. Робота з базою даних вимагає встановленої системи керування базами даних реляційного типу, яка забезпечує збереження, цілісність і захист інформації про програмні фрагменти, користувачів і команди розробників.

На стороні користувача для доступу до функціоналу програмного забезпечення достатньо наявності сучасного веббраузера з підтримкою стандартів HTML, CSS і JavaScript, що дозволяє працювати з інтерфейсом без встановлення додаткових клієнтських компонентів. Сукупність зазначених апаратних і програмних вимог забезпечує стабільну роботу програмного забезпечення, його доступність для широкого кола користувачів та можливість ефективного використання бібліотеки програмних фрагментів у процесі розробки програмного забезпечення.

Функціональні характеристики розробленого програмного забезпечення для управління бібліотекою програмних фрагментів наведено у

технічному завданні (додаток А). Фрагмент тексту програмного забезпечення для управління бібліотекою програмних фрагментів наведено у додатку Б.

4.2 Характеристики програми для управління бібліотекою програмних фрагментів

Програмне забезпечення для управління бібліотекою програмних фрагментів характеризується орієнтацією на підвищення ефективності процесів розробки шляхом централізованого зберігання та впорядкування програмного коду. Його архітектура побудована з урахуванням потреб як індивідуальних розробників, так і команд, що забезпечує гнучкість у використанні та можливість адаптації до різних організаційних моделей роботи. Функціонування системи базується на чіткому розмежуванні ролей і прав доступу, що дозволяє контролювати використання програмних фрагментів і підтримувати цілісність бібліотеки.

Важливою характеристикою такого програмного забезпечення є підтримка повторного використання коду, що досягається завдяки наявності механізмів класифікації, семантичного опису та контекстного пошуку. Фрагменти програмного коду зберігаються разом із супровідною інформацією, яка полегшує їх ідентифікацію, аналіз і інтеграцію в нові проєкти. Це сприяє зниженню кількості помилок, пов’язаних із дублюванням або некоректним застосуванням коду, а також підвищує загальну якість програмних продуктів.

Програмне забезпечення відзначається зручністю взаємодії з користувачем, що забезпечується інтуїтивно зрозумілим інтерфейсом і логічною організацією функціональних можливостей. Реалізація веб-орієнтованого доступу дозволяє працювати із системою незалежно від конкретного робочого середовища, зберігаючи єдиний інформаційний простір для всіх учасників розробки. При цьому особлива увага приділяється захисту даних, автентифікації користувачів і збереженню конфіденційності інформації.

Ще однією характерною рисою є масштабованість і відкритість до розширення, що дозволяє доповнювати систему новими функціональними можливостями без порушення її базової логіки. Завдяки модульній побудові забезпечується простота супроводу, тестування та подальшого розвитку програмного забезпечення. У сукупності ці характеристики визначають програмне забезпечення для управління бібліотекою програмних фрагментів як ефективний інструмент, що сприяє оптимізації процесів розробки та раціональному використанню програмного коду.

4.3 Інструкція по експлуатації програми

4.3.1 Звернення до програми

Для запуску програмного забезпечення для управління бібліотекою програмних фрагментів необхідно запустити відповідний файл.

4.3.2 Вхідні й вихідні дані

Вхідними даними до програмного забезпечення для управління бібліотекою програмних фрагментів є облікові відомості для доступу до системи, текст програмних фрагментів, їх назви, описи, параметри, а також пошукові запити й налаштування персонального та командного середовища роботи.

Вихідними даними програмного забезпечення для управління бібліотекою програмних фрагментів є результати обробки запитів користувача, що відображаються у вигляді структурованої інформації в інтерфейсі системи. До них відносяться переліки знайдених програмних фрагментів відповідно до заданих умов пошуку, відображення вмісту фрагментів програмного коду.

4.3.3 Повідомлення

Користувач програмного забезпечення для управління бібліотекою програмних фрагментів може отримати такі повідомлення: повідомлення про успішне виконання дій, пов’язаних із додаванням, редагуванням або видаленням програмних фрагментів, системні сповіщення про зміну стану облікового запису чи параметрів доступу, а також попередження або повідомлення про помилки у разі некоректного введення даних, відсутності необхідних прав або виникнення внутрішніх збоїв у роботі системи.

4.4 Виконання програмного забезпечення для управління бібліотекою програмних фрагментів

Після запуску програмного забезпечення для управління бібліотекою програмних фрагментів користувачу відображається головна сторінка, де для доступу до основних функцій програмі необхідно авторизуватися (рис. 4.1), ввівши логін та пароль.

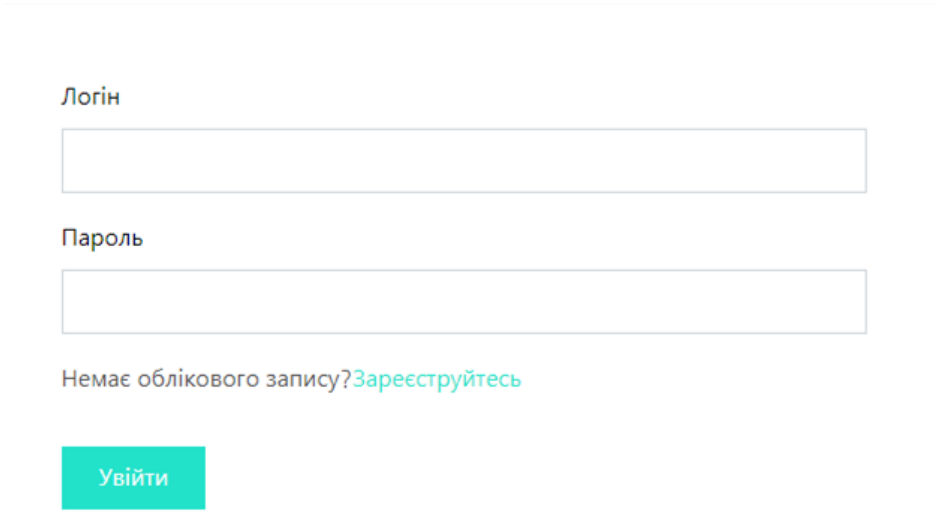

Рисунок 4.1 – Форма авторизації програмного забезпечення для управління бібліотекою програмних фрагментів

Після авторизації для пошуку програмних фрагментів користувачу необхідно у пункті Фрагменти головному меню обрати підпункт пошук та у формі пошуку (рис. 4.2) ввести дані пошукового запиту.

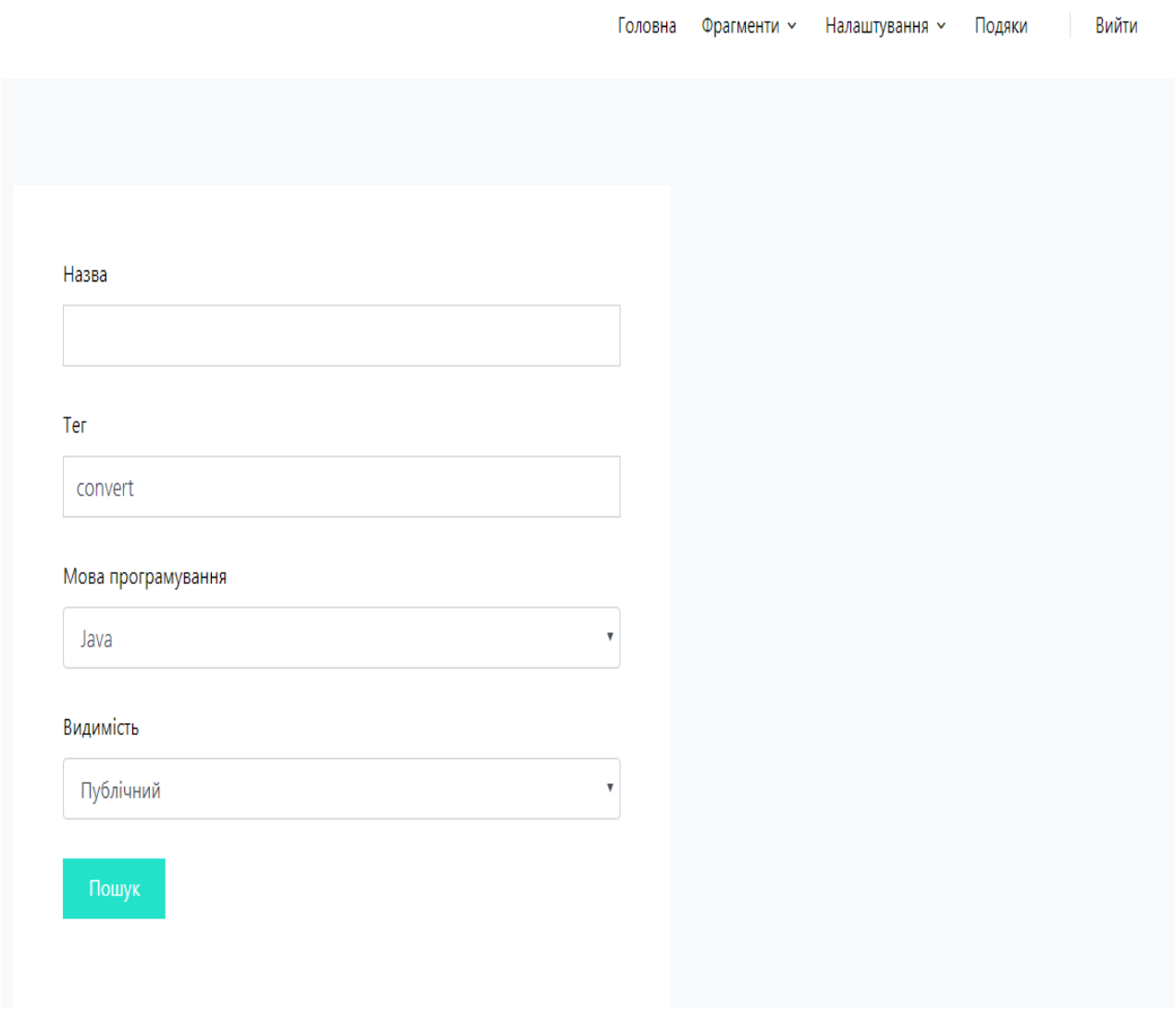

Рисунок 4.2 – Форма пошуку програмних фрагментів у внутрішній базі даних

Після пошуку система видає результати, що містять назви програмних фрагментів, їх опис та деяку іншу інформацію (рис. 4.3).

При виборі конкретного фрагменту відкривається сторінка перегляду та редагування відповідного програмного фрагменту (рис. 4.4), де користувач може змінити його назву, опис, категорію та інші параметри.

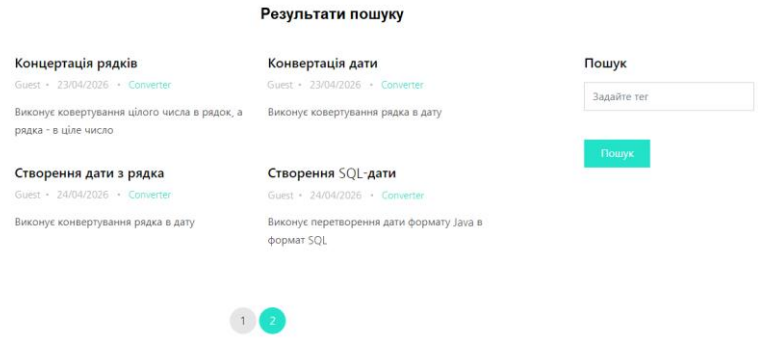

Рисунок 4.3 – Сторінка відображення результатів пошуку програмних фрагментів

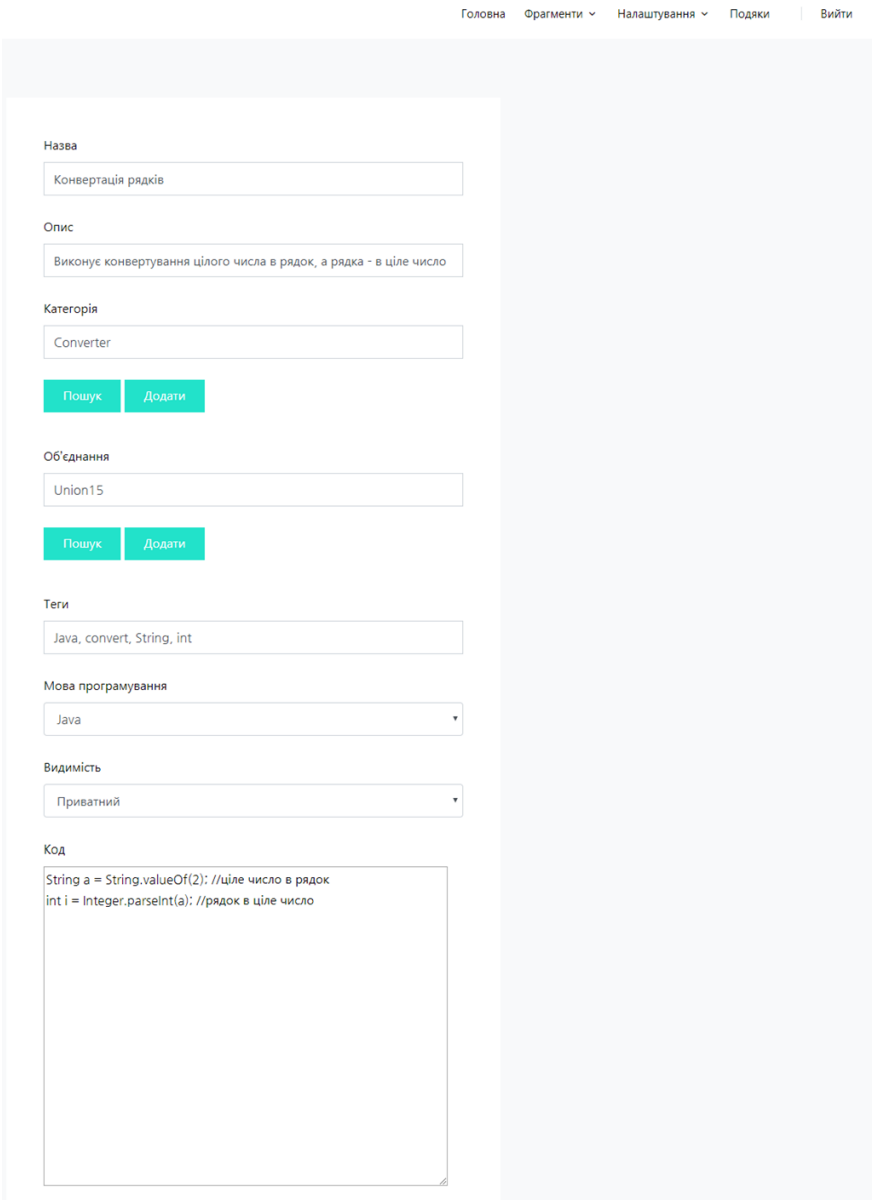

Рисунок 4.4 – Сторінка перегляду та редагування програмного фрагменту з внутрішньої бази

При виборі пункту Додавання фрагменту у меню Фрагменти відобразиться форма (рис. 4.5), де користувач може додати фрагмент коду для його подальшого використання, а також певну інформацію про доданий фрагмент.

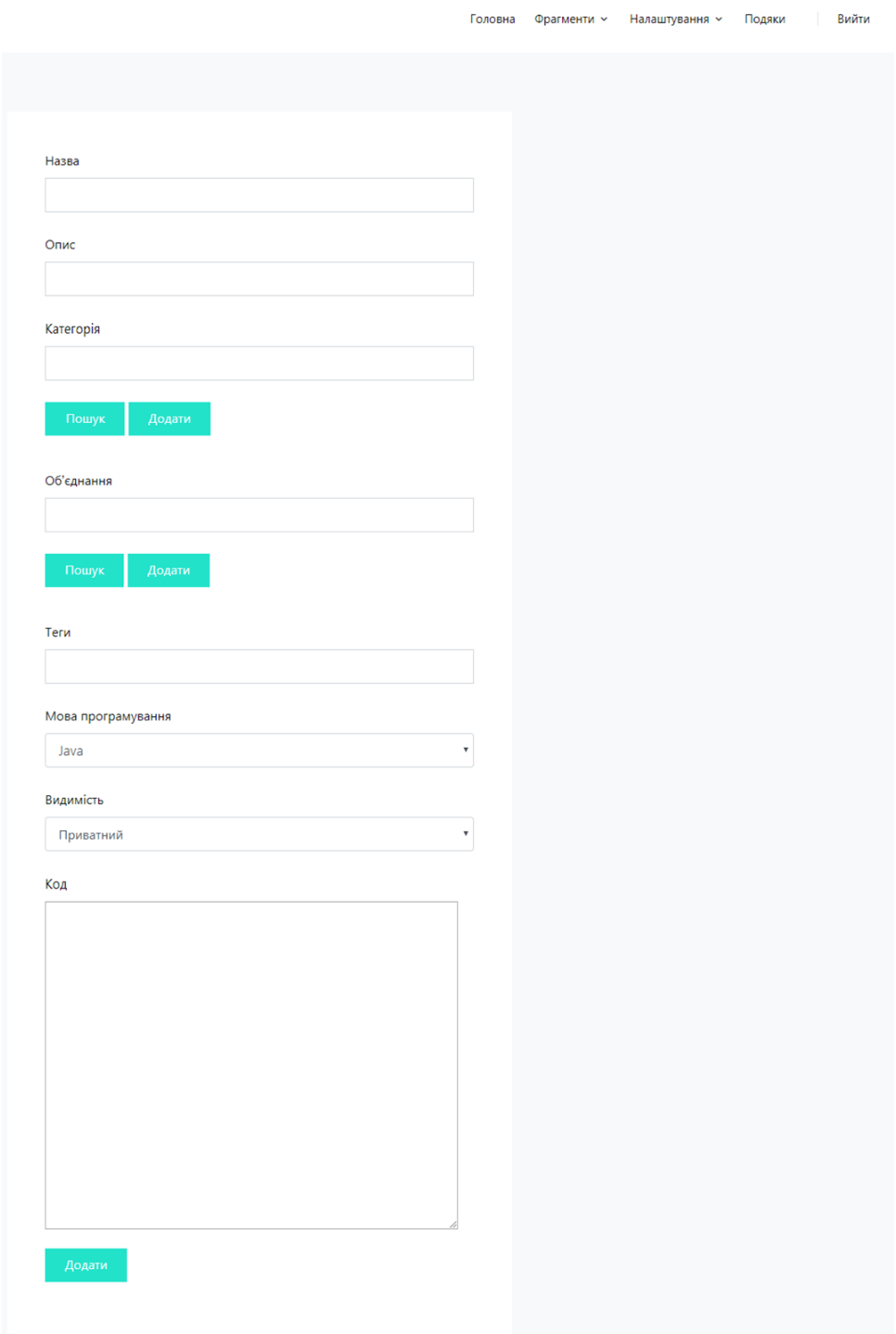

Рисунок 4.5 – Додавання програмного фрагменту

При виборі пункту Відкрити зовнішній фрагмент у меню Фрагменти відобразиться форма (рис. 4.6), де користувач повинен ввести http-адресу для завантаження фрагменту коду із зовнішнього джерела.

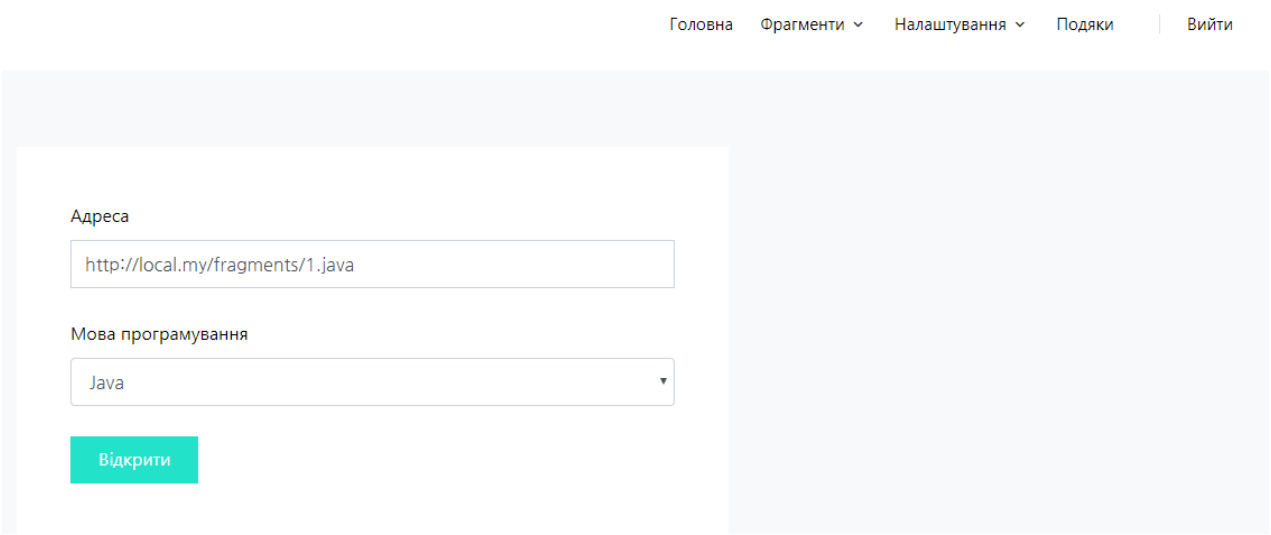

Рисунок 4.6 – Форма відкриття програмного фрагменту з зовнішнього джерела

За допомогою пункту меню Налаштування (рис. 4.7) є можливість змінити персональні налаштування користувача.

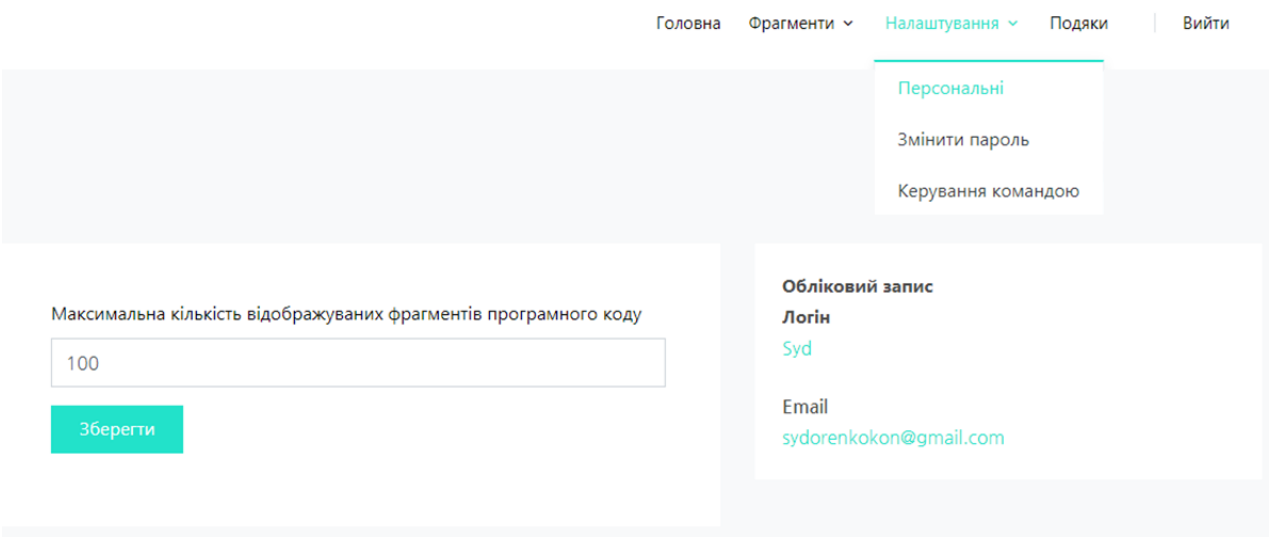

Рисунок 4.7 – Сторінка налаштувань користувача

Для доступу окремих категорій користувачів до програмних фрагментів адміністратор (керівник проєктів) може створювати команди, а

також додавати та вилучати з них користувачів. Користувачі відповідних команд мають доступ до програмних фрагментів (рис. 4.8).

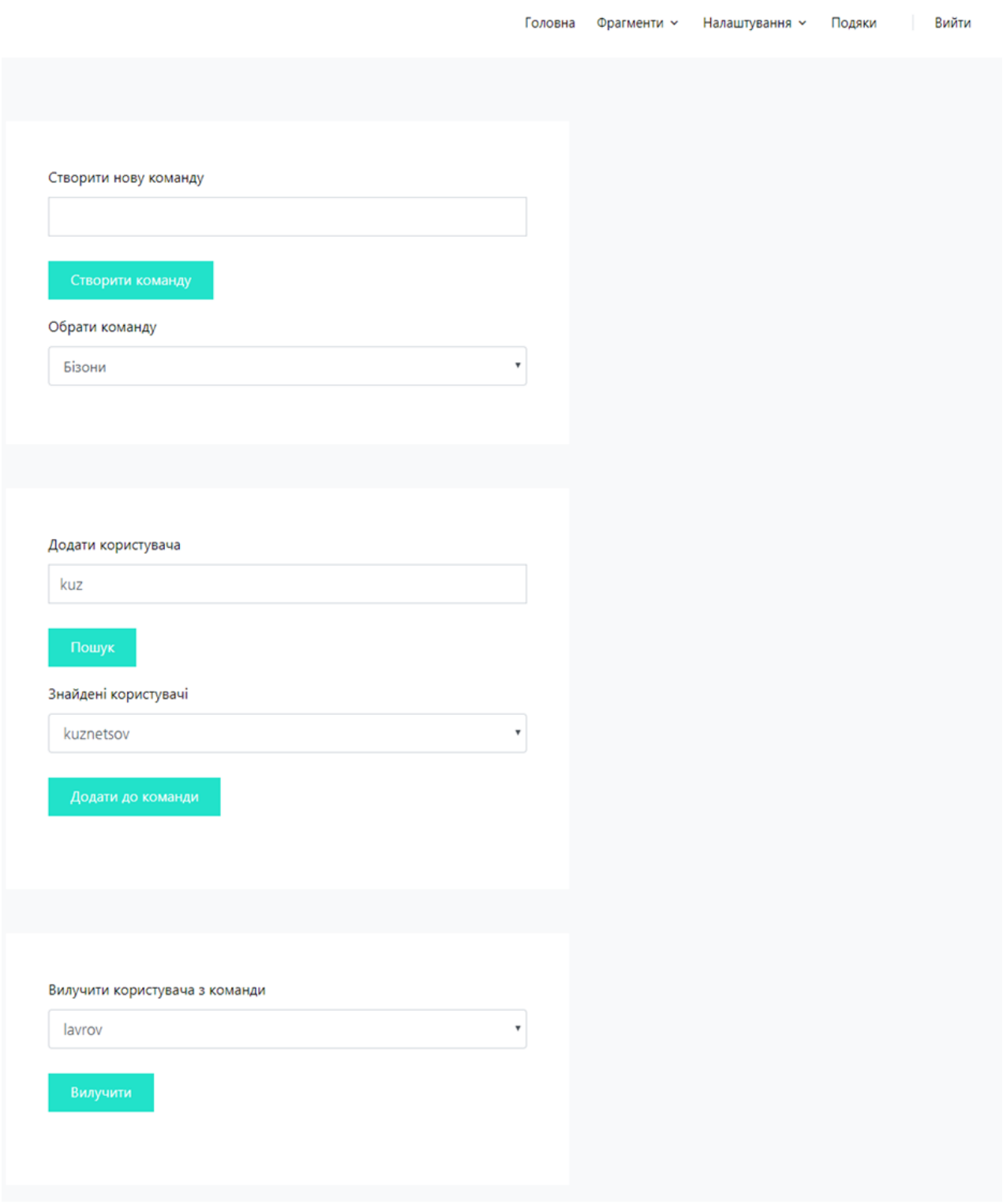

Рисунок 4.8 – Сторінка керування доступом користувачів до програмних фрагментів

4.5 Тестування програмного забезпечення для управління бібліотекою програмних фрагментів

Виконано тестування програмного забезпечення для управління бібліотекою програмних фрагментів.

Виявлено, що програма для управління бібліотекою програмних фрагментів функціонує правильно та злагоджено. Вона реалізує всі функціональні вимоги та успішно виконує свою основну задачу управління бібліотекою програмних фрагментів.

Розроблене програмне забезпечення дозволяє забезпечувати автоматизацію процесів, пов’язаних з підтримкою процесу управління бібліотекою програмних фрагментів.

4.6 Висновки за розділом 4

Описано програмне забезпечення для управління бібліотекою програмних фрагментів. Виконано тестування розробленого програмного забезпечення для управління бібліотекою програмних фрагментів. Результати тестування програмного забезпечення показали, що розроблена програма дозволяє забезпечити автоматизацію процесів, пов’язаних з підтримкою процесу управління бібліотекою програмних фрагментів.

ВИСНОВКИ

В ході виконання дипломної кваліфікаційної роботи бакалавра було проаналізовано та досліджено процес розробки програмного забезпечення для управління бібліотекою програмних фрагментів.

Визначено, що програмне забезпечення для управління бібліотекою програмних фрагментів є класом спеціалізованих інформаційних систем, призначених для централізованого зберігання, впорядкування та повторного використання типових елементів програмного коду. Такі програмні системи забезпечують формування єдиного структурованого сховища фрагментів, що можуть застосовуватися під час розробки програмного забезпечення незалежно від конкретного проєкту або середовища виконання.

За результатами проведеного аналізу зроблено висновок, що у наш час існує досить багато програмних засобів для управління бібліотекою програмних фрагментів. Проте деякі програмні засоби для управління бібліотеками програмних фрагментів можуть бути досить складними для інтеграції у вже існуючі проєкти, оскільки часто виникає потреба у налаштуванні специфічних робочих процесів, сумісності з різними середовищами розробки та мовами програмування. Крім того, використання деяких програмних засобів для централізованого зберігання коду створює ризики безпеки та доступності, оскільки помилки або збої у програмній системі можуть впливати на роботу всього колективу розробників. Важливою проблемою виступає й навчання користувачів, оскільки ефективне використання бібліотеки потребує певного рівня знань і дисципліни у підтриманні структури та стандартів. У підсумку, незважаючи на очевидні переваги, програмне забезпечення для управління бібліотекою програмних фрагментів залишається не позбавленим обмежень, що вимагає ретельного планування та регулярного контролю його експлуатації. Тому актуальною є розробка програмного забезпечення для управління бібліотекою програмних фрагментів.

Сформульовано функціональні вимоги до програмного забезпечення для управління бібліотекою програмних фрагментів.

Для реалізації програмного забезпечення для управління бібліотекою програмних фрагментів обрано мову програмування C#, яка є сучасною, універсальною та потужною мовою, яка поєднує безпеку, стабільність і гнучкість, що робить її оптимальним вибором для широкого спектра завдань, включаючи розробку програмних систем для управління бібліотекою фрагментів коду. Крім того, мова C# забезпечує високу інтеграцію з платформою .NET, що відкриває доступ до широкого спектра готових бібліотек і компонентів, спрощуючи реалізацію функціоналу роботи з базами даних, керування версіями, пошуку та сортування коду.

Для створення програмного забезпечення для управління бібліотекою програмних фрагментів обрано середовище розробки Visual Studio, яке поєднує потужний редактор коду, інструменти для відлагодження, профілювання та тестування програм, що дозволяє значно скоротити час розробки і підвищити її ефективність. Підтримка C# та платформи .NET забезпечує безшовну інтеграцію з усіма необхідними бібліотеками, фреймворками та технологіями для роботи з базами даних, управління версіями коду та реалізації складної логіки пошуку та сортування фрагментів.

Розроблено програмне забезпечення для управління бібліотекою програмних фрагментів.

Запропоновано структуру програмного забезпечення для управління бібліотекою програмних фрагментів, яка побудована за принципами чіткого розмежування відповідальності між основними логічними та функціональними компонентами системи, що забезпечує зручність розробки, супроводу та масштабування вебзастосунку. В основу архітектури покладено підхід, за якого окремі частини системи відповідають за представлення даних, обробку бізнес-логіки та взаємодію з користувачем. Такий підхід дозволяє зменшити зв’язність між компонентами та підвищити стабільність роботи програмного забезпечення.

Описано функціонування програмного забезпечення для управління бібліотекою програмних фрагментів. Визначено, що функціонування програмного забезпечення для управління бібліотекою програмних фрагментів базується на інтеграції механізмів автентифікації, командної взаємодії, зберігання та пошуку програмного коду в єдиному вебсередовищі. Реалізований підхід забезпечує зручність використання, підвищує продуктивність розробників і створює умови для ефективного повторного застосування програмних рішень у процесі розробки програмного забезпечення.

Розроблено базу даних програмного забезпечення для управління бібліотекою програмних фрагментів, яка спроєктована з урахуванням необхідності централізованого зберігання програмного коду, метаданих про нього, а також інформації, що забезпечує організацію спільної роботи розробників. Вона виконує роль ядра системи, оскільки саме через неї реалізується збереження, пошук, класифікація та повторне використання фрагментів програмного коду. Структура бази даних орієнтована на підтримку цілісності даних, логічну впорядкованість зв’язків між сутностями та можливість подальшого розширення функціональних можливостей програмного забезпечення.

Виконано проєктування інтерфейсу взаємодії користувача з програмним забезпеченням для управління бібліотекою програмних фрагментів.

Виконано тестування розробленого програмного забезпечення для управління бібліотекою програмних фрагментів. Результати тестування програмного забезпечення показали, що розроблена програма дозволяє забезпечити автоматизацію процесів, пов’язаних з підтримкою процесу управління бібліотекою програмних фрагментів.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Code Snippets. URL: https://www.uniccm.com/coding/code-snippets (date of access: 29.04.2026).

2. Built a Library of Code Snippets for Developers. URL: https://dev.to/technoph1le/i-built-a-library-of-code-snippets-d66 (date of access: 29.04.2026).

3. Best Code Snippet Manager Built for Modern Teams. URL: https://www.dhiwise.com/post/best-code-snippet-manager-built-for-modern-teams (date of access: 29.04.2026).

4. Code Snippet Manager Using Flask and SQLite. URL: https://www.researchgate.net/publication/396812583_Code_Snippet_Manager_Using_Flask_and_SQLite (date of access: 29.04.2026).

5. Walkthrough: Create a code snippet in Visual Studio. URL: https://learn.microsoft.com/en-us/visualstudio/ide/walkthrough-creating-a-code-snippet?view=visualstudio (date of access: 29.04.2026).

6. Snippet Manager. URL: https://snippetmanager.net/ (date of access: 29.04.2026).

7. Best Code Snippet Managers for Devs. URL: https://daily.dev/blog/7-best-code-snippet-managers-for-devs-2024 (date of access: 29.04.2026).

8. Best Code Snippets Manager with AI Features. URL: https://snappify.com/blog/best-code-snippets-manager (date of access: 29.04.2026).

9. Top 5 Code Snippet Managers to Boost Your Coding Efficiency. URL: https://bladedocs.com/dev-tools/top-5-code-snippet-managers-to-boost-your-coding-efficiency/ (date of access: 29.04.2026).

10. Masscode. URL: https://masscode.io/ (date of access: 29.04.2026).

11. Snipit. URL: https://snipit.io/ (date of access: 29.04.2026).

12. CodePen. URL: https://codepen.io/ (date of access: 29.04.2026).

13. C# language documentation. URL: https://learn.microsoft.com/en-us/dotnet/csharp/ (date of access: 29.04.2026).

14. Free C# Training Materials. URL: https://bestedlessons.org/free-c-sharp-training-materials/ (date of access: 29.04.2026).

15. Visual Studio tutorial. URL: https://www.tektutorialshub.com/visual-studio-tutorial/ (date of access: 29.04.2026).

16. Visual Studio IDE documentation. URL: https://learn.microsoft.com/en-us/visualstudio/ide/ (date of access: 29.04.2026).

ДОДАТОК А

Технічне завдання

Вступ

Програмне забезпечення може використовуватися для управління бібліотекою програмних фрагментів.

А.1 Підстава для розробки

Підставою для розробки є завдання на дипломну кваліфікаційну роботу на тему «Програмне забезпечення для управління бібліотекою програмних фрагментів», затверджене наказом Національного університету «Запорізька політехніка» № 139 від 7 квітня 2026 р.

А.2 Призначення розробки

Програмний продукт призначений для роботи з бібліотекою фрагментів програмних кодів.

А.3 Основні вимоги до програми, що розробляється

А.3.1 Вимоги до функціональних характеристик

Програмне забезпечення для управління бібліотекою програмних фрагментів забезпечує виконання таких функцій:

– підтримка можливості збереження програмних фрагментів та пов’язаних з ними описів у базі даних системи;

– підтримка можливості редагування, оновлення та видалення програмних фрагментів і їхніх метаданих;

– можливість класифікації програмних фрагментів за мовами програмування, призначенням та іншими ознаками;

– можливість пошуку програмних фрагментів за ключовими словами, категоріями та змістом опису;

– можливість контролю доступу користувачів до операцій перегляду, редагування та адміністрування бібліотеки;

– забезпечення коректної обробки помилок і інформування користувача про результати виконання операцій у системі.

А.3.2 Вимоги до інтерфейсу програми

Інтерфейс програмного забезпечення для управління бібліотекою програмних фрагментів повинен бути зручним для користувачів. Повинна бути забезпечена можливість роботи в візуальному режимі.

А.3.3 Вимоги до надійності

Програмне забезпечення для управління бібліотекою програмних фрагментів повинно забезпечити надійне функціонування.

А.3.4 Умови експлуатації

Для експлуатації програмного забезпечення для управління бібліотекою програмних фрагментів необхідна наявність персонального комп’ютера.

А.3.5 Вимоги до складу та параметрів технічний засобів

Для роботи програмного забезпечення для управління бібліотекою програмних фрагментів потрібні такі технічні та програмні ресурси: обчислювальна система з процесором загального призначення з тактовою частотою не нижче середнього сучасного рівня, що забезпечує стабільне

виконання серверної та клієнтської логіки веб-додатка, а також оперативна пам’ять обсягом не менше кількох гігабайтів, достатня для одночасної роботи вебсервера, системи керування базами даних і браузера користувача. Для зберігання програмних фрагментів, метаданих та службової інформації необхідна наявність вільного дискового простору, обсяг якого визначається інтенсивністю використання системи та кількістю збереженого коду, при цьому рекомендується застосування швидких накопичувачів для підвищення продуктивності доступу до даних.

З програмного боку функціонування системи потребує встановленої операційної системи, сумісної з платформою .NET, що забезпечує виконання серверної частини додатка та підтримку сучасних мережевих протоколів. Для коректної роботи вебзастосунку необхідне наявне середовище виконання .NET відповідної версії, а також вебсервер, здатний обробляти HTTP-запити та забезпечувати взаємодію клієнтської і серверної частин програми. Робота з базою даних вимагає встановленої системи керування базами даних реляційного типу, яка забезпечує збереження, цілісність і захист інформації про програмні фрагменти, користувачів і команди розробників.

А.3.6 Вимоги до маркування і пакування

Програма для управління бібліотекою програмних фрагментів може бути записана на будь-якому носії інформації.

На пакуванні повинна бути назва програми – «Програмне забезпечення для управління бібліотекою програмних фрагментів».

А.4 Вхідні дані до роботи

Вхідними даними до програмного забезпечення для управління бібліотекою програмних фрагментів є облікові відомості для доступу до системи, текст програмних фрагментів, їх назви, описи, параметри, а також

пошукові запити й налаштування персонального та командного середовища роботи.

Вихідними даними програмного забезпечення для управління бібліотекою програмних фрагментів є результати обробки запитів користувача, що відображаються у вигляді структурованої інформації в інтерфейсі системи. До них відносяться переліки знайдених програмних фрагментів відповідно до заданих умов пошуку, відображення вмісту фрагментів програмного коду.

ДОДАТОК Б

Фрагмент тексту програми

```
var bldr = WebApplication.CreateBuilder(args);
bldr.Services.AddControllersWithViews();
bldr.Services.AddSingleton<StrgSrv>();
bldr.Services.AddSingleton<FrgmntSrv>();

var pp = bldr.Build();

pp.UseStaticFiles();
pp.UseRouting();
pp.MapControllerRoute(
    name: "dflt",
    pattern: "{controller=Cntrl}/{action=Indx}/{id?}"
);

pp.Run();

public class StrgSrv
{
    private List<FrgmntMdl> dtLst = new List<FrgmntMdl>();
    private int nxt = 1;

    public List<FrgmntMdl> GtAll()
    {
        List<FrgmntMdl> rs = new List<FrgmntMdl>();
        foreach (var x in dtLst)
        {
            rs.Add(x);
        }
        return rs;
    }

    public FrgmntMdl GtByDntf(int dntf)
    {
        FrgmntMdl rs = null;
        foreach (var x in dtLst)
        {
            if (x.Dntf == dntf)
            {
                rs = x;
            }
        }
        return rs;
    }

    public void AddNw(string nmn, string cntnt, string tgs)
    {
        FrgmntMdl mdl = new FrgmntMdl();
        mdl.Dntf = nxt;
```

```
        mdl.Nmn = nmn;
        mdl.Cntnt = cntnt;
        mdl.Tgs = tgs;
        mdl.Crtd = System.DateTime.Now.ToString();
        mdl.Mdfd = mdl.Crtd;
        nxt = nxt + 1;
        dtLst.Add(mdl);
    }

    public void Updt(int dntf, string nmn, string cntnt, string
tgs)
    {
        foreach (var x in dtLst)
        {
            if (x.Dntf == dntf)
            {
                x.Nmn = nmn;
                x.Cntnt = cntnt;
                x.Tgs = tgs;
                x.Mdfd = System.DateTime.Now.ToString();
            }
        }
    }

    public void Rmv(int dntf)
    {
        FrgmntMdl trg = null;
        foreach (var x in dtLst)
        {
            if (x.Dntf == dntf)
            {
                trg = x;
            }
        }
        if (trg != null)
        {
            dtLst.Remove(trg);
        }
    }

    public List<FrgmntMdl> Srch(string txt)
    {
        List<FrgmntMdl> rs = new List<FrgmntMdl>();
        foreach (var x in dtLst)
        {
            if ((x.Nmn != null && x.Nmn.Contains(txt)) ||
                (x.Tgs != null && x.Tgs.Contains(txt)))
            {
                rs.Add(x);
```

```
            }
        }
        return rs;
    }
}

public class FrgmntSrv
{
    private readonly StrgSrv strg;

    public FrgmntSrv(StrgSrv s)
    {
        strg = s;
    }

    public List<FrgmntMdl> LdAll()
    {
        List<FrgmntMdl> rs = strg.GtAll();
        List<FrgmntMdl> fnl = new List<FrgmntMdl>();
        foreach (var x in rs)
        {
            fnl.Add(x);
        }
        return fnl;
    }

    public FrgmntMdl LdOne(int dntf)
    {
        FrgmntMdl rs = strg.GtByDntf(dntf);
        if (rs != null)
        {
            FrgmntMdl cp = new FrgmntMdl();
            cp.Dntf = rs.Dntf;
            cp.Nmn = rs.Nmn;
            cp.Cntnt = rs.Cntnt;
            cp.Tgs = rs.Tgs;
            cp.Crtd = rs.Crtd;
            cp.Mdfd = rs.Mdfd;
            return cp;
        }
        return null;
    }

    public void Crt(string nmn, string cntnt, string tgs)
    {
        if (nmn != null && cntnt != null)
        {
            strg.AddNw(nmn, cntnt, tgs);
        }
```

```
    }

    public void Mdf(int dntf, string nmn, string cntnt, string
tgs)
    {
        if (dntf > 0)
        {
            strg.Updt(dntf, nmn, cntnt, tgs);
        }
    }

    public void Dlt(int dntf)
    {
        if (dntf > 0)
        {
            strg.Rmv(dntf);
        }
    }

    public List<FrgmntMdl> Fnd(string txt)
    {
        List<FrgmntMdl> rs = new List<FrgmntMdl>();
        if (txt != null)
        {
            var tmp = strg.Srch(txt);
            foreach (var x in tmp)
            {
                rs.Add(x);
            }
        }
        return rs;
    }
}

public class Cntrl : Controller
{
    private readonly FrgmntSrv srv;

    public Cntrl(FrgmntSrv s)
    {
        srv = s;
    }

    public IActionResult Indx()
    {
        List<FrgmntMdl> lst = srv.LdAll();
        return View(lst);
    }
```

```
    public IActionResult Vw(int dntf)
    {
        FrgmntMdl mdl = srv.LdOne(dntf);
        return View(mdl);
    }

    public IActionResult Nw()
    {
        return View();
    }

    [HttpPost]
    public IActionResult NwSbmt(string nmn, string cntnt, string
tgs)
    {
        srv.Crt(nmn, cntnt, tgs);
        return RedirectToAction("Indx");
    }

    public IActionResult Edt(int dntf)
    {
        FrgmntMdl mdl = srv.LdOne(dntf);
        return View(mdl);
    }

    [HttpPost]
    public IActionResult EdtSbmt(int dntf, string nmn, string
cntnt, string tgs)
    {
        srv.Mdf(dntf, nmn, cntnt, tgs);
        return RedirectToAction("Indx");
    }

    public IActionResult Dlt(int dntf)
    {
        srv.Dlt(dntf);
        return RedirectToAction("Indx");
    }

    public IActionResult Srch(string txt)
    {
        List<FrgmntMdl> rs = srv.Fnd(txt);
        return View("Indx", rs);
    }
}

public bool ChckNmnExst(string nmn)
{
    bool rs = false;
```

```
    foreach (var x in dtLst)
    {
        if (x.Nmn == nmn)
        {
            rs = true;
        }
    }
    return rs;
}

public int GtTtlCnt()
{
    int cnt = 0;
    foreach (var x in dtLst)
    {
        cnt = cnt + 1;
    }
    return cnt;
}

public List<string> GtAllTgs()
{
    List<string> rs = new List<string>();
    foreach (var x in dtLst)
    {
        if (x.Tgs != null)
        {
            var splt = x.Tgs.Split(',');
            foreach (var tg in splt)
            {
                if (!rs.Contains(tg))
                {
                    rs.Add(tg);
                }
            }
        }
    }
    return rs;
}

public bool VldDt(string nmn, string cntnt)
{
    bool rs = true;
    if (string.IsNullOrWhiteSpace(nmn))
    {
        rs = false;
    }
    if (string.IsNullOrWhiteSpace(cntnt))
    {
```

```
        rs = false;
    }
    return rs;
}

public List<FrgmntMdl> FltrByTg(string tg)
{
    List<FrgmntMdl> rs = new List<FrgmntMdl>();
    var all = strg.GtAll();
    foreach (var x in all)
    {
        if (x.Tgs != null && x.Tgs.Contains(tg))
        {
            rs.Add(x);
        }
    }
    return rs;
}

public Dictionary<string, int> GtTgStts()
{
    Dictionary<string, int> rs = new Dictionary<string, int>();
    var all = strg.GtAll();
    foreach (var x in all)
    {
        if (x.Tgs != null)
        {
            var splt = x.Tgs.Split(',');
            foreach (var tg in splt)
            {
                if (!rs.ContainsKey(tg))
                {
                    rs[tg] = 1;
                }
                else
                {
                    rs[tg] = rs[tg] + 1;
                }
            }
        }
    }
    return rs;
}

public void LgActn(string usr, string actn, string trg)
{
    string ln = "";
    ln = ln + usr;
    ln = ln + "|";
```

```
    ln = ln + actn;
    ln = ln + "|";
    ln = ln + trg;
    ln = ln + "|";
    ln = ln + DateTime.Now.ToString();
    lgLst.Add(ln);
}

public List<string> GtLg()
{
    List<string> rs = new List<string>();
    foreach (var x in lgLst)
    {
        rs.Add(x);
    }
    return rs;
}

public List<string> FltrLg(string usr)
{
    List<string> rs = new List<string>();
    foreach (var x in lgLst)
    {
        if (x.StartsWith(usr + "|"))
        {
            rs.Add(x);
        }
    }
    return rs;
}

public void CrtVrsn(int frgId, string cntnt)
{
    if (!vrsnMp.ContainsKey(frgId))
    {
        vrsnMp[frgId] = new List<string>();
    }
    vrsnMp[frgId].Add(cntnt);
}

public List<string> GtVrsns(int frgId)
{
    if (vrsnMp.ContainsKey(frgId))
    {
        return vrsnMp[frgId];
    }
    return new List<string>();
}
```

```
public string GtLstVrsn(int frgId)
{
    if (vrsnMp.ContainsKey(frgId))
    {
        var lst = vrsnMp[frgId];
        if (lst.Count > 0)
        {
            return lst[lst.Count - 1];
        }
    }
    return "";
}

public List<FrgmntMdl> MprtTxt(string txt)
{
    List<FrgmntMdl> rs = new List<FrgmntMdl>();
    var lns = txt.Split('\n');
    foreach (var ln in lns)
    {
        var prts = ln.Split('|');
        if (prts.Length >= 3)
        {
            FrgmntMdl m = new FrgmntMdl();
            m.Nmn = prts[0];
            m.Cntnt = prts[1];
            m.Tgs = prts[2];
            rs.Add(m);
        }
    }
    return rs;
}

public void MprtTStrg(string txt)
{
    var lst = MprtTxt(txt);
    foreach (var x in lst)
    {
        strg.Add(x);
    }
}
```

ДОДАТОК В

Слайди презентації

Національний університет "Запорізька політехніка"
Кафедра програмних засобів

ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ ДЛЯ УПРАВЛІННЯ БІБЛІОТЕКОЮ ПРОГРАМНИХ ФРАГМЕНТІВ

Студент: Михайло МІКУЛІН, гр. КНТ-112
Керівник: к.т.н., доцент Михайло КОЦУР

Рисунок В.1 – Слайд 1

Об'єкт, предмет та мета роботи

Об'єкт дослідження – процес розробки програмного забезпечення для управління бібліотекою програмних фрагментів.

Предмет дослідження – програмні засоби для управління бібліотекою програмних фрагментів.

Мета роботи – розробка програмного забезпечення для управління бібліотекою програмних фрагментів, що забезпечує зменшення часу пошуку та повторного написання коду.

Рисунок В.2 – Слайд 2

Завдання роботи

Для досягнення поставленої мети у кваліфікаційній роботі бакалавра необхідно розв'язати такі задачі:
— виконати аналіз предметної області та програмних засобів для управління бібліотекою програмних фрагментів;
— здійснити проєктування програмного забезпечення для управління бібліотекою програмних фрагментів;
— створити програмне забезпечення для управління бібліотекою програмних фрагментів;
— виконати тестування розробленого програмного забезпечення для управління бібліотекою програмних фрагментів.

Рисунок В.3 – Слайд 3

Порівняння існуючих аналогів

Критерій порівняння	Masscode	Snipit	CodePen
підтримка різних мов програмування	+	+	+–
веборієнтований доступ	–	+	+
можливості структурування бібліотеки фрагментів коду	+	+	+–
підтримка командної роботи	+–	+–	+
інтеграція з процесом розробки	+	+–	+–
масштабованість бібліотеки	+	+–	+–

Рисунок В.4 – Слайд 4

Вибір мови програмування

Критерій порівняння мов програмування	Мова програмування		
	C#	Java	Python
Простота розробки	+	+–	+
Інтеграція з базами даних	+	+	+–
Підтримка графічного інтерфейсу	+	+–	+–
Масштабованість та модульність	+	+	+–
Безпека та контроль помилок	+	+	+–

Рисунок В.5 – Слайд 5

Вибір середовища розробки

Критерій порівняння середовищ розробки	Середовища розробки		
	Visual Studio	MonoDevelop	SharpDevelop
Потужні інструменти налагодження	+	+–	–
Засоби побудови графічного інтерфейсу	+	+–	–
Інструменти тестування та профілювання	+	+–	+–
Інтеграція з різними системами контролю версій	+	+–	+–
Підтримка розширень і плагінів	+	+–	–

Рисунок В.6 – Слайд 6

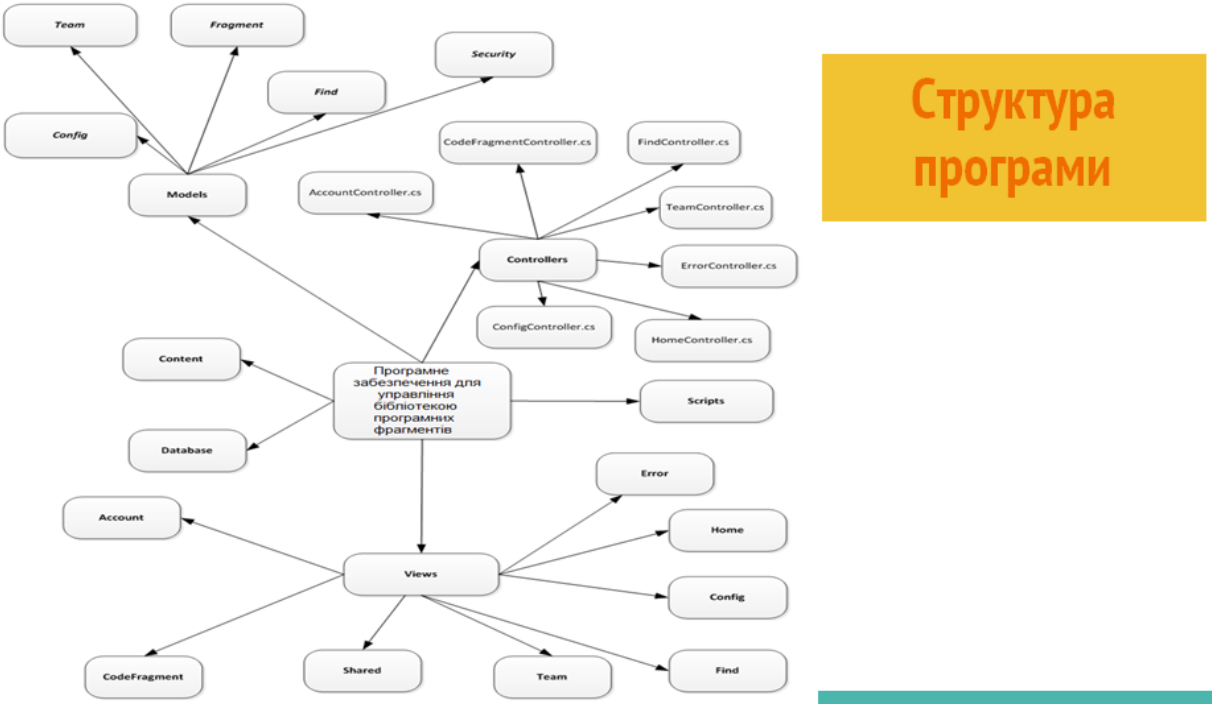

Рисунок В.7 – Слайд 7

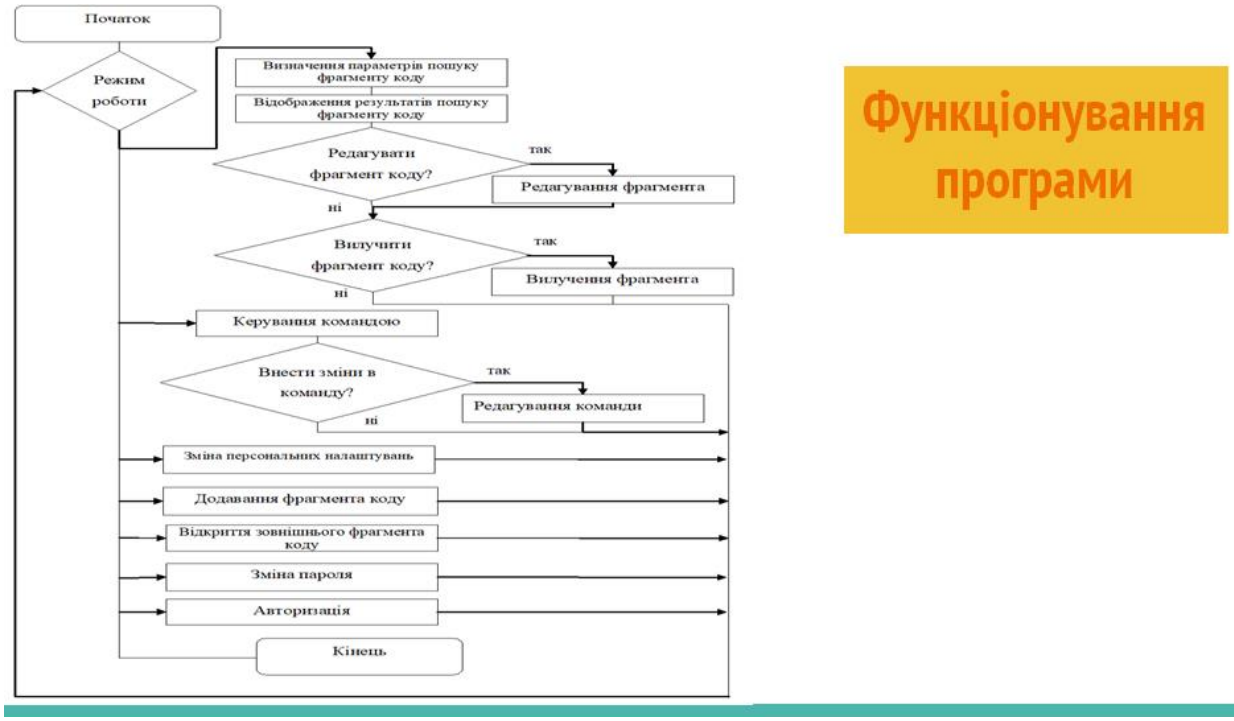

Рисунок В.8 – Слайд 8

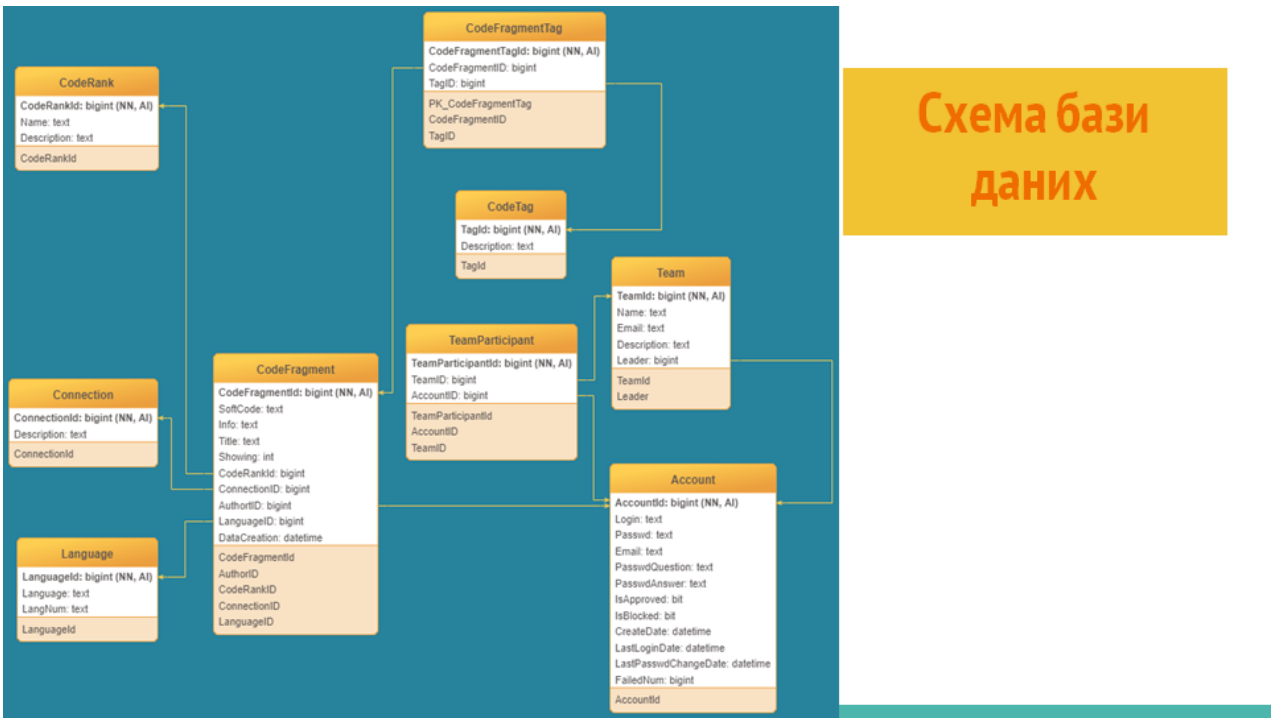

Рисунок В.9 – Слайд 9

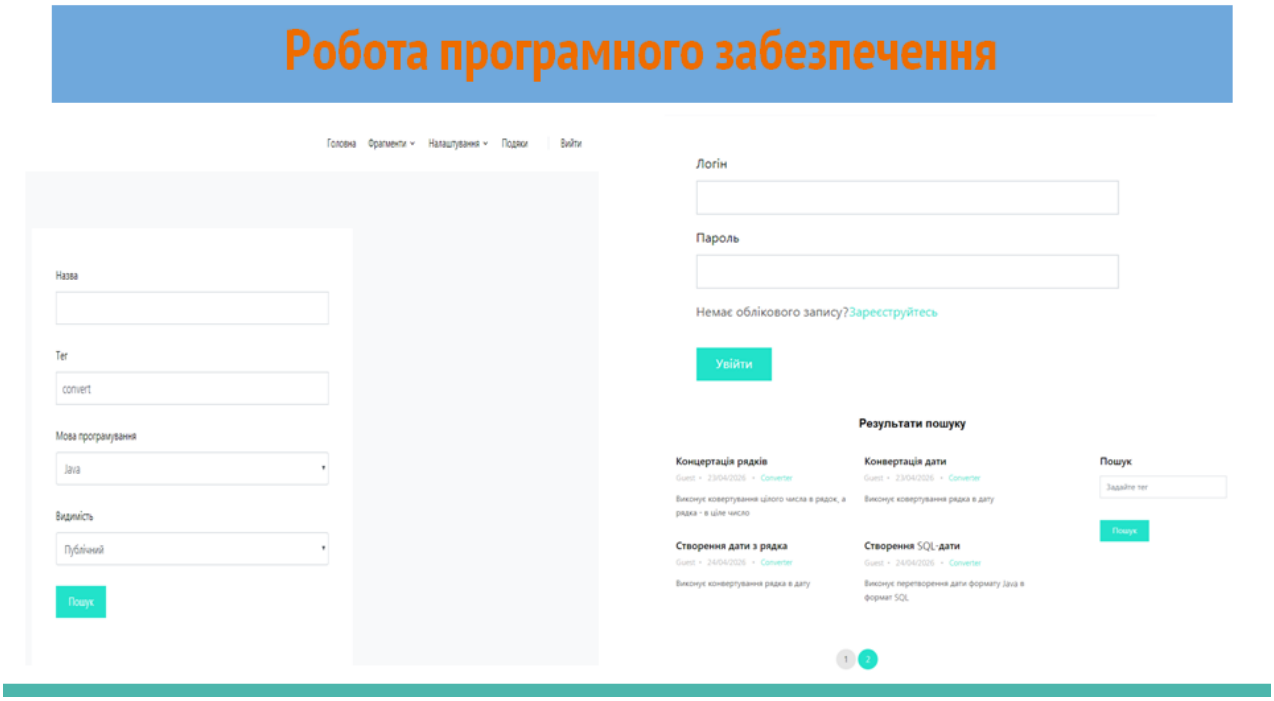

Рисунок В.10 – Слайд 10

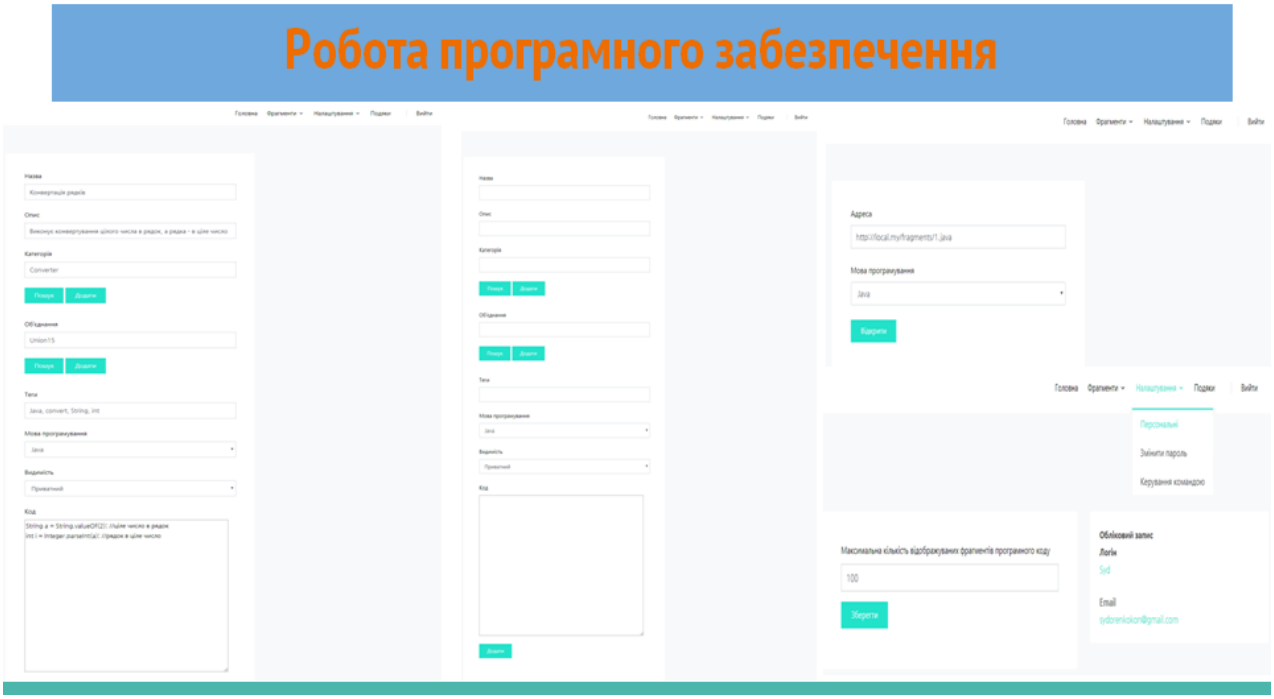

Рисунок В.11 – Слайд 11

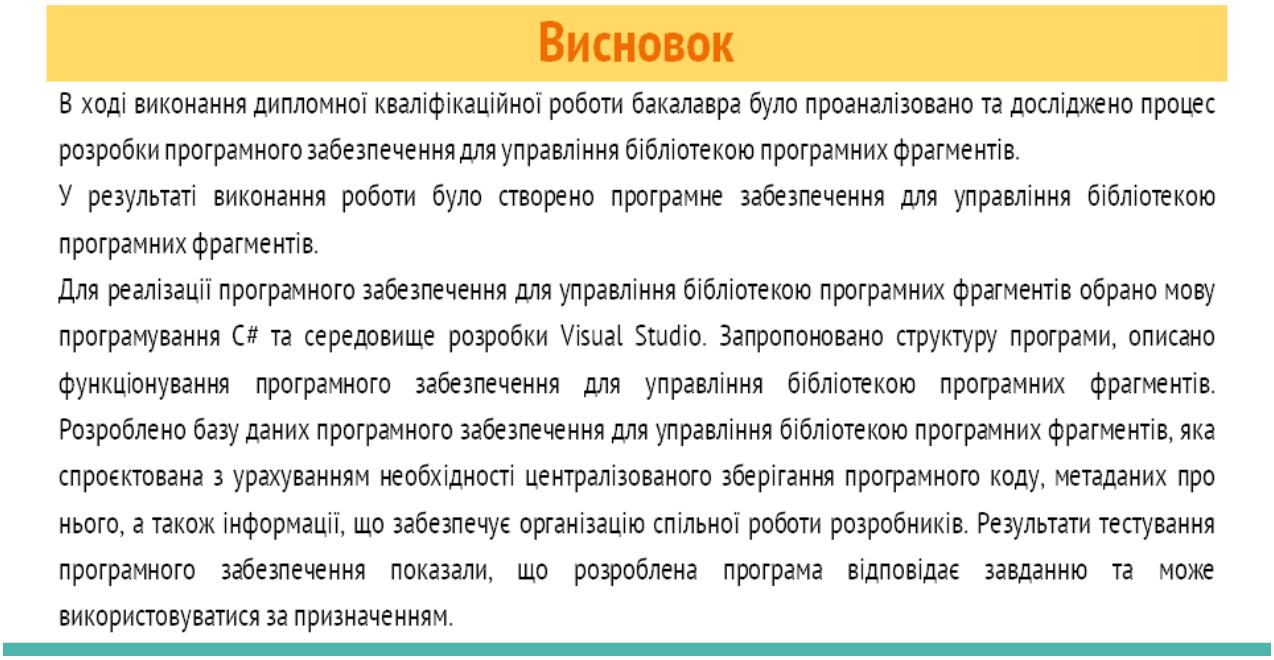

Рисунок В.12 – Слайд 12