

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»

Інститут інформатики та радіoeлектроніки,
Факультет комп'ютерних наук і технологій
(повне найменування інституту, назва факультету)

Кафедра програмних засобів
(повне найменування кафедри)

Пояснювальна записка

до дипломного проєкту (роботи)

бакалавра

(ступінь вищої освіти)

на тему ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ КЕРУВАННЯ РЕПОЗИТОРІЄМ
ФРАГМЕНТІВ ПРОГРАМНОГО КОДУ
CODE SNIPPETS REPOSITORY SOFTWARE

Виконав: студент(ка) 4 курсу, групи КНТз-118сп
Спеціальності 121 Інженерія програмного
забезпечення

(код і найменування спеціальності)

Освітня програма (спеціалізація)

Інженерія програмного забезпечення

Гранкін А.О.

(прізвище та ініціали)

Керівник Скрупський С.Ю.

(прізвище та ініціали)

Рецензент Зеленьова І.Я.

(прізвище та ініціали)

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»
(повне найменування закладу вищої освіти)

Інститут, факультет ІРЕ, ФКНТ
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.
С.О. Субботін
“ ” 2021 року

З А В Д А Н Н Я

НА ДИПЛОМНИЙ ПРОЄКТ (РОБОТУ) СТУДЕНТА(КИ)

Гранкіна Андрія Олеговича

(прізвище, ім'я, по батькові)

1. Тема проєкту (роботи) Програмне забезпечення керування репозиторієм
фрагментів програмного коду
Code Snippets Repository Software

керівник проєкту (роботи) Скрупський Степан Юрійович, к.т.н., доцент,
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом закладу вищої освіти від “30” березня 2021 року № 103

2. Строк подання студентом проєкту (роботи) 12 травня 2021 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. Розробка архітектури програми.
3. Розробка програми. 4. Експлуатація та тестування програми.

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень)
Слайди презентації

6. Консультанти розділів проєкту (роботи)

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	прийняв виконане завдання
1-4 Основна частина	Скрупський С.Ю., доцент		
Нормоконтролер	Андрєєв М.О., асистент		

7. Дата видачі завдання “9” березня 2021 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	2-3 тижні	Розділ 1
3	Розробка архітектури програми	4 тиждень	Розділ 2
4	Розробка програми	5-6 тижні	Розділ 3
5	Тестування та експериментальне дослідження програми	7 тиждень	Розділ 4
6	Оформлення пояснювальної записки та документів до неї. Нормоконтроль та рецензування	8 тиждень	Додатки
7	Захист роботи	9 тиждень	

Студент(ка)

(підпис)

Гранкін А.О.

(прізвище та ініціали)

Керівник проєкту (роботи)

(підпис)

Скрупський С.Ю.

(прізвище та ініціали)

РЕФЕРАТ

Пояснювальна записка до дипломної кваліфікаційної роботи бакалавра: 82 с., 30 рис., 2 табл., 3 дод., 15 джерел.

РЕПОЗИТОРІЙ, ФРАГМЕНТ ПРОГРАМНОГО КОДУ, ПОВТОРНЕ ВИКОРИСТАННЯ КОДУ, РОБОТА В КОМАНДІ, ВЕБЗАСТОСУНОК.

Об'єкт роботи – процес накопичення фрагментів програмного коду.

Предмет роботи – програмне забезпечення керування репозиторієм фрагментів програмного коду.

Мета роботи – розробити програмне забезпечення керування репозиторієм фрагментів програмного коду для підвищення зручності накопичення і повторного використання існуючого програмного коду.

Матеріали, методи та технічні засоби: об'єктно-орієнтоване програмування, мова програмування C#, фреймворк ASP.NET, архітектурний шаблон Model-View-Controller.

Результати. Результати роботи охоплюють результати аналізу предметної області, проєктування і реалізації програми та безпосередньо реалізовану програму.

Висновки. Вебзастосунок, який було створено у результаті виконання роботи, можна застосовувати для накопичення невеликих фрагментів програмного коду, створеного різними мовами програмування, редагування цих фрагментів як під час індивідуальної роботи, так і під час роботи в команді розробників, використання фрагментів для створення нових проєктів, відслідковування зацікавлення інших користувачів власними фрагментами програмного коду.

Галузь використання – розробка програмного забезпечення на основі повторного використання існуючих фрагментів коду, у тому числі в команді, створення репозиторію фрагментів програмного коду.

ABSTRACT

Explanatory note to the diploma qualifying work of the bachelor: 82 pages, 30 figures, 2 tables, 3 appendixes, 15 sources.

REPOSITORY, CODE SNIPPET, CODE REUSAGE, TEAMWORK, WEB APPLICATION.

The object of the work is the process of accumulating code snippets.

The subject of the work is the software for managing the repository of code snippets.

The purpose of the work is to develop software for managing repository of code snippets to increase the convenience of accumulation and reuse of existing program code.

Materials, methods and technical means: object-oriented programming, C# programming language, ASP.NET framework, Model-View-Controller architectural template.

Results. The results of the work include the results of the analysis of the subject area, design and implementation of the program and the implemented software.

Conclusions. The resulting web application can be used to accumulate small snippets of code created using different programming languages, edit these snippets both individually and in a team, use snippets to create new projects, and track interest of other users in their own code snippets.

Field of usage includes software development based on the reuse of existing code snippets, including development in a team, the creation of a repository of code snippets.

ЗМІСТ

	С.
Перелік скорочень та умовних познач.....	7
Вступ.....	8
1 Аналіз предметної області.....	9
1.1 Поняття фрагмента програмного коду.....	9
1.2 Ліцензії програмного забезпечення.....	9
1.3 Огляд програмних аналогів.....	11
1.4 Висновки за розділом 1	18
2 Розробка архітектури програми.....	19
2.1 Вимоги до програмного забезпечення	19
2.2 Вибір і використання засобів проєктування і розробки.....	23
2.3 Проєктування структури даних	24
2.4 Висновки за розділом 2	29
3 Розробка програми	31
3.1 Послідовність виконання застосунку.....	31
3.2 Розроблені структурні елементи програми	33
3.3 Висновки за розділом 3	40
4 Експлуатація та тестування програми	41
4.1 Призначення програми	41
4.2 Умови виконання програми	41
4.3 Робота з програмою	41
Висновки	51
Перелік джерел посилання	52
Додаток А Технічне завдання	54
Додаток Б Текст програми	59
Додаток В Слайди презентації.....	75

ПЕРЕЛІК СКОРОЧЕНЬ ТА УМОВНИХ ПОЗНАК

BSD	– Berkeley Software Distribution;
GPL	– General Public License;
LGPL	– Lesser General Public License;
MIT	– Massachusetts Institute of Technology;
MVC	– Model-View-Controller.

ВСТУП

Програмна реалізація давно перестала складатися з постійного написання власного коду, велика частина даного процесу в сучасний момент займає застосування існуючого коду та додавання до нього коду, який дозволить якісно і ефективно розв'язувати всю задачу загалом.

До коду, який може повторно використовуватися, відносяться програмні бібліотеки, компоненти, а також фрагменти програмного коду. Останні – це менший за обсягом код, який забезпечує уникнення рутинної роботи програмістів. Такий код часто повторюється, недостатньо великий для винесення його у окрему бібліотеку або не поєднаний з іншими засобами таким чином, щоб їх разом можна було виділити в окрему бібліотеку. З іншого боку, якщо кожен раз писати такий код заново, то це підвищує кількість можливих помилок і вимагає постійного часу на відлагодження. Саме на вирішення цієї проблеми спрямоване створення репозиторію фрагментів програмного коду. І саме процес накопичення фрагментів програмного коду визначено в результаті в якості предмету об'єкту роботи.

На даний момент проблему повністю не розв'язано – не створено програмне забезпечення, яке би повністю вирішувало всі існуючі проблеми в цій області, тому розробку, що виконується в роботі, можна вважати актуальною.

Предмет роботи – програмне забезпечення керування репозиторієм фрагментів програмного коду.

Мета роботи – розробити програмне забезпечення керування репозиторієм фрагментів програмного коду для підвищення зручності накопичення і повторного використання існуючого програмного коду.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Поняття фрагмента програмного коду

Фрагмент програмного коду – це термін, що використовується для опису невеликої частини повторно використовованого вихідного коду, машинного коду або тексту, вони дозволяють програмісту уникати введення повторюваного коду під час рутинного програмування, корисні в наступному:

- рятують від необхідності виконання набору коду, який можна скопіювати;
- полегшують запам'ятовування певного форматування функцій, так деякі функції можуть бути надзвичайно складними або можуть містити важко запам'ятовуваний синтаксис, замість того, щоб шукати помилки друку та синтаксичні помилки, варто використати готовий фрагмент;
- фрагменти доступні для спільного використання: при роботі в команді, яка працює над проектом, фрагмент коду може допомогти гарантувати, що всі учасники пишуть певні блоки коду однаково, у результаті код, створений всією командою, більш послідовний і читабельний [1].

1.2 Ліцензії програмного забезпечення

Як власники авторських прав, розробники мають виключне право контролювати використання та розповсюдження програм, які вони створюють, тому ліцензія на програмне забезпечення – це юридично зобов'язуюча угода між постачальником програмного забезпечення та іншою стороною, яка бажає використовувати ці програмні продукти, що встановлює правила, вимоги та керівні принципи для обох сторін, особливо стосовно того, як ліцензіат може використовувати програмне забезпечення, як

визначатимуться витрати на використання та міру, в якій ліцензіат може копіювати, модифікувати або поширювати програмне забезпечення [2]-[3].

Усі ліцензії програмного забезпечення можна поділити на наступні групи:

- відкритий доступ – це найбільш дозвоільний тип ліцензії на програмне забезпечення: коли програмне забезпечення знаходиться у відкритому доступі, кожен може змінювати та використовувати програмне забезпечення без будь-яких обмежень, при цьому при використанні потрібно спочатку переконатися, що це безпечно, варто також враховувати, що код, який не має явної ліцензії, не автоматично перебуває у відкритому доступі;

- дозвоільні: дозвоільні ліцензії також відомі як “стиль Apache” або “стиль Berkeley Software Distribution (BSD)”, вони містять мінімальні вимоги щодо того, як програмне забезпечення може бути змінено або розповсюджене, цей тип ліцензії на програмне забезпечення є загалом найпопулярнішою ліцензією, що використовується з безкоштовним програмним забезпеченням із відкритим кодом, окрім ліцензії Apache та ліцензії BSD, ще одним поширеним варіантом є ліцензія Massachusetts Institute of Technology (MIT);

- Lesser General Public License (LGPL) – загальнодоступна ліцензія, що дозволяє посилатися на бібліотеки з відкритим кодом у програмному забезпеченні: якщо просто скопіювати або зв’язати бібліотеку, що має ліцензію LGPL, із власним кодом, можна випустити свою програму за будь-якою ліцензією, якою потрібно, навіть за власною ліцензією, але якщо буде змінено бібліотеку або скопійовано її частини у свій код, то доведеться випустити програму на таких самих умовах, як LGPL;

- ліцензії Copyleft, також відомі як взаємні ліцензії або обмежувальні ліцензії: найбільш відомим прикладом Copyleft або взаємної ліцензії є General Public License (GPL), ці ліцензії дозволяють змінювати ліцензійний код та розповсюджувати нові розробки на їх основі, якщо розповсюджуються будь-які нові розробки або адаптації відповідно до тієї самої ліцензії на

програмне забезпечення, наприклад, у ліцензії компонента може бути зазначено, що твір можна використовувати та розповсюджувати лише для особистого користування, отже будь-які похідні розробки також обмежуються лише особистим використанням;

– пропріетарні: з усіх типів ліцензій на програмне забезпечення дана є найбільш обмежувальною, ідея полягає в тому, що всі права захищені, зазвичай вона використовується для власного програмного забезпечення, де код не може бути змінений або перерозповсюджений [4]-[7].

1.3 Огляд програмних аналогів

На даний момент у багатьох сучасних середовищах програмування існують засоби створення фрагментів програмного коду, а також можливо їх використання. Але такі засоби не дозволяють створити репозиторій фрагментів програмного коду для повторного використання.

Вебсайт Snipplr [8] (рис. 1.1) надає засоби роботи з репозиторієм фрагментів програмного коду.

Snipplr забезпечує створення власних фрагментів, щодо яких вносяться:

- назва фрагмента;
- опис фрагмента;
- код фрагмента;
- пов'язані з ним фрагменти програмного коду;
- дата створення фрагмента;
- власник;
- теги;
- мова програмування, якою створено фрагмент;
- коментарі до фрагмента;
- відкритість фрагмента.

SNIPPLR

CODE 2.0

CODE SNIPPETS

BLOG

PRIVACY

ABOUT SNIPPLR

all code snippets / popular code snippets / your code snippets

[New Snippet](#) / [Register](#) / [Login](#)

Browser Detection

SAV

/ Published in: JavaScript

Detects the browser and its version.

Expand | Embed | Plain Text

```

1. // Browser detection
2.
3. // Internet Explorer
4. var ie = document.all != null; //ie4 and above
5. var ie5 = document.getElementById && document.all;
6. var ie6 = document.getElementById && document.all&&(navigator.appVersion.indexOf("MSIE 6.")>=0);
7.
8. // Netscape
9. var ns4 = document.layers != null;
10. var ns6 = document.getElementById && !document.all;
11. var ns = ns4 || ns6;
12.
13. // Firefox
14. var ff = !document.layers && !document.all;
15.
16. // Opera
17. var op = navigator.userAgent.indexOf("opera")>0;
18. var op7 = op && operaVersion() <= 7;
19. var op8 = op && operaVersion() >= 8;
20.
21. // Detects the Opera version
22. function operaVersion() {
23.     agent = navigator.userAgent;
24.     idx = agent.indexOf("opera");
25.     if (idx>-1) {
26.         return parseInt(agent.subString(idx+6,idx+7));
27.     }
28. }
```

Comments

Posted By

MyKey_ on 07/08/06

Tagged

ie browser

Versions (?)

Last Edited at 07/08/06 07:28am

Statistics

Viewed 1825 times

Favorited by 99 user(s)

Related snippets

289

Konqueror Browser OR KHTML Engine Detection

312

Browser Detection

307

deprecated jQuery browser detection

362

ASP.net & MS AJAX: Enabling WebKit support (chrome, safari) in MS AJAX (fixing webkit browser detection)

348

Wordpress Browser Detection Function

366

js javascript browser language detection, get the user's browser's language preference setting using jquery to detect users br

301

PHP browser detection light

Рисунок 1.1 – Вебсайт Snipplr

Щодо перегляду фрагментів надаються наступні засоби:

- пошук фрагментів за назвою або описом;
- перегляд всіх фрагментів програмного коду;
- перегляд популярних фрагментів програмного коду;
- перегляд власних фрагментів програмного коду;
- перегляд фрагментів, створених заданою мовою програмування;
- перегляд пов'язаних з даним фрагментів;
- перегляд фрагментів за тегом;
- перегляд найактивніших користувачів.

Авторизовані користувачі мають можливість створювати власні фрагменти програмного коду (рис. 1.2), додавати коментарі до існуючих фрагментів, ставити уподобання фрагментам, переглядати уподобані фрагменти.

SNIPPLR CODE 2.0

CODE SNIPPETS BLOG PRIVACY ABOUT SNIPPLR

all code snippets / popular code snippets / your folders / your code snippets / favorites

New Snippet / Dashboard / Settings / Profile / Logout [vlovkin]

Search by title, description Advanced
Search

New Snippet

Title

Language
ActionScript
Don't See Your Language?

Tags

URL

Comments/Description: (Markdown [Enabled](#))

Source

☐ Make this snippet temporary (?)
☐ Make this snippet private (?)

Рисунок 1.2 – Створення власного фрагмента програмного коду в Snipplr

Варто відзначити, що для кожного фрагмента надається можливість перегляду його версій.

Даний програмний засіб можна вважати достатньо актуальним у розглядаємій предметній області, але при цьому необхідно відзначити, що у ньому відсутні засоби роботи в команді. Тобто команда розробників не може працювати над фрагментом, зокрема змінювати його. Це обмежує роботу, адже зазвичай використання фрагментів у реальному середовищі є командним, відповідно це впливає і на процес подальшого редагування фрагментів.

Djangosnippets [9] (рис. 1.3) також надає можливість роботи з фрагментами програмного коду, в тому числі і власними, але всі ці фрагменти програмного коду мають стосуватися фреймворку Django і охоплювати мови HTML, JavaScript, Python, Shell, SQL.

"Magic Link" Management Command

Author: webology Posted: March 24, 2021 Language: Python Version: Not specified Tags: authentication magic management command Score: 2 (after 2 ratings)

```

Django Management Command to print a "Magic Link" for one-click log-in. This is nice for people who project switch or don't want to
remember passwords.

1  """
2  Django Management Command to print a "Magic Link" for one-click login
3  by Jeff Triplett <jefftriplett> / https://twitter.com/webology
4
5  https://gist.github.com/jefftriplett/e87b36d750f94e48080a2be46e71dbe1
6
7  Relies on: https://github.com/aaugustin/django-sesame
8
9  $ pip install django-click
10 $ pip install django-sesame[ua]
11
12 This should live in one of your app that is installed in your `settings.INSTALLED_APPS`
13 /management/commands/magic-link.py
14
15 Example:
16
17 $ python manage.py magic-link
18 http://localhost:8000/?sesame=AAAAZXgFCVouyZCceM
19
20 """
21 import dclick as click
22
23 from django.contrib.auth import get_user_model
24 from django.contrib.sites.models import Site
25 from sesame.utils import get_query_string
26
27
28 @click.command()
29 @click.option("username", "--user", default=None)
30 def command(username):
31     User = get_user_model()
32
33     if username:
34         try:
35             user = User.objects.get(username=username)
36         except User.DoesNotExist:
37             user = None
38     else:
39         user = User.objects.first()
40
41     domain_name = Site.objects.get_current()
42
43     if user:
44         click.echo(f"http://{domain_name}/{get_query_string(user)}")
45     else:
46         click.secho(f"No user or username was found", fg="red")

```

More like this

1. "Magic Link" Management Command by webology 1 month, 1 week ago
2. Closest ORM models to a latitude/longitude point by simonw 1 month, 1 week ago

Рисунок 1.3 – Вебсайт Djangosnippets

Djangosnippets надає засоби виконання наступних функціональних можливостей:

- створення власного фрагмента програмного коду (рис. 1.4);
- перегляду всіх фрагментів програмного коду;
- перегляду фрагментів програмного коду за однією з доступних мов програмування;
- перегляду фрагментів за тегом;
- перегляду найбільш оцінених фрагментів;
- пошуку фрагментів;
- перегляду подібних фрагментів на відкритий.

Кожен фрагмент програмного коду має наступні характеристики:

- назва фрагмента;
- опис фрагмента;
- код фрагмента;
- дата створення фрагмента;
- власник;
- теги;
- мова програмування, якою створено фрагмент;
- версія фрагмента;
- рейтинг;
- коментарі до фрагмента.

Даний програмний засіб також не має засобів роботи в команді. Окрім того усі фрагменти є відкритими для всіх користувачів, відсутнє також прив'язування до інших існуючих фрагментів, надається тільки можливість знаходження подібних фрагментів.

Ще одним недоліком даного рішення є те, що можна працювати тільки з обмеженим набором мов, при цьому для даних мов як раз характерні проблеми з безпекою під час внесення фрагментів, тобто SQL або HTML-код може бути потенційно небезпечним. Оскільки саме ці мови і підтримуються, то логічно, що це рішення ніяк не продиктовано безпекою.

Please remember that this site is not a pastebin or debugging tool, it is intended for finished, working pieces of code.

Version: 3.0

Save

Remember: by posting a snippet, you agree to let anyone make use of the code.

Bold and italic text:
*****bold** and *italic* text***

Code:

```
`code`
```

Blockquote:

Start each line with >

Unordered lists:
Start each item with ^

Ordered lists:

Start each item with a number and a period (e.g., 1. for the first item, 2. for the second, and so on)

Рисунок 1.4 – Створення фрагмента програмного коду в DjangoSnippets

Для виконання порівняння аналогів (табл. 1.1) було виділено наступні показники:

- створення власних фрагментів: можливість авторизації та внесення власних фрагментів, а не тільки роботи з існуючими;
- велика кількість засобів перегляду і пошуку фрагментів: потрібно для того, щоб можна було зручно знаходити те, що потрібно в роботі;
- створення закритих фрагментів: дозволяє використовувати програму для індивідуальної роботи, тобто створення приватного репозиторію;
- мови програмування: коло мов, фрагменти за якими можна створювати;
- засоби роботи в команді;
- статистика фрагментів у динаміці: перегляд даних про популярність фрагментів, наприклад, у щомісячному розрізі.

Таблиця 1.1 – Порівняння програмних аналогів

Показник	Snipplr	Djangosnippets
Створення власних фрагментів	+	+
Велика кількість засобів перегляду і пошуку фрагментів	+	+
Створення закритих фрагментів	+	–
Мови програмування	Усі широковідомі	HTML, JavaScript, Python, Shell, SQL
Засоби роботи в команді	–	–
Статистика фрагментів у динаміці	–	–

У результаті порівняння було визначено Snipplr у якості основного програмного аналогу, адже він дозволяє працювати з будь-якими відомими мовами програмування, тобто не є обмеженим, надає можливість створення приватних фрагментів.

1.4 Висновки за розділом 1

У результаті проведеного аналізу предметної області було визначено поняття фрагментів програмного коду, особливості їх використання, основні ліцензії програмного забезпечення, виконано огляд програмних аналогів, включаючи Snipplr,.djangosnippets, для них визначено їх функціональні можливості, переваги і недоліки та виконано порівняння зі створенням відповідної таблиці порівняння.

У якості основного аналогу для подальшої розробки було обрано Snipplr. Основною особливістю розроблюваного програмного забезпечення має стати надання можливості роботи в команді над фрагментами програмного коду.

Завданнями роботи є:

- виділення вимог до програми;
- проєктування програми;
- розробка з налагодженням програми;
- тестування і налагодження програми.

2 РОЗРОБКА АРХІТЕКТУРИ ПРОГРАМИ

2.1 Вимоги до програмного забезпечення

На основі виконаного аналізу предметної області було вирішено використати виділені функціональні вимоги в якості бази для програми. У результаті функціональні вимоги до програми (рис. 2.1) включають:

- створення фрагментів програмного коду;
- прив'язування фрагментів програмного коду до команд розробників;
- оновлення фрагментів програмного коду;
- визначення режиму перегляду фрагмента;
- перегляд власних фрагментів програмного коду;
- пошук фрагментів програмного коду;
- перегляд найпопулярніших фрагментів програмного коду;
- перегляд найбільш уподобаних фрагментів програмного коду;
- перегляд фрагмента програмного коду;
- перегляд фрагментів програмного коду команди;
- визначення команди розробників;
- змінювання учасників у команді розробників;
- перегляд команд, до складу яких входить розробник;
- перегляд фрагментів за мовою програмування;
- перегляд фрагментів за тегом;
- перегляд популярності мов програмування;
- перегляд статистики уподобань фрагмента;
- перегляд статистики переглядів фрагмента;
- перегляд статистики уподобань усіх фрагментів розробника;
- перегляд статистики переглядів усіх фрагментів розробника;
- перегляд усіх уподобаних фрагментів користувачем;
- перегляд усіх переглянутих раніше фрагментів користувачем;
- реєстрація розробників;
- авторизація розробників.

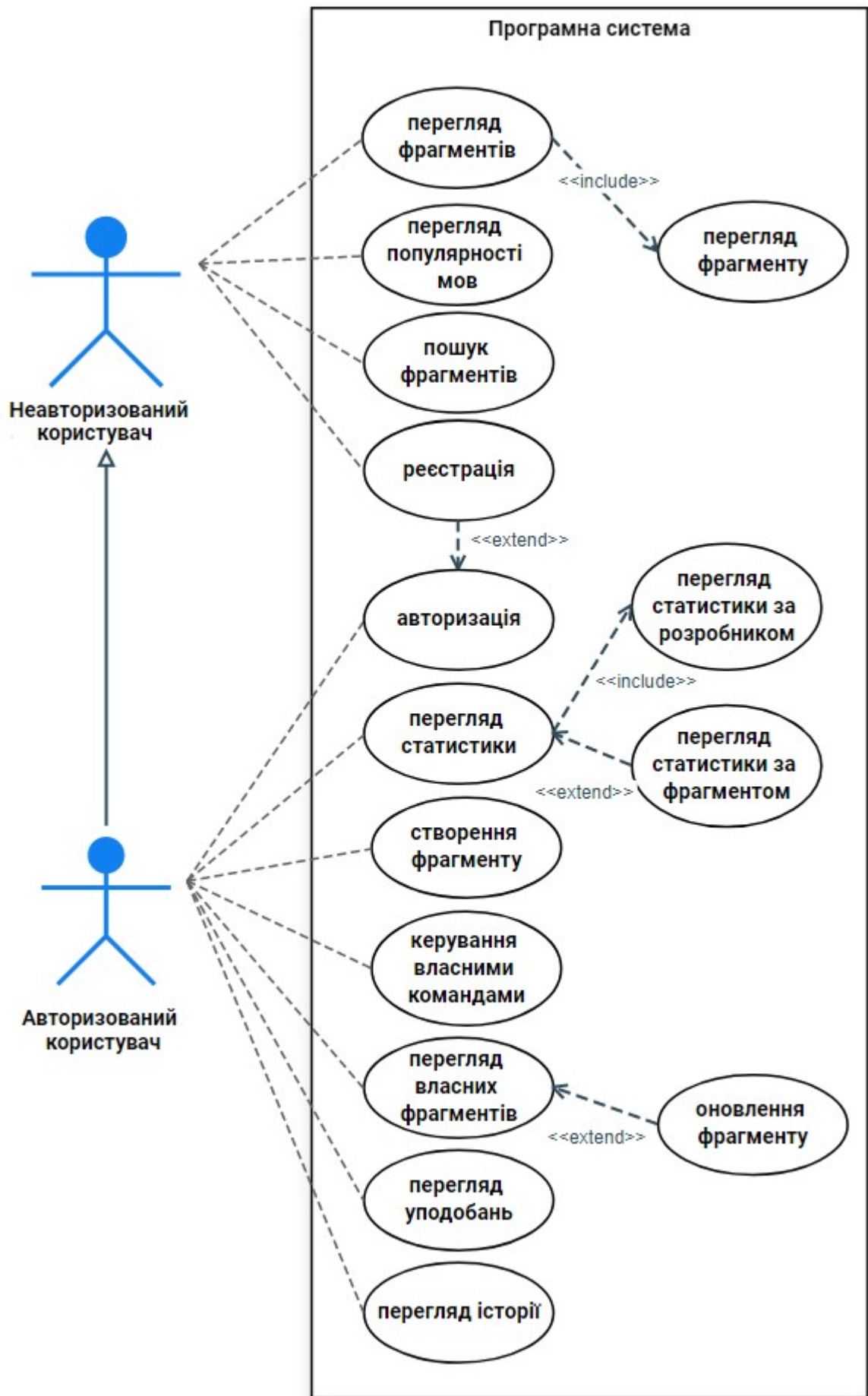


Рисунок 2.1 – Діаграма варіантів використання «Робота з системою»

Для даних функціональних вимог було виконано моделювання роботи з системою неавторизованого та авторизованого користувачів у вигляді діаграми варіантів використання (рис. 2.1).

Загалом неавторизований користувач може переглядати різного роду набори фрагментів програмного коду, переглядати конкретні фрагменти програмного коду, але не може вносити ніякі дані в базу даних та не має ніяких засобів, пов'язаних з роботою з власними фрагментами програмного коду.

Одним з основних варіантів використання програмної системи є перегляд фрагмента програмного коду (рис. 2.2), який охоплює наступні сценарії:

- перегляд інформації фрагмента програмного коду;
- перегляд коду фрагмента;
- перегляд історії оновлення фрагмента програмного коду;
- уподобання фрагмента програмного коду;
- перегляд коментарів.

Альтернативним сценарієм є прибирання уподобання фрагмента, оскільки даний сценарій може реалізуватися тільки в тому випадку, якщо попередньо фрагмент вже було уподобано раніше. Реалізується і уподобання, і прибирання уподобання тільки авторизованим користувачем.

Також альтернативним сценарієм є копіювання коду фрагмента в пам'ять, що реалізується під час перегляду коду фрагмента. Реалізується і перегляд коду, і копіювання коду і неавторизованим користувачем, і авторизованим.

Третім альтернативним сценарієм є створення коментаря до фрагмента програмного коду, як реалізується під час перегляду коментарів до фрагмента, але якщо перегляд може виконуватися і авторизованим, і неавторизованим користувачем, то створення коментаря може бути виконано тільки авторизованим користувачем.

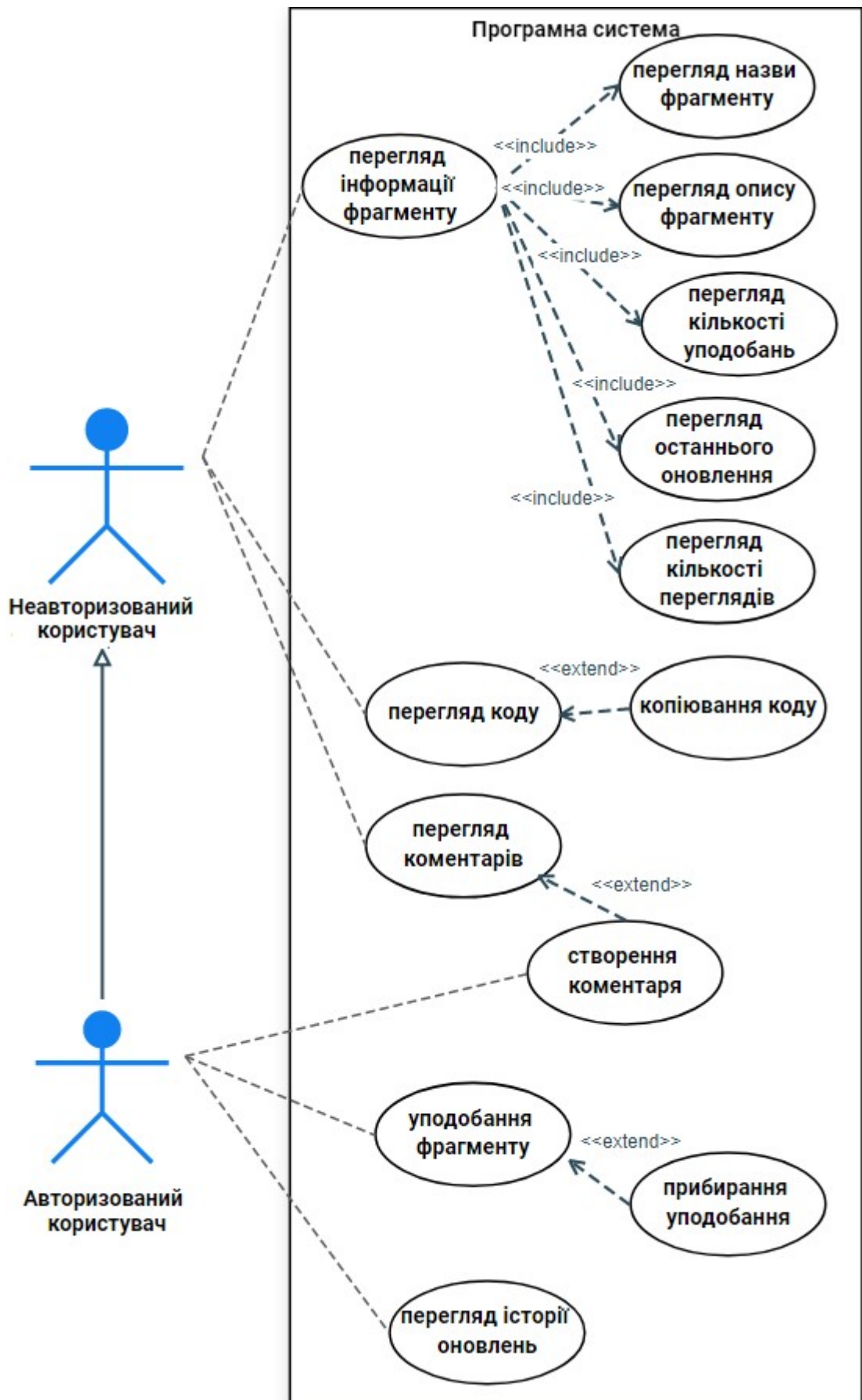


Рисунок 2.2 – Діаграма варіантів використання «Перегляд фрагмента програмного коду»

2.2 Вибір і використання засобів проєктування і розробки

Проведений аналіз програмних аналогів продемонстрував, що такі засоби логічно розробляти у вигляді вебзастосунку. Тому у даному підрозділі вирішено було порівняти (табл. 2.1) сучасні популярні фреймворки розробки вебзастосунків ASP.NET [10] мовою програмування C#, що і було обрано в результаті, та Laravel [11] мовою програмування PHP.

Таблиця 2.1 – Порівняння фреймворків веброзробки

Показник	ASP.NET	Laravel
Мова програмування	C#	PHP
Підтримка великих проєктів	++	+-
Безпека	++	+-
Підтримка спільноти	Якісна за рахунок Microsoft	Якісна і широка за рахунок вільної спільноти розробників

Показниками для виконання порівняння було визначено:

- мова програмування, якою будуть створюватися рішення: C# має більше засобів та можливостей ніж PHP;
- підтримка великих проєктів: ASP.NET [12] відомий за рахунок забезпечення підтримки великих проєктів, на відміну від Laravel, з розвитком проєкт в даній предметній області має стати масштабним;
- безпека: C# та ASP.NET надають більше засобів перевірки і забезпечення безпеки ніж Laravel;
- підтримка спільноти: Laravel має ширшу підтримку величезної спільноти розробників мовою PHP за рахунок того, що це вільне і активно використовуване рішення, проте ASP.NET є технологією Microsoft,

відповідно на стороні даного рішення досвід компанії і досвід розв’язання проблем, з якими можна стикнутися.

В якості архітектурного шаблону буде використовуватися традиційний саме для ASP.NET шаблон Model-View-Controller (MVC).

2.3 Проєктування структури даних

За результатами аналізу предметної області було виділено такі основні сутності предметної області:

- Codesnippet – фрагмент програмного коду, що є головною сутністю даної предметної області і зберігає безпосередньо код, з яким відбувається взаємодія в програмі;

- Comment – коментар до фрагмента коду користувача;

- View – перегляд фрагмента коду користувачем;

- Like – уподобання фрагмента коду користувачем;

- Tag – тег;

- Language – мова;

- License – ліцензія на використання і розповсюдження програмного коду;

- Developer – розробник;

- Team – команда розробників;

- SnippetUpdate – оновлення коду фрагмента користувачем, що необхідне для накопичення історії роботи над фрагментом.

Оскільки кожен тег може характеризувати різні фрагменти коду, а до того ж має бути дозволено визначати для фрагмента коду різні теги, в тому числі і не один, то потрібно було створити зв’язуючу таблицю TagSnippet, що визначає один тег для одного фрагмента.

Оскільки кожен розробник не обмежений і може входити в різні команди розробників, а кожна команда за своєю природою складається з

певної кількості розробників, то потрібно було створити зв'язуючу таблицю `DeveloperTeam`, що визначає участь розробника у команді розробників.

До таблиці `Codesnippet` входять наступні поля:

- `Id` типу `int` – номер фрагмента програмного коду (первинний ключ таблиці);
- `Name` типу `varchar` – назва фрагмента програмного коду;
- `Code` типу `text` – код фрагмента;
- `ShortDescription` типу `varchar` – короткий опис фрагмента, який може відображатися в переліку всіх фрагментів для його першого розуміння користувачем;
- `CreatorId` типу `int` – розробник, який створив запис про фрагмент коду, ж зовнішнім ключем з таблиці `Developer`;
- `Description` типу `text` – повний опис фрагмента програмного коду, який буде відображатися разом з фрагментом на розгорнутій сторінці;
- `CreationDate` типу `datetime` – дата створення фрагмента;
- `ModificationDate` типу `datetime` – дата останнього оновлення фрагмента програмного коду;
- `Mode` типу `tinyint` – режим перегляду фрагмента коду (доступний для всіх, тільки для команди розробників);
- `LicenseId` типу `int` – ліцензія, за якою розповсюджується фрагмент коду, є зовнішнім ключем до таблиці `License`;
- `LanguageId` типу `int` – мова, якою написано фрагмент програмного коду, є зовнішнім ключем таблиці `Language`;
- `TeamId` типу `int` – команда, яка працює над даним фрагментом.

До таблиці `Tag` входять наступні поля:

- `Id` типу `int` – номер тегу (первинний ключ таблиці);
- `Name` типу `varchar` – назва тегу.

До таблиці `Team` входять наступні поля:

- `Id` типу `int` – номер команди (первинний ключ таблиці);
- `Name` типу `varchar` – назва команди.

До таблиці License входять наступні поля:

- Id типу int – номер ліцензії (первинний ключ таблиці);
- Name типу varchar – назва ліцензії;
- Description типу text – опис ліцензії.

До таблиці Language входять наступні поля:

- Id типу int – номер мови програмування (первинний ключ таблиці);
- Name типу varchar – назва мови програмування;
- SnippetCount типу int – кількість фрагментів програмного коду,

створених цією мовою.

До таблиці Developer входять наступні поля:

- Id типу int – номер розробника;
- Login типу varchar – логін розробника;
- Pass типу varchar – пароль розробника;
- Email типу varchar – електронна пошта розробника;
- ActionDate типу datetime – останнє виконання авторизації

розробником;

- CreationDate типу datetime – дата створення облікового запису;
- Photo типу varchar – фотографія розробника;
- Description типу varchar – опис розробника (може включати зокрема

компанію, посаду в ній).

До таблиці Comment входять наступні поля:

- Id типу int – номер коментаря (первинний ключ таблиці);
- CodesnippetId типу int – фрагмент програмного коду, до якого

відноситься коментар (зовнішній ключ з таблиці Codesnippet);

- DeveloperId типу int – розробник, який створив коментар, зовнішній

ключ з таблиці Developer;

- Text типу text – текст коментаря;
- CreationDate типу datetime – дата створення коментаря.

До таблиці Like входять наступні поля:

- Id типу int – номер уподобання (первинний ключ таблиці);

- `CodesnippetId` типу `int` – фрагмент програмного коду, який уподобав розробник (зовнішній ключ з таблиці `Codesnippet`);

- `DeveloperId` типу `int` – розробник, який поставив уподобання (зовнішній ключ таблиці `Developer`);

- `Mode` типу `bit` – активність або неактивність уподобання;

- `CreationDate` `datetime` – дата створення уподобання.

До таблиці `SView` входять наступні поля:

- `Id` типу `int` – номер перегляду фрагмента програмного коду (первинний ключ таблиці);

- `CodesnippetId` типу `int` – фрагмент програмного коду, який було переглянуто (зовнішній ключ з таблиці `Codesnippet`);

- `DeveloperId` – розробник, який переглянув фрагмент (зовнішній ключ таблиці `Developer`);

- `CreationDate` типу `datetime` – дата перегляду.

До таблиці `TagSnippet` входять наступні поля:

- `Id` типу `int` – номер мови програмування (первинний ключ таблиці);

- `TagId` типу `int` – тег (зовнішній ключ з таблиці `Tag`);

- `CodesnippetId` типу `int` – фрагмент програмного коду (зовнішній ключ з таблиці `Codesnippet`).

До таблиці `DeveloperTeam` входять наступні поля:

- `Id` типу `int` – номер запису;

- `TeamId` типу `int` – команда, до якої входить розробник (зовнішній ключ з таблиці `Team`);

- `DeveloperId` типу `int` – розробник (зовнішній ключ з таблиці `Developer`).

Структура бази даних (рис. 2.3) була підготовлена для визначення подальшої роботи з даними в програмі як на рівні бази даних, так і на рівні об'єктів у програмі.

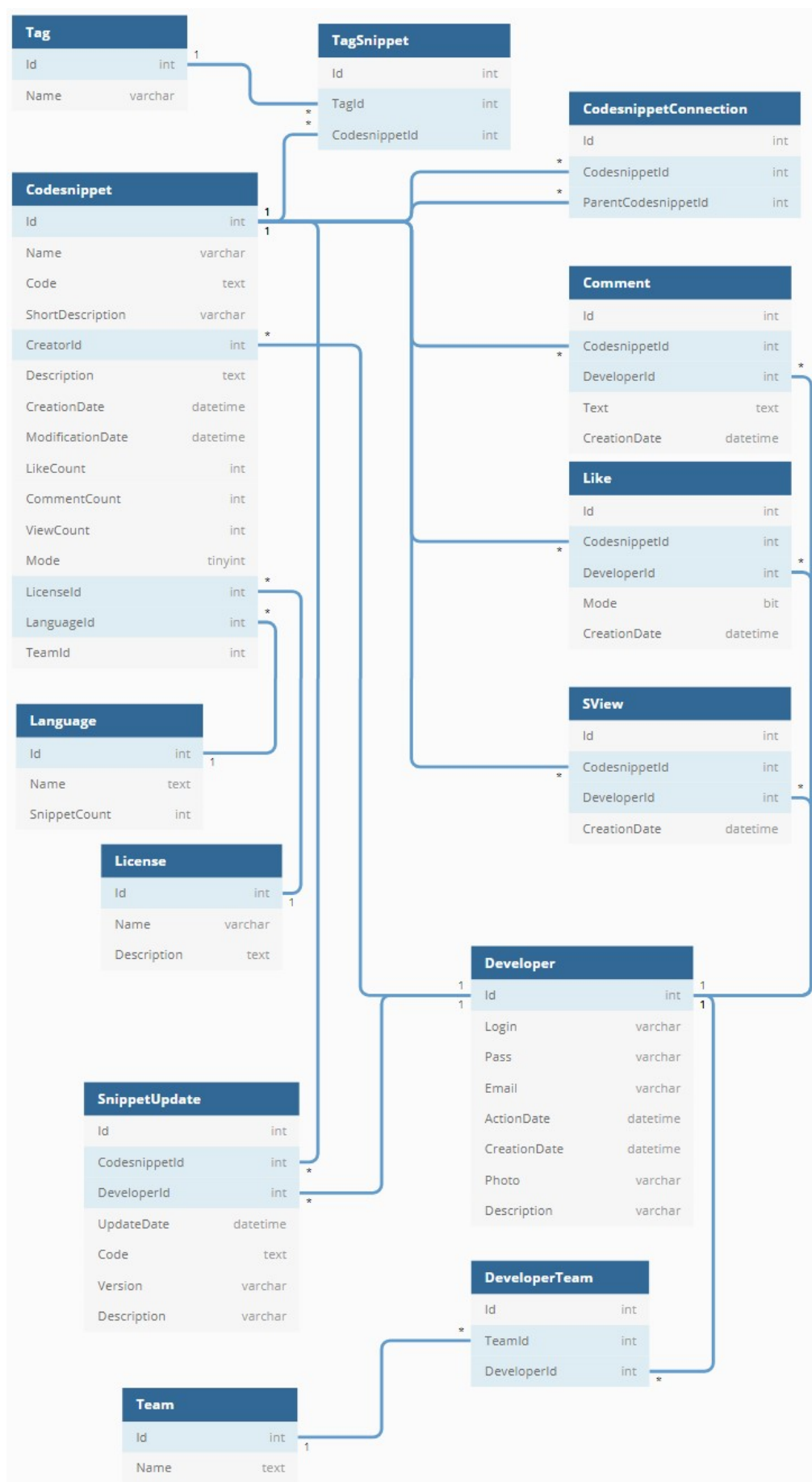


Рисунок 2.3 – Структура бази даних

До таблиці SnippetUpdate входять наступні поля:

- Id типу int – номер оновлення фрагмента програмного коду (первинний ключ таблиці);
- CodesnippetId типу int – фрагмент програмного коду, оновлення якого виконано (зовнішній ключ таблиці Codesnippet);
- DeveloperId типу int – розробник, який виконав оновлення (зовнішній ключ таблиці Developer);
- UpdateDate типу datetime – дата оновлення;
- Code типу text – новий код фрагмент (після оновлення);
- Version типу varchar – номер версії;
- Description типу varchar – опис оновлення.

До структури таблиці CodesnippetConnection, яка визначає по суті ієрархію зв'язків між фрагментами, входять наступні поля:

- Id типу int – номер зв'язування між фрагментами (первинний ключ таблиці);
- CodesnippetId типу int – фрагмент, який унаслідував код (зовнішній ключ таблиці Codesnippet);
- ParentCodesnippetId типу int – батьківський фрагмент програмного коду (зовнішній ключ таблиці Codesnippet).

Наявність таблиці CodesnippetConnection дозволяє визначати, на основі яких фрагментів було створено даний, що полегшує навігацію між фрагментами додатково як самим авторам фрагментів, так і учасникам команд розробки і пошукачам фрагментів, потрібних для їх роботи над новими проєктами.

2.4 Висновки за розділом 2

Виконане проєктування програмного забезпечення дозволило прийняти рішення про використання мови програмування C#, фреймворку розробки вебзастосунків ASP.NET, системи керування базами даних

Microsoft SQL Server, об'єктно-реляційного відображення Entity Framework, визначити архітектуру програми на основі архітектурного шаблону MVC, виділити функціональні вимоги до програми, визначити структуру бази даних.

3 РОЗРОБКА ПРОГРАМИ

3.1 Послідовність виконання застосунку

Застосунок реалізовано на основі використання фреймворку ASP.NET MVC, що визначає зокрема процедуру (рис. 3.1), за якою обробляється запит користувача і надається відповідь.

Після того, як користувач ставить запит, який браузером передається вебсерверу, на стороні вебсервера реалізується його обробка. `HttpApplication` відповідає за потік запитів, запускаючи події, які сигналізують програмі про те, що відбувається, реалізуючи цей процес як частину методу `HttpApplication.Init ()`, який встановлює та запускає послідовно низку подій, включаючи виклик для виконання будь-яких обробників, обробники подій відображають події, які автоматично налаштовуються в `global.asax`, а також відображають будь-які підключені `HttpModules`, які по суті є зовнішнім каналом для подій, які публікує `HttpApplication`, `HttpModules` і `HttpHandlers` динамічно завантажуються через записи в `web.config` і приєднуються до ланцюжка подій (`HttpModules` – це фактичні обробники подій, які підключають певні події `HttpApplication`, тоді як `HttpHandlers` – кінцева точка, яка отримує виклик для обробки запитів на рівні програми), і модулі, і обробники завантажуються і приєднуються до ланцюжка викликів як частина виклику методу `HttpApplication.Init ()` [13].

Наступним кроком реалізується маршрутизація, за допомогою якої визначається, який саме контролер має обробляти даний запит і вибирається з нього потрібний для оброблення фільтр, тобто конкретний метод, який забезпечить отримання результату за запитом. Після цього реалізується виконання методу і отримання результату. В залежності від типу отриманого об'єкту, або реалізується формування подання на основі рушія подань з формування відповідної вебсторінки, що повертається користувачу і

відображається у браузері, або реалізується безпосередньо передача результату (наприклад, у випадку виявленого виключення).



Рисунок 3.1 – Життєвий цикл запиту в ASP.NET-застосунку

3.2 Розроблені структурні елементи програми

Ключовим класом у організації взаємодії між моделями та контролерами застосунку є клас `DataContext`, створений шляхом наслідування класу `DbContext` – головним класом фреймворку роботи з даними Entity Framework, що забезпечує взаємодію з контекстом даних. Entity Framework забезпечує реалізацію об’єктно-реляційного відображення, в результаті чого не потрібно виконувати подвійну роботу з виконання тих самих дій і для об’єктів, і для таблиць бази даних, а можна застосовувати підхід Code First. Витягання і змінювання потрібних даних як раз реалізується через клас контексту даних `DataContext`, який містить наступні атрибути:

- `DbSet<Tag> Tags` – набір тегів, доступних у базі даних;
- `DbSet<Codesnippet> Codesnippets` – усі наявні фрагменти програмного коду;
- `DbSet<License> Licenses` – усі наявні ліцензії на програмне забезпечення;
- `DbSet<Language> Languages` – усі наявні мови програмування;
- `DbSet<SView> SViews` – усі наявні перегляди користувачем фрагментів;
- `DbSet<Like> Likes` – усі наявні уподобання користувачем фрагментів;
- `DbSet<Comment> Comments` – усі наявні коментарі до фрагментів;
- `DbSet<Developer> Developers` – усі наявні розробники в базі даних;
- `DbSet<Team> Teams` – усі наявні команди розробників.

Контекст даних існує з моменту його створення, відповідно в цей момент реалізується виклик методу `OnModelCreating(ModelBuilder mb)`, який визначає прив’язування відповідних атрибутів до таблиць бази даних.

Базовими для даного класу є класи моделей. Для кожної таблиці бази даних, окрім таблиць, що безпосередньо тільки реалізують зв'язок багато-до-багатьох, було створено окремі моделі. У даному випадку буде розглянуто тільки основні з них для демонстрації підходу для організації даних.

Клас моделі `Codesnippet` з простору імен моделей `CodeSnippetRepository.Models` включає такі атрибути:

- `public int Id` – атрибут, що відповідає полю ідентифікатора таблиці бази даних, позначений анотацією `[Key]` для визначення ролі первинного ключа з одночасним відображенням на однойменне поле таблиці бази даних;
- `public string Name` – атрибут, що зберігає назву фрагмента програмного коду і відображається на однойменне поле таблиці бази даних;
- `public string Code` – атрибут, що зберігає код фрагмента і відображається на однойменне поле таблиці бази даних, під час збереження значення виконується його оброблення з подальшим збереженням в результаті коду, розміченого за допомогою тегів, що дозволяє реалізувати підсвітку синтаксису, при цьому в якості базової бібліотеки для роботи над підсвічуванням синтаксису було використано бібліотеку `Highlight` [14];
- `public string ShortDescription` – атрибут, що зберігає короткий опис програмного коду і відображається на однойменне поле таблиці бази даних;
- `public int CreatorId` – атрибут, що зберігає номер запису розробника, який створив фрагмент програмного коду, і відображається на однойменне поле таблиці бази даних;
- `public string Description` – атрибут, що зберігає повний опис фрагмента програмного коду і відображається на однойменне поле таблиці бази даних;
- `public DateTime CreationDate` – атрибут, що зберігає дату створення фрагмента програмного коду і відображається на однойменне поле таблиці бази даних;
- `public string CreationDateStr` – атрибут, який не має відображення на поле таблиці бази даних (позначений анотацією `[NotMapped]`), формується

програмно і визначає рядкове представлення дати створення фрагмента програмного коду, необхідний для зручного відображення в поданні користувачу дати;

- `public DateTime ModificationDate` – атрибут, що зберігає дату останньої модифікації фрагмента програмного коду і відображається на однойменне поле таблиці бази даних;

- `public string ModificationDateStr` – атрибут, який не має відображення на поле таблиці бази даних, формується програмно і визначає рядкове представлення дати останньої модифікації фрагмента програмного коду, необхідний для зручного відображення в поданні користувачу дати;

- `public int LikeCount` – атрибут, який визначає кількість уподобань даного фрагмента, розраховується автоматично на основі колекції уподобань, не відображається в базу даних;

- `public int CommentCount` – атрибут, який визначає кількість коментарів даного фрагмента, розраховується автоматично на основі колекції коментарів, не відображається в базу даних;

- `public int SViewCount` – атрибут, який визначає кількість переглядів даного фрагмента, розраховується автоматично на основі колекції переглядів, не відображається в базу даних;

- `public int Mode` – атрибут, що зберігає режим відображення фрагмента програмного коду і відображається на однойменне поле таблиці бази даних;

- `public int LicenseId` – атрибут, що зберігає номер ліцензії, за якою розповсюджується фрагмент програмного коду, і відображається на однойменне поле таблиці бази даних;

- `public int LanguageId` – атрибут, що зберігає номер мови програмування, якою створено фрагмент програмного коду і відображається на однойменне поле таблиці бази даних;

- `public virtual Developer Creator` – атрибут, що визначає повну інформацію про автора фрагмента програмного коду на основі об'єкту

відповідного класу, на відміну від зовнішнього ключа `CreatorId`, де зберігається тільки номер запису;

– `public virtual License License` – атрибут, що визначає повну інформацію про ліцензію, за якою розповсюджується фрагмент програмного коду, на основі об'єкту відповідного класу, на відміну від зовнішнього ключа `LicenseId`, де зберігається тільки номер запису;

– `public virtual Language Language` – атрибут, що визначає повну інформацію про мову програмування, якою створено фрагмент програмного коду, на основі об'єкту відповідного класу, на відміну від зовнішнього ключа `LanguageId`, де зберігається тільки номер запису;

– `public virtual ICollection<Comment> Comments` – колекція коментарів, що реалізує зв'язок один-до-багатьох між таблицями фрагмента програмного коду та коментарів бази даних, включає коментарі, які були встановлені у відповідність даному фрагменту;

– `public virtual ICollection<SView> SViews` – колекція переглядів, що реалізує зв'язок один-до-багатьох між таблицями фрагмента програмного коду та переглядів бази даних, включає перегляди даного фрагмента;

– `public virtual ICollection<Like> Likes` – колекція уподобань, що реалізує зв'язок один-до-багатьох між таблицями фрагмента програмного коду та уподобань бази даних, включає уподобання даного фрагмента;

– `public virtual ICollection<Tag> Tags` – колекція коментарів, що реалізує зв'язок один-до-багатьох між таблицями фрагмента програмного коду та коментарів бази даних, включає теги даного фрагмента;

– `public virtual ICollection<CodesnippetConnection> CodesnippetConnections` – колекція пов'язаних фрагментів з даним фрагментом, що реалізує зв'язок один-до-багатьох між таблицями фрагментів програмного коду;

– `public virtual ICollection<SnippetUpdate> SnippetUpdates` – колекція оновлень фрагмента, що реалізує зв'язок один-до-багатьох між таблицями фрагмента програмного коду та оновлень бази даних, включає оновлення

даного фрагмента, при цьому самі оновлення в контексті даних не зберігаються, тільки у складі моделі фрагмента.

На основі аналогічного підходу було створено класи інших моделей.

Серед контролерів, найчастіше використовуваним, є контролер для роботи з фрагментами програмного коду – `CodesnippetController`. Окрім нього створено також зокрема контролери команд `TeamController`, уподобань `LikeController`, коментарів `CommentController` та інші. Кожен контролер визначає подання, що повертається в результаті роботи. Для кожного такого контролера було розроблено відповідний набір подань, що забезпечує виведення користувачу відповідних даних.

Контролер `CodesnippetController` містить атрибут для звернення до контексту даних та наступні методи.

Метод `Task<IActionResult> PopularSnippets()` формує подання для відображення найбільш популярних фрагментів програмного коду, тобто фрагментів, відсортованих за зменшенням кількості переглядів розробниками. Для цього спочатку було реалізовано наповнення через контекст даних відповідними даними фрагментів програмного коду, що необхідно для відображення інформації про них у переліках:

```
var snippets = await _dbc.Codesnippets.Include(s => s.Creator).Include(s
=> s.License).Include(s => s.Language).Include(s =>
s.Team).Include(Comments).Include(Likes).Include(SnippetUpdate).Include(SVie
ws).Include(SViewCount).Include(LikeCount).Include(CommentCount).Include(C
reationDateStr).Include(ModificationDateStr).Include(CodesnippetConnections).In
clude(SnippetUpdates);
```

Далі забезпечується сортування за зменшенням кількості переглядів, а після цього сформований перелік передається в якості об'єкту результату в подання, що повертається в результаті роботи методу контролера:

```
snippets = snippets.OrderByDescending(s => s.SViewCount);
return View(await snippets.ToListAsync());
```

Метод `Task<IActionResult> BestSnippets()` формує подання для відображення найкращих фрагментів програмного коду, тобто фрагментів, які отримали найбільшу кількість уподобань користувачів.

Метод `Task<IActionResult> OwnSnippets()` формує подання для відображення фрагментів програмного коду, для яких авторизований у даний момент розробник є автором.

Метод `Task<IActionResult> TeamSnippets(int? teamid)` формує подання для відображення фрагментів, з якими працює команда з номером `teamid`.

Метод `Task<IActionResult> TagSnippets(string tagname)` формує подання для відображення фрагментів програмного коду, що відзначені тегом `tagname`.

Метод `Task<IActionResult> FindSnippets(string snippetname, string language)` формує подання, що відображається за результатом пошуку фрагментів, написаних мовою програмування `language`, та таких, що містять у своїй назві текст `snippetname`.

Метод `Task<IActionResult> OwnTeamsSnippets()` формує подання для відображення усіх фрагментів програмного коду, з якими може працювати авторизований у даний момент користувач через те, що він є учасником команд розробки, з якими пов'язані дані фрагменти.

Метод `Task<IActionResult> Details(int? snippetId)` виконує відображення деталізованого представлення фрагмента програмного коду з номером `snippetId`.

Метод `IActionResult Create()` виконує підготовку до створення фрагмента програмного коду.

Метод `Task<IActionResult> Create([Bind("Id, Name, ShortDescription, Description, Mode, LicenseId, LanguageId")] Codesnippet codesnippet)` виконує збереження фрагмента програмного коду.

Основні описані класи, що реалізують основну логіку роботи, винесено на діаграму класів (рис. 3.2).

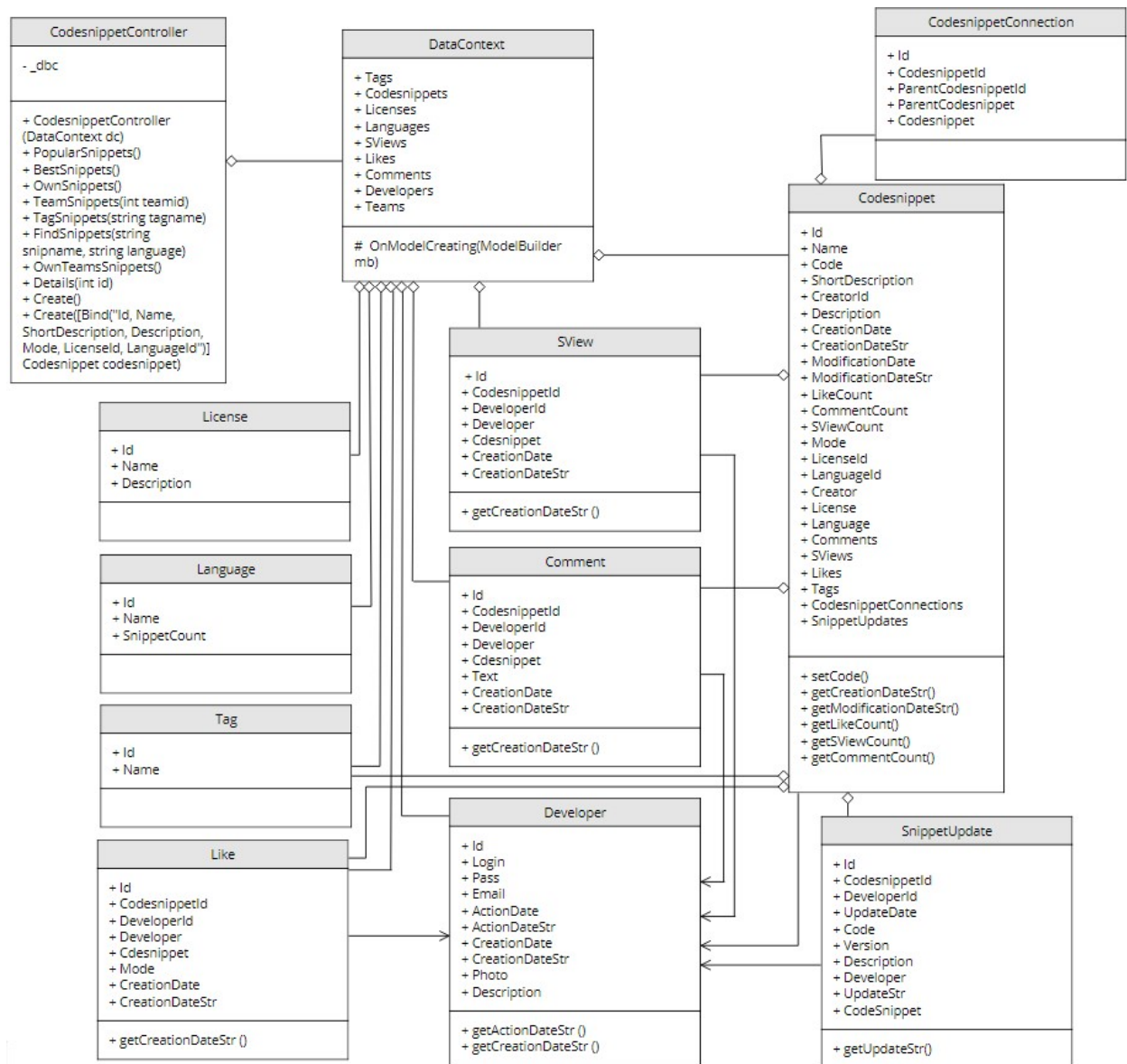


Рисунок 3.2 – Діаграма класів

Для створення подань було застосовано шаблон [15].

Основним елементом, який включає більшість подань, є таблиця перегляду переліку фрагментів програмного коду, елементи якої визначаються у поданні наступним чином:

```
@foreach (var item in Model) {
```

```

<tr>
    <td><a                asp-action="Details"                asp-route-
snippetId="@item.Id">@Html.DisplayFor(modelItem => item.Name)</a></td>
    <td>@Html.DisplayFor(modelItem => item.Creator.Login)</td>
    <td>@Html.DisplayFor(modelItem                                =>
item.ModificationDateStr)</td>
    <td><a                asp-action="LanguageSnippets"                asp-route-
languageId="@item.LanguageId">@Html.DisplayFor(modelItem                                =>
item.Language.Name)</a></td>
    <td>@Html.DisplayFor(modelItem => item.LikeCount)</td>
    <td>@Html.DisplayFor(modelItem => item.SViewCount)</td>
    <td>@Html.DisplayFor(modelItem => item.CommentCount)</td>
</tr>
}

```

3.3 Висновки за розділом 3

У розділі розглянуто послідовність виконання застосунку на основі життєвого циклу запиту до вебсервера, представленого у вигляді діаграми діяльності. Описано основні розроблені класи контексту даних, моделей, контролерів та процес формування подань.

4 ЕКСПЛУАТАЦІЯ ТА ТЕСТУВАННЯ ПРОГРАМИ

4.1 Призначення програми

Програма призначена для підтримки роботи репозиторію фрагментів програмного коду та забезпечення доступу до них індивідуальним розробникам та командам розробників. Задля цього надано широку кількість засобів перегляду і пошуку фрагментів програмного коду.

Програма направлена на забезпечення роботи з власними фрагментами з одного боку, а з іншого – на пошук фрагментів інших розробників з метою впровадження фрагментів у власний код.

4.2 Умови виконання програми

Умовою виконання програми є використання наступних апаратних вимог для запуску вебсервера:

- процесор: 4 ядра, тактова частота – 2,9 ГГц;
- оперативна пам'ять – 10 Гб;
- платформа – 32-розрядна або 64-розрядна;
- жорсткий диск – 1 Гб вільного простору.

4.3 Робота з програмою

Пункти меню вебсайту для неавторизованого користувача містять наступні розділи.

Розділ «Фрагменти» призначений для переходу до перегляду фрагментів. Після вибору даного розділу відкривається сторінка, вгорі якої розташовано підменю, що для неавторизованого користувача включає посилання «Популярні» і «Найкращі».

Підпункт «Популярні» дозволяє переглянути перелік фрагментів програмного коду, відсортованих за кількістю переглядів.

Підпункт меню «Найкращі» дозволяє переглянути перелік фрагментів програмного коду, відсортованих за кількістю уподобань (рис. 4.1).

РЕПОЗИТОРІЙ
ФРАГМЕНТІВ
ПРОГРАМНОГО
КОДУ

Пошук

ФРАГМЕНТИ

ТЕГИ

МОВИ

АВТОРИЗАЦІЯ

| ПОПУЛЯРНІ | **НАЙКРАЩІ** |

Фрагменти з найбільшою кількістю уподобань

Назва	Автор	Оновлення	Мова	♥	👁	✉
Код створення форми з кнопкою для завантаження файлу на сервер						
File Upload Form	bestdeveloper	28/04/2021	PHP	12	14	2
Скрипт для створення CAPTCHA-зображення у вигляді розв'язання логічної задачі						
CAPTCHA creator	heroku	29/04/2021	PHP	11	15	0
Реалізація алгоритму швидкого сортування						
Quick Sort	chucha	01/05/2021	Java	11	12	3
Код для визначення розташування, включаючи місто і країну або штат, за IP-адресою						
Location detector	developer81	28/04/2021	PHP	11	11	0
Код для генерації дерева за заданим рядком						
Tree Generator	bestdeveloper	28/04/2021	C++	10	10	1
Фрагмент для підрахунку кількості входжень символів у рядок						
Symbol Calculator	sanalberto	30/04/2021	C++	9	10	0
Універсальний парсер дат з конвертором						
Date Parser	bestdeveloper	01/05/2021	Java	9	10	1
Код оптимізованого виконання конкатенації двох і більше рядків у один						
String Concatenation	sanalberto	28/04/2021	C	8	9	0
Код для генерації XML-файлу з заданого шаблону						
XML Creator	ruddi	02/05/2021	PHP	8	8	2
Код для автоматичного відправлення повідомлень в Twitter за заданим у текстовому файлі шаблоном						
Twitter Poster	luka	02/05/2021	Python	8	8	1

Рисунок 4.1 – Перегляд найкращих фрагментів програмного коду за оцінками користувачів

Для кожного фрагмента в переліку відображається спочатку його короткий опис на білому рядку, а далі на сірому – його назва, автор (логін

розробника, який створив даний фрагмент), дата останнього оновлення, мова програмування, якою створено фрагмент, сумарна кількість уподобань, переглядів та коментарів до фрагмента.

Якщо натиснути на назву будь-якого фрагмента, то відкриється сторінка з повною інформацією про даний фрагмент програмного коду. Якщо натиснути на назву мови програмування, якою було створено фрагмент, то відкриється перелік фрагментів, також написаних цією мовою програмування.

Розділ меню «Теги» дозволяє переглянути всі теги, відсортовані за кількістю фрагментів, які було створено за цим тегом. Звідси можна обрати будь-який натиснувши на його назву та відобразити сторінку з переліком фрагментів, які пов'язані з даним тегом.

Розділ меню «Мови» (рис. 4.2) дозволяє переглянути перелік усіх мов програмування, якими було створено фрагменти програмного коду на вебсайті. Усі мови відсортовані за кількістю фрагментів, які було створено відповідною мовою програмування. Натискаючи на будь-яку з них, можна перейти до перегляду фрагментів, які написані цією мовою.

Меню також містить рядок пошуку, який дозволяє швидко виконати пошук фрагмента за його назвою або її частиною. Якщо ввести пошуковий запит і натиснути на кнопку Enter, то відкриється сторінка з результатами пошуку, окрім того на цій сторінці можна ввести заново пошуковий запит та задати мову програмування, якою має бути створено фрагменти.

Пункт меню «Авторизація» дозволяє перейти до сторінки авторизації вебсайту, щоб увійти під власним акаунтом і мати можливість створювати нові фрагменти програмного коду і керувати існуючими.

За роботи з програмою авторизованого користувача для нього відображається розширене меню, що включає наступні розділи.

Розділ «Створити» дозволяє внести в репозиторій новий фрагмент програмного коду.

РЕПОЗИТОРІЙ ФРАГМЕНТІВ ПРОГРАМНОГО КОДУ	
Пошук	
ФРАГМЕНТИ	
ТЕГИ	
МОВИ	
АВТОРИЗАЦІЯ	

Мови програмування	
Назва	Кількість фрагментів
PHP	41
Java	37
C++	32
Python	31
C	25
C#	24
JavaScript	22
Kotlin	16
Ruby	14
HTML	13
Go	11
Shell	8
Perl	7
ActionScript	6
Objective C	6
Lua	4
Matlab	2
R	2
Assembler	1
Pascal	1

Рисунок 4.2 – Перегляд мов програмування

Розділ «Фрагменти» дозволяє, як і для неавторизованого користувача, працювати з переліками фрагментів програмного коду, але набір підрозділів включає додатково «Власні» (рис. 4.3) та «Командні», що дозволяють відповідно переглянути перелік всіх фрагментів, автором якого є авторизований розробник, та перелік фрагментів, над якими працюють команди, учасником яких є авторизований розробник. Саме з підрозділу «Власні» можна виконувати редагування існуючих фрагментів за необхідності.

Розділ «Уподобання» дозволяє переглянути перелік фрагментів програмного коду, які раніше уподобав користувач. Фрагменти відображаються в порядку віддаленості від поточної дати. Даний розділ

зокрема дозволяє накопичувати фрагменти, які розробник вважає потенційно важливими для подальшого використання у власних проектах.

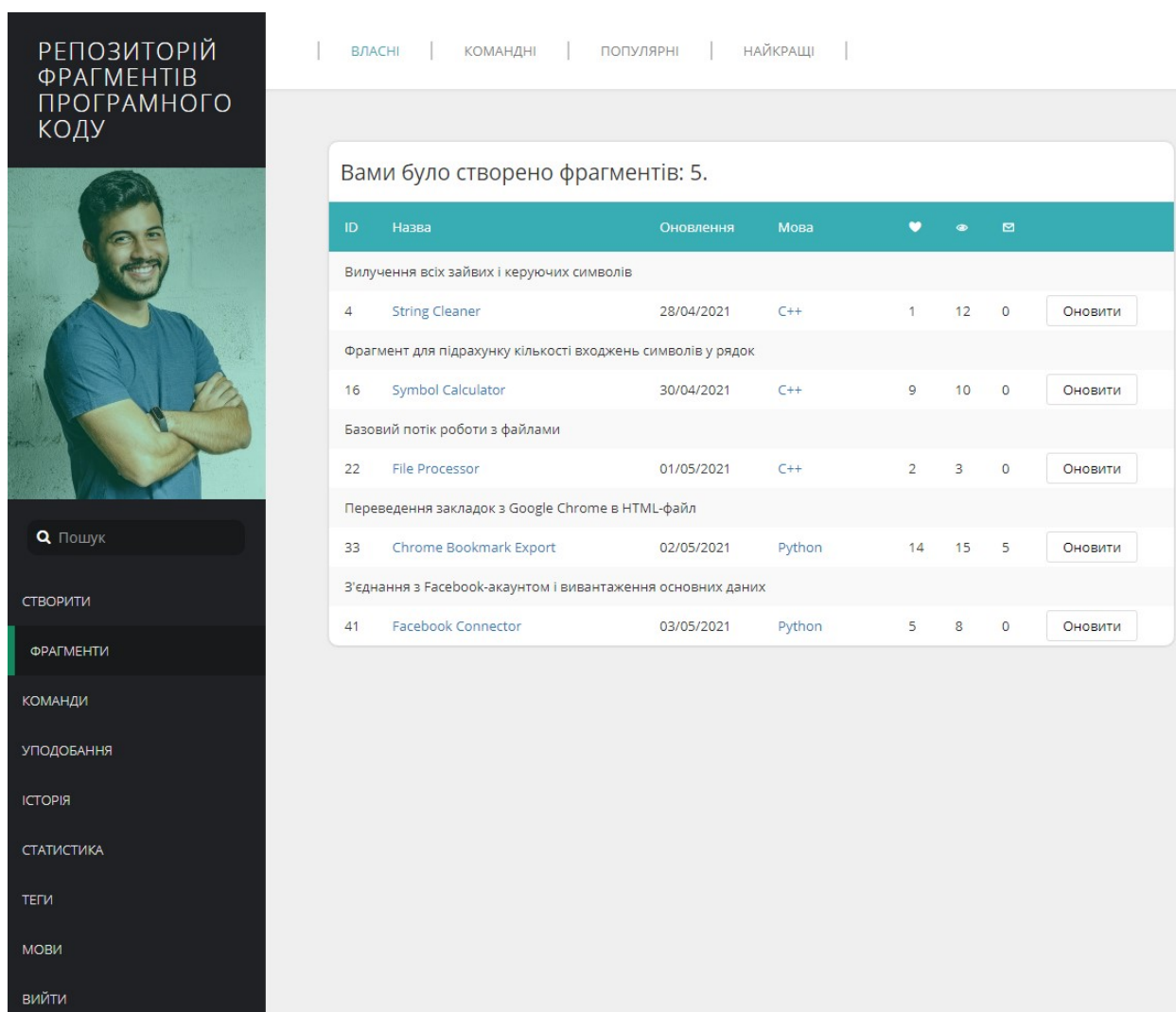


Рисунок 4.3 – Перегляд власних фрагментів програмного коду

Розділ «Команди» дозволяє виконувати керування командами, до яких входить авторизований розробник (рис. 4.4).

Підменю для розділу «Команди» включає пункти «Створити команду», що дозволяє створити нову команду, визначаючи її назву та додаючи до неї учасників, та «Команди», що дозволяє переглянути перелік команд.

Підрозділ «Команди» представляє таблицю, у якій містяться назви всіх команд, до складу яких входить авторизований розробник. Для кожної

команди виводиться назва, склад її учасників, кожен з яких визначений логіном. За необхідності відредагувати назву команди потрібно натиснути на кнопку «Редагувати» у стовпці з назвою команди. За необхідності змінити склад учасників, потрібно натиснути на кнопку «Редагувати», розташовану за стовпчиком учасників. Кнопка «Додати» дозволяє додати до складу команди нового учасника. Кнопка «Фрагменти» дозволяє перейти до переліку фрагментів обраної команди. Звідти фрагменти можна і редагувати.

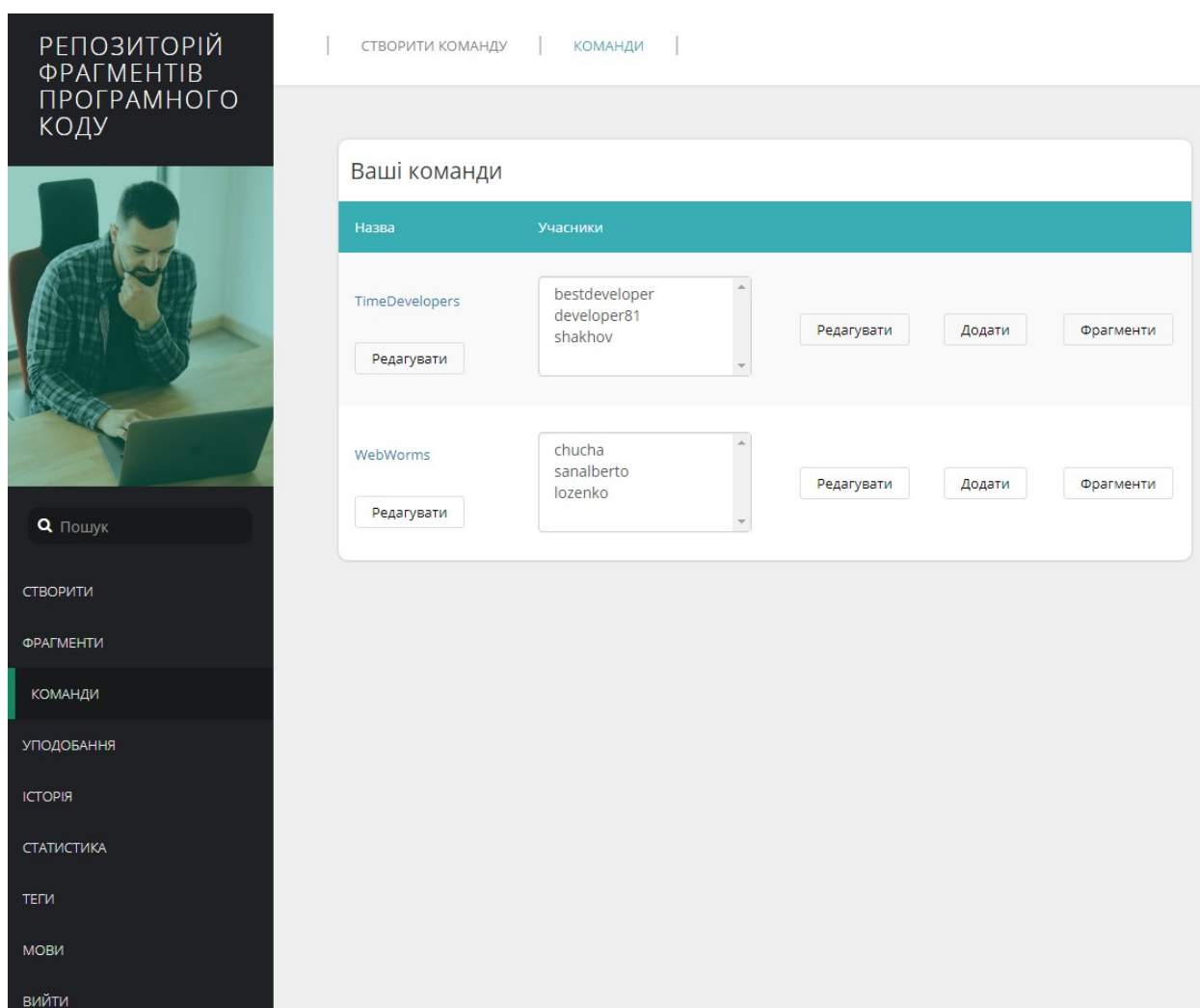


Рисунок 4.4 – Керування командами розробників

Розділ «Історія» дозволяє переглянути всі раніше переглянуті авторизованим користувачем фрагменти програмного коду. За допомогою даного розділу можна повернутися до будь-якого фрагмента, який зацікавив, але не залишився в улюблених.

Розділ «Статистика» дозволяє переглянути статистичні показники щодо роботи з фрагментами за минулі місяці.

Розділ «Статистика» включає два підрозділи: «Загальна» і «Пофрагментна».

Підрозділ «Загальна» дозволяє переглянути загальну статистику за авторизованим розробником (рис. 4.5). У такому випадку розробнику відображається сторінка, яка містить зверху загальну його статистику, а знизу – помісячну статистику.

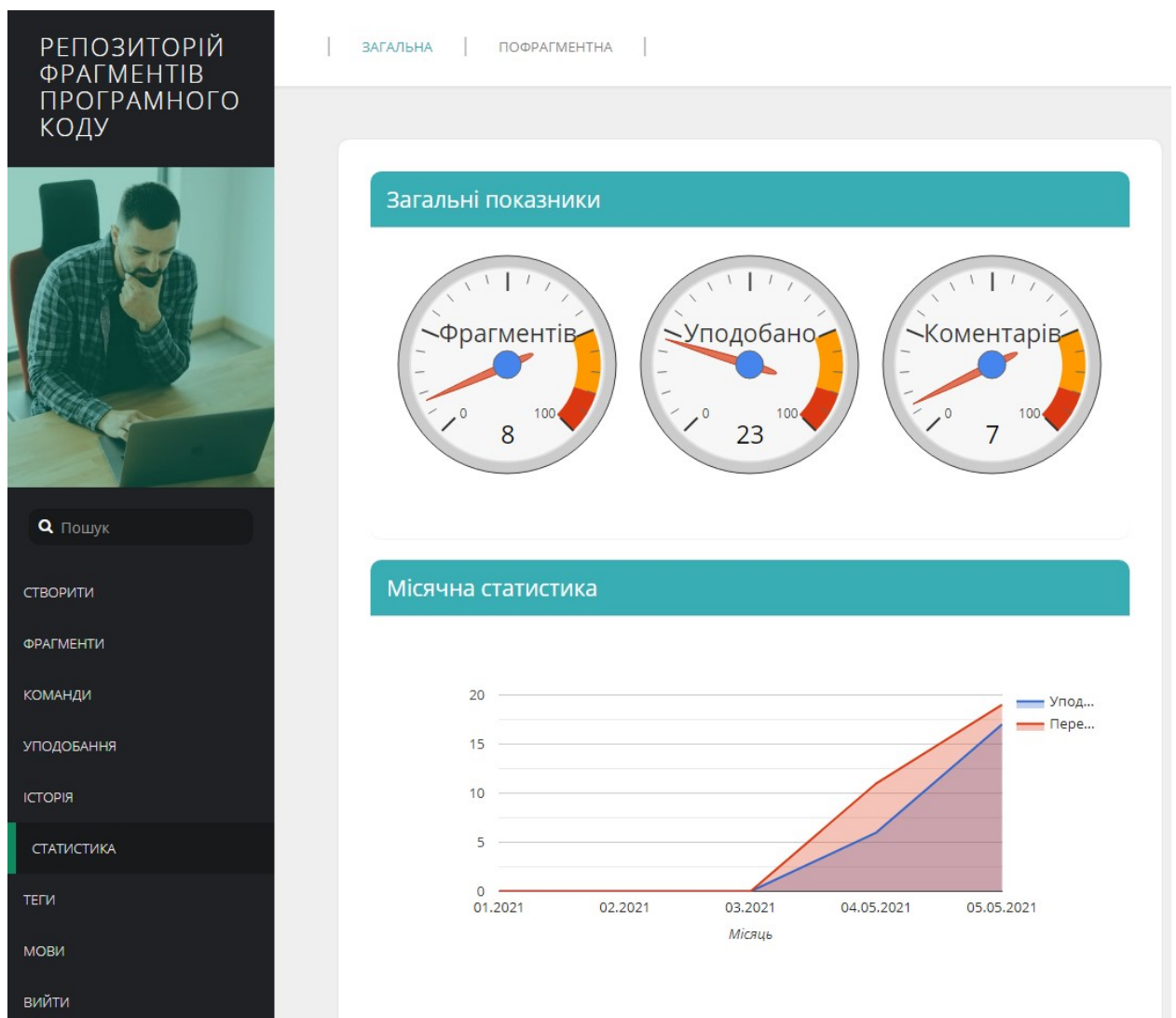


Рисунок 4.5 – Перегляд статистики

Загальна статистика включає сумарну кількість фрагментів, які створив розробник протягом всього часу роботи, кількість уподобань, яку за

весь час було поставлено усім фрагментам розробника, де він є автором, а також кількість коментарів, які отримали його авторські фрагменти програмного коду.

Місячна статистика представлена у вигляді графіку, на якому червоним відображається кількість переглядів, а синім – уподобань. Дані відображаються за минулі 5 місяців (включаючи поточний). За кожен місяць відображається сумарна кількість уподобань та переглядів фрагментів, в яких автором є авторизований розробник.

Для перегляду відповідних графіків за конкретними фрагментами програмного коду потрібно перейти до підрозділу «Пофрагментна», звідки можна обрати потрібний фрагмент та переглянути за ним статистику.

Під час вибору розділу «Створити», відкривається сторінка створення нового фрагмента програмного коду (рис. 4.6).

Під час створення фрагмента програмного коду потрібно задати наступне:

- назву фрагмента;
- короткий опис фрагмента;
- повний опис фрагмента, що буде відображатися на сторінці деталізації;
- код фрагмента;
- обрати мову програмування, якою написано фрагмент, зі списку;
- обрати команду, яка може працювати над фрагментом, зі списку;
- задати можливість перегляду фрагмента всіма користувачами програми (встановивши прапор), або надати таку можливість тільки автору і команді (скасувавши прапор);
- обрати зі списку варіант ліцензії, за якою має розповсюджуватися програмний код;
- додати до списку назви фрагментів, на основі яких було створено даний фрагмент;
- додати до списку теги або створити нові теги і додати їх до списку.

РЕПОЗИТОРІЙ
ФРАГМЕНТІВ
ПРОГРАМНОГО
КОДУ



Пошук

- СТВОРИТИ
- ФРАГМЕНТИ
- КОМАНДИ
- УПОДОБАННЯ
- ІСТОРІЯ
- СТАТИСТИКА
- ТЕГИ
- МОВИ
- ВИЙТИ

Внесення нового фрагмента програмного коду

Внесіть невеликий фрагмент програмного коду в репозиторій.

Назва

Короткий опис

Повний опис

Код

```
using System.Net;
using System.Net.Mail;
var email_from = new MailAddress( "from@example.com", "From Name");
var email_to = new MailAddress("to@example.com", "To Name");
const string senderPassword = "Password";
const string subject = "Subject";
const string body = "Body";
var smtp = new SmtpClient
{
    Host = "smtp.gmail.com",
    Port = 587,
    EnableSsl = true,
    DeliveryMethod = SmtpDeliveryMethod.Network,
    UseDefaultCredentials = false,
    Credentials = new NetworkCredential(
```

Мова програмування

Команда

Ліцензія

☒ Відкрити для всіх

Рисунок 4.6 – Створення нового фрагмента програмного коду

Якщо авторизований користувач натисне на назву будь-якого з власних або командних фрагментів програмного коду, то відкриється як і для неавторизованого користувача сторінка деталізації фрагмента (рис. 4.7).

Під час перегляду фрагмента можна побачити назву фрагмента, автора, набір тегів з можливістю переходу на фрагменти за ними, інформацію про кількість уподобань, переглядів, дату створення і останнього оновлення, мову програмування, ліцензію, команду, опис, виконані оновлення (щоб відобразити інформацію за цією версією, потрібно обрати зі списку один з варіантів і натиснути на кнопку «Показати»), пов'язані фрагменти (можна перейти до відповідного фрагмента натисканням на його назву або вибором декількох і натисканням на кнопку «Фрагмент», тоді відкриється одразу набір вкладок у вікні браузера).

Код фрагмента відображається окремою плашкою з підсвічуванням синтаксису. Кожну з плашок при потребі можна закрити (правий верхній кут) і концентруватися на потрібній.

Якщо при перегляді сторінки натиснути на фотографію в панелі меню зліва, яка є фотографією авторизованого користувача, то можна перейти до сторінки внесення змін до профілю користувача.

Усі інші розділи аналогічні розділам неавторизованого користувача.

The screenshot displays a web application interface. On the left is a dark sidebar menu with the title 'РЕПОЗИТОРІЙ ФРАГМЕНТІВ ПРОГРАМНОГО КОДУ' and a list of navigation items: 'Пошук', 'СТВОРИТИ', 'ФРАГМЕНТИ' (highlighted), 'КОМАНДИ', 'УПОДОБАННЯ', 'ІСТОРІЯ', 'СТАТИСТИКА', 'ТЕГИ', 'МОВИ', and 'ВИЙТИ'. The main content area features a 'PASSWORD GENERATOR' card by 'lozenko'. This card includes a 'ПОДОБАЄТЬСЯ' button, a hashtag '#password #c #randomizer', and a list of statistics: 'Уподобань: 3.', 'Переглядів: 5.', 'Створено: 03/05/2021.', 'Оновлено востаннє: 04/05/2021.', 'Мова програмування: C#.', 'Ліцензія: немає.', and 'Команда: немає.'. It also contains a description: 'Фрагмент коду, який може використовуватися для забезпечення автоматизованої генерації пароля користувача під час реєстрації.' To the right of the statistics is an 'Оновлення:' section with a dropdown menu showing versions '03/05/2021 версія 1.0' and '04/05/2021 версія 1.1', a 'ПОКАЗАТИ' button, and a 'Пов'язані:' section with a dropdown menu showing 'Randomizer' and 'String Processor', and a 'ФРАГМЕНТ' button. Below the card, a code snippet is displayed with syntax highlighting:

```

public static string PasswordGenerator()
{
    string alphabet = "1234567890ABCDEFGHIJKLMNOPQRSTUVWXYZ";
    Random randNum = new Random();
    char[] pass = new char[10];
    int alphabetLength = alphabet.Length;
    for (int i = 0; i < 10; i++)
    {
        password[i] = alphabet[(int)((alphabet.Length) * randNum.NextDouble())];
    }
    return new string(pass);
}

```

Рисунок 4.7 – Перегляд фрагмента програмного коду

ВИСНОВКИ

У результаті проведеного аналізу предметної області було визначено поняття фрагментів програмного коду, особливості їх використання, основні ліцензії програмного забезпечення, виконано огляд програмних аналогів, включаючи Snipplr, Djangosnippets, для них визначено їх функціональні можливості, переваги і недоліки та виконано порівняння зі створенням відповідної таблиці порівняння. У якості основного програмного аналогу для розробки було обрано Snipplr. Основною відмінністю створеного програмного забезпечення порівняно зі Snipplr є можливість роботи в команді над фрагментами програмного коду та надання статистики про фрагменти програмного коду в динаміці.

Виконане проєктування програмного забезпечення дозволило прийняти рішення про використання мови програмування C#, фреймворку розробки вебзастосунків ASP.NET, системи керування базами даних Microsoft SQL Server, об'єктно-реляційного відображення Entity Framework, визначити архітектуру програми на основі архітектурного шаблону MVC, виділити функціональні вимоги до програми, визначити структуру бази даних.

Виконана розробка є вебзастосунком, що призначений для підтримки роботи з репозиторієм фрагментів програмного коду та забезпечення доступу до них індивідуальним розробникам і командам розробників. Програма направлена на забезпечення роботи з власними фрагментами з одного боку, задля цього надаються засоби доступу до створення фрагментів, редагування власних та командних фрагментів, моніторингу статистики роботи з фрагментами інших користувачів, а з іншого – на пошук фрагментів інших розробників з метою впровадження фрагментів у власний код, задля чого надано широку кількість засобів перегляду і пошуку фрагментів програмного коду.

Усі завдання дипломної кваліфікаційної роботи повністю виконано.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Brophy, M. Code Snippets - A Beginner's Guide [Electronic resource] / M. Brophy. – Access mode : <https://dev.to/matthewbrophy/code-snippets---a-beginners-guide-p4d>
2. Nelson, K. Software License Types Explained [Electronic resource] : What You Need to Know / K. Nelson. – Access mode : <https://www.cherwell.com/software-asset-management/library/blog/software-license-types/>
3. Tollen, D.W. The Tech Contracts Handbook [Text] : Cloud Computing Agreements, Software Licenses, and Other IT Contracts for Lawyers and Businesspeople / D.W. Tollen. – New York : American Bar Association, 2016. – 275 p.
4. 5 types of software licenses you need to understand [Electronic resource]. – Access mode : <https://www.synopsys.com/blogs/software-security/5-types-of-software-licenses-you-need-to-understand/>
5. Young, M. Software Licensing Agreements [Text] : What You Need To Know About Software Licenses / M. Young. – Dallas : Hokkaido Ventures LLC, 2019. – 51 p.
6. Classen, H. W. A Practical Guide to Software Licensing for Licensees and Licensors [Text] / H. W. Classen. – Chicago : American Bar Association, 2017. – 848 p.
7. Blokdyk, G. Software License Management Process A Complete Guide [Text] / G. Blokdyk. – Brendale : 5STARCooks, 2021. – 307 p.
8. Code Snippet – Snippet Social Repository [Electronic resource]. – Access mode : <https://snipplr.com/>
9. Djangosnippets [Electronic resource] : Welcome. – Access mode : <https://djangosnippets.org/>
10. Price, M. J. C# 9 and .NET 5 – Modern Cross-Platform Development [Text] : Build intelligent apps, websites, and services with Blazor, ASP.NET Core,

and Entity Framework Core using Visual Studio Code / M. J. Price. – Birmingham : Packt Publishing, 2020. – 822 p.

11. Stauffer, M. Laravel: Up & Running [Text] : A Framework for Building Modern PHP Apps / M. Stauffer. – Sebastopol : O'Reilly, 2019. – 554 p.

12. Murach, J. Murach's ASP.NET Core MVC [Text] / J. Murach. – Fresno, CA : Mike Murach & Associates, 2020. – 780 p.

13. Strahl, R. A Low Level Look at ASP.NET Architecture [Electronic resource] / S. Strahl. – Access mode : <https://www.codemag.com/article/0511061>

14. Thomasjo/highlight [Electronic resource] : A relatively simple and extensible syntax highlighter written in C#. – Access mode : <https://github.com/thomasjo/highlight>

15. Free Template 455 Visual Admin [Electronic resource]. – Access mode : <https://templatemo.com/tm-455-visual-admin>

ДОДАТОК А
Технічне завдання

Вступ

Виконувана розробка має підтримувати репозиторій фрагментів програмного коду, тобто забезпечувати накопичення в базі даних фрагментів програмного коду з можливістю віддаленого доступу, має підтримуватися робота з різними мовами програмування, не обмежуючись певним набором.

A.1 Підстави для розробки

Підставою для виконуваної розробки в межах дипломної кваліфікаційної роботи бакалавра є завдання, що було визначене за темою «Програмне забезпечення керування репозиторієм фрагментів програмного коду» і затверджене у відповідності з наказом № 103 від 30 березня 2021 р. за Національним університетом «Запорізька політехніка».

A.2 Призначення розробки

Розробка призначена для підтримки роботи репозиторію фрагментів програмного коду та забезпечення доступу до них індивідуальним розробникам та командам розробників. Задля цього має бути надано широку кількість засобів перегляду і пошуку фрагментів програмного коду.

Програма має бути направленою на забезпечення роботи з власними фрагментами з одного боку, а з іншого – на пошук фрагментів інших розробників з метою впровадження фрагментів у власний код.

A.3 Основні вимоги до програми

A.3.1 Вимоги до функціональних характеристик

Програма повинна забезпечувати виконання наступних функціональних характеристик:

- створення фрагментів програмного коду;
- прив'язування фрагментів програмного коду до команд розробників;
- оновлення фрагментів програмного коду;
- визначення режиму перегляду фрагмента;
- перегляд власних фрагментів програмного коду;
- пошук фрагментів програмного коду;
- перегляд найпопулярніших фрагментів програмного коду;
- перегляд найбільш уподобаних фрагментів програмного коду;
- перегляд фрагмента програмного коду;
- перегляд фрагментів програмного коду команди;
- копіювання коду фрагмента в пам'ять;
- визначення команди розробників;
- змінювання учасників у команді розробників;
- перегляд команд, до складу яких входить розробник;
- перегляд фрагментів за мовою програмування;
- перегляд фрагментів за тегом;
- перегляд історії оновлення фрагмента програмного коду;
- перегляд популярності мов програмування;
- перегляд статистики уподобань фрагмента;
- перегляд статистики переглядів фрагмента;
- перегляд статистики уподобань усіх фрагментів розробника;
- перегляд статистики переглядів усіх фрагментів розробника;
- перегляд усіх уподобаних фрагментів користувачем;
- перегляд усіх переглянутих раніше фрагментів користувачем;
- прибирання уподобання фрагмента;
- уподобання фрагмента програмного коду;
- створення коментаря до фрагмента програмного коду;
- реєстрація розробників;

– авторизація розробників.

A.3.2 Умови експлуатації

Експлуатація програмного забезпечення має виконуватися на основі клієнт-серверної моделі. Для цього потрібно виділити окремий комп'ютер або скористатися виділеним сервером, який буде забезпечувати підтримку реалізації вебсервера. Вебсервер має бути доступним для запитів користувачів постійно.

A.3.3 Вимоги до складу та параметрів технічних засобів

Склад та параметри технічних засобів вебсервера було визначено для реалізації можливості доступу 300 користувачів одночасно. Відповідно вимоги можуть бути знижені, якщо потрібне забезпечення меншої кількості запитів, а для більшої кількості одночасних запитів вимоги навпаки мають бути збільшені. Мінімальні системні вимоги до вебсервера включають:

- процесор: 4 ядра, тактова частота – 2,9 ГГц;
- оперативна пам'ять – 10 Гб;
- платформа – 32-розрядна або 64-розрядна;
- жорсткий диск – 1 Гб вільного простору.

A.3.4 Вимоги до транспортування та збереження

Безпосередньо програма має транспортуватися та зберігатися через репозиторій вебсервера. Збереження на окремих фізичних носіях не передбачається.

A.3.5 Вимоги до програмної документації

Основою розробки даного програмного проєкту є технічне завдання на нього.

Опис прийнятих рішень виконується в пояснювальній записці. У розділі «Експлуатація та тестування програми» має бути представлено опис процесу роботи з програмою. Інших програмних документів не передбачається.

A.4 Порядок контролю та приймання

Календарний план завдання на дипломну кваліфікаційну роботу визначає набір етапів, за якими має забезпечуватися виконання, а також результати, які мають бути отримані за виконання відповідного етапу. Саме на основі даних результатів має забезпечуватися приймання роботи.

ДОДАТОК Б
Текст програми

Б.1 Текст файла DataContext.cs

```

using CodeSnippetRepository.Models;
using Microsoft.EntityFrameworkCore;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;

namespace CodeSnippetRepository.Data
{
    public class DataContext : DbContext
    {
        public DbSet<Tag> Tags
        { get; set; }
        public DbSet<Codesnippet> Codesnippets
        { get; set; }
        public DbSet<License> Licenses
        { get; set; }
        public DbSet<Language> Languages
        { get; set; }
        public DbSet<SView> SViews
        { get; set; }
        public DbSet<Like> Likes
        { get; set; }
        public DbSet<Comment> Comments
        { get; set; }
        public DbSet<Developer> Developers
        { get; set; }
        public DbSet<Team> Teams
        { get; set; }
        protected override void
OnModelCreating(ModelBuilder mb)
        {
            mb.Entity<Tag>().ToTable("Tag");

mb.Entity<Codesnippet>().ToTable("Codesnippet");
            mb.Entity<License>().ToTable("License");
            mb.Entity<Language>().ToTable("Language");
            mb.Entity<SView>().ToTable("SView");
            mb.Entity<Like>().ToTable("Like");
            mb.Entity<Comment>().ToTable("Comment");
        }
    }
}

```

```

        mb.Entity<Developer>().ToTable("Developer");
        mb.Entity<Team>().ToTable("Team");
    }
}

```

Б.2 Текст файла Codesnippet.cs

```

using System;
using System.Collections.Generic;
using System.ComponentModel.DataAnnotations;
using System.ComponentModel.DataAnnotations.Schema;
using System.Linq;
using System.Threading.Tasks;
using CodeSnippetRepository.CodeProcessor;

namespace CodeSnippetRepository.Models
{
    public class Codesnippet
    {
        [Key]

[DatabaseGenerated(DatabaseGeneratedOption.Identity)]
        public int Id
        { get; set; }
        public string Name
        { get; set; }
        public string Code
        {
            get
            {
                return this.Code;
            }
            set
            {
                Code = CodeProcessor.getProcessedCode(value);
            }
        }
        public string ShortDescription
        { get; set; }
        public int CreatorId
        { get; set; }
        public string Description
        { get; set; }
    }
}

```

```

    public DateTime CreationDate
    { get; set; }
    [NotMapped]
    public string CreationDateStr
    {
        get
        {
            return this.CreationDate == null ? "-" :
((DateTime)this.CreationDate).ToString("dd/MM/yyyy");
        }
    }
    public DateTime ModificationDate { get; set; }
    [NotMapped]
    public string ModificationDateStr
    {
        get
        {
            return this.ModificationDate == null ? "-"
: ((DateTime)this.ModificationDate).ToString("dd/MM/yyyy");
        }
    }
    [NotMapped]
    public int LikeCount
    {
        get
        {
            return this.Likes.Count;
        }
    }
    [NotMapped]
    public int CommentCount
    {
        get
        {
            return this.Comments.Count;
        }
    }
    [NotMapped]
    public int SViewCount
    {
        get
        {
            return this.SViews.Count;
        }
    }

```

```

    }
    public int Mode
    { get; set; }
    public int LicenseId
    { get; set; }
    public int LanguageId
    { get; set; }
    public virtual Developer Creator
    { get; set; }
    public virtual License License
    { get; set; }
    public virtual Language Language
    { get; set; }
    public virtual ICollection<Comment> Comments
    { get; set; }
    public virtual ICollection<SView> SViews
    { get; set; }
    public virtual ICollection<Like> Likes
    { get; set; }
    public virtual ICollection<Tag> Tags
    { get; set; }
    public virtual ICollection<CodesnippetConnection>
CodesnippetConnections
    { get; set; }
    public virtual ICollection<SnippetUpdate>
SnippetUpdates
    { get; set; }
}

```

Б.3 Текст файла Developer.cs

```

using System;
using System.Collections.Generic;
using System.ComponentModel.DataAnnotations;
using System.ComponentModel.DataAnnotations.Schema;
using System.Linq;
using System.Threading.Tasks;

namespace CodeSnippetRepository.Models
{
    public class Developer
    {

```

```

        [Key]
[DatabaseGenerated(DatabaseGeneratedOption.Identity)]
        public int Id
        { get; set; }
        public string Login
        { get; set; }
        public string Pass
        { get; set; }
        public string Email
        { get; set; }
        public DateTime ActionDate
        { get; set; }
[NotMapped]
        public string ActionDateStr
        {
            get
            {
                return this.ActionDate == null ? "-" :
((DateTime)this.ActionDate).ToString("dd/MM/yyyy");
            }
        }
        public DateTime CreationDate
        { get; set; }
[NotMapped]
        public string CreationDateStr
        {
            get
            {
                return this.CreationDate == null ? "-" :
((DateTime)this.CreationDate).ToString("dd/MM/yyyy");
            }
        }
        public string Photo
        { get; set; }
        public string Description
        { get; set; }
    }

```

Б.4 Текст файла SnippetUpdate.cs

```

using System;
using System.Collections.Generic;
using System.ComponentModel.DataAnnotations;

```

```

using System.ComponentModel.DataAnnotations.Schema;
using System.Linq;
using System.Threading.Tasks;

namespace CodeSnippetRepository.Models
{
    public class SnippetUpdate
    {
        [Key]
        [DatabaseGenerated(DatabaseGeneratedOption.Identity)]
        public int Id
        { get; set; }
        public int CodesnippetId
        { get; set; }
        public int DeveloperId
        { get; set; }
        public virtual Developer Developer
        { get; set; }
        public virtual Codesnippet Codesnippet
        { get; set; }
        public string Code
        { get; set; }
        public string Version
        { get; set; }
        public DateTime UpdateDate
        { get; set; }
        [NotMapped]
        public string UpdateDateStr
        {
            get
            {
                return this.UpdateDate == null ? "-" :
                ((DateTime)this.UpdateDate).ToString("dd/MM/yyyy");
            }
        }
        public string Description
        { get; set; }
    }
}

```

Б.5 Текст файла Like.cs

```

using System;
using System.Collections.Generic;

```

```

using System.ComponentModel.DataAnnotations;
using System.ComponentModel.DataAnnotations.Schema;
using System.Linq;
using System.Threading.Tasks;

namespace CodeSnippetRepository.Models
{
    public class Like
    {
        [Key]

[DatabaseGenerated(DatabaseGeneratedOption.Identity)]
        public int Id
        { get; set; }
        public int CodesnippetId
        { get; set; }
        public int DeveloperId
        { get; set; }
        public virtual Developer Developer
        { get; set; }
        public virtual Codesnippet Codesnippet
        { get; set; }
        public int Mode
        { get; set; }
        public DateTime CreationDate
        { get; set; }
        [NotMapped]
        public string CreationDateStr
        {
            get
            {
                return this.CreationDate == null ? "-" :
((DateTime)this.CreationDate).ToString("dd/MM/yyyy");
            }
        }
    }
}

```

Б.6 Текст файла Comment.cs

```

using System;
using System.Collections.Generic;
using System.ComponentModel.DataAnnotations;
using System.ComponentModel.DataAnnotations.Schema;

```

```

using System.Linq;
using System.Threading.Tasks;
namespace CodeSnippetRepository.Models
{
    public class Comment
    {
        [Key]

[DatabaseGenerated(DatabaseGeneratedOption.Identity)]
        public int Id
        { get; set; }
        public int CodesnippetId
        { get; set; }
        public int DeveloperId
        { get; set; }
        public virtual Developer Developer
        { get; set; }
        public virtual Codesnippet Codesnippet
        { get; set; }
        public string Text
        { get; set; }
        public DateTime CreationDate
        { get; set; }
        [NotMapped]
        public string CreationDateStr
        {
            get
            {
                return this.CreationDate == null ? "-" :
((DateTime)this.CreationDate).ToString("dd/MM/yyyy");
            }
        }
    }
}

```

Б.7 Текст файла SView.cs

```

using System;
using System.Collections.Generic;
using System.ComponentModel.DataAnnotations;
using System.ComponentModel.DataAnnotations.Schema;
using System.Linq;
using System.Threading.Tasks;
namespace CodeSnippetRepository.Models

```

```

{
    public class Comment
    {
        [Key]

[DatabaseGenerated(DatabaseGeneratedOption.Identity)]
        public int Id
        { get; set; }
        public int CodesnippetId
        { get; set; }
        public int DeveloperId
        { get; set; }
        public virtual Developer Developer
        { get; set; }
        public virtual Codesnippet Codesnippet
        { get; set; }
        public DateTime CreationDate
        { get; set; }
        [NotMapped]
        public string CreationDateStr
        {
            get
            {
                return this.CreationDate == null ? "-" :
((DateTime)this.CreationDate).ToString("dd/MM/yyyy");
            }
        }
    }
}

```

Б.8 Текст файла CodesnippetController.cs

```

using System;
using System.Collections.Generic;
using System.Globalization;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Authorization;
using Microsoft.AspNetCore.Mvc;
using Microsoft.AspNetCore.Mvc.Rendering;
using Microsoft.EntityFrameworkCore;
using CodeSnippetRepository.Data;
using CodeSnippetRepository.Models;
namespace CodesnippetRepository.Controllers

```

```

{
    public class CodesnippetController : Controller
    {
        private readonly DataContext _dbc;

        public CodesnippetController (DataContext dc)
        {
            _dbc = dc;
        }

        [Authorize]
        public async Task<IActionResult> PopularSnippets()
        {
            var snippets = await
            _dbc.Codesnippets.Include(s => s.Creator).Include(s =>
            s.License).Include(s => s.Language).Include(s =>
            s.Team).Include(Comments).Include(Likes).Include(SnippetUpdate).
            Include(SViews).Include(SViewCount).Include(LikeCount).Include(C
            ommentCount).Include(CreationDateStr).Include(ModificationDateSt
            r).Include(CodesnippetConnections).Include(SnippetUpdates);
            snippets = snippets.OrderByDescending(s =>
            s.SViewCount);
            return View(await snippets.ToListAsync());
        }

        [Authorize]
        public async Task<IActionResult> BestSnippets()
        {
            var snippets = await
            _dbc.Codesnippets.Include(s => s.Creator).Include(s =>
            s.License).Include(s => s.Language).Include(s =>
            s.Team).Include(Comments).Include(Likes).Include(SnippetUpdate).
            Include(SViews).Include(SViewCount).Include(LikeCount).Include(C
            ommentCount).Include(CreationDateStr).Include(ModificationDateSt
            r).Include(CodesnippetConnections).Include(SnippetUpdates);
            snippets = snippets.OrderByDescending(s =>
            s.LikeCount);
            return View(await snippets.ToListAsync());
        }

        [Authorize]

```

```

        public async Task<IActionResult> OwnSnippets()
        {
            var snippets = await
            _dbc.Codesnippets.Include(s => s.Creator).Include(s =>
            s.License).Include(s => s.Language).Include(s =>
            s.Team).Include(Comments).Include(Likes).Include(SnippetUpdate).
            Include(SViews).Include(SViewCount).Include(LikeCount).Include(C
            ommentCount).Include(CreationDateStr).Include(ModificationDateSt
            r).Include(CodesnippetConnections).Include(SnippetUpdates);
            var developername = User.Identity.Name;
            IQueryable<Developer> developers =
            _dbc.Developers;
            var developer =
            developers.FirstOrDefaultAsync(d => d.Login == developername);
            snippets = snippets.Where(s => s.CreatorId ==
            developer.Id);
            snippets = snippets.OrderByDescending(a =>
            a.Id);
            return View(await snippets.ToListAsync());
        }

```

```

        [Authorize]
        public async Task<IActionResult> TeamSnippets(int?
        teamid)
        {
            var snippets = await
            _dbc.Codesnippets.Include(s => s.Creator).Include(s =>
            s.License).Include(s => s.Language).Include(s =>
            s.Team).Include(Comments).Include(Likes).Include(SnippetUpdate).
            Include(SViews).Include(SViewCount).Include(LikeCount).Include(C
            ommentCount).Include(CreationDateStr).Include(ModificationDateSt
            r).Include(CodesnippetConnections).Include(SnippetUpdates);
            snippets = snippets.Where(s => s.TeamId ==
            teamid);
            snippets = snippets.OrderByDescending(s =>
            s.Id);
            return View(await snippets.ToListAsync());
        }

```

```

        [Authorize]
        public async Task<IActionResult> TagSnippets(string
        tagname)

```

```

        {
            var snippets = await
_dbc.Codesnippets.Include(s => s.Creator).Include(s =>
s.License).Include(s => s.Language).Include(s =>
s.Team).Include(Comments).Include(Likes).Include(SnippetUpdate).
Include(SViews).Include(SViewCount).Include(LikeCount).Include(C
ommentCount).Include(CreationDateStr).Include(ModificationDateSt
r).Include(CodesnippetConnections).Include(SnippetUpdates);
            var tags = await _dbc.Tags;
            var tag = tags.FirstOrDefaultAsync(t => t.Name
== tagname);

            foreach (var snip in snippets)
            {
                if (!snip.Tags.Contains(tag))
                    snippets.Remove(snip);
            }
            snippets = snippets.OrderByDescending(s =>
s.LikeCount);
            return View(await snippets.ToListAsync());
        }

        [Authorize]
        public async Task<IActionResult>
FindSnippets(string snipname, string language)
        {
            var snippets = await
_dbc.Codesnippets.Include(s => s.Creator).Include(s =>
s.License).Include(s => s.Language).Include(s =>
s.Team).Include(Comments).Include(Likes).Include(SnippetUpdate).
Include(SViews).Include(SViewCount).Include(LikeCount).Include(C
ommentCount).Include(CreationDateStr).Include(ModificationDateSt
r).Include(CodesnippetConnections).Include(SnippetUpdates);
            if (language)
            {
                var languages = await _dbc.Languages;
                var lang = languages.FirstOrDefaultAsync(l
=> l.Name == language);
                snippets = snippets.Where(s => s.LanguageId
== lang.Id);
            }
            if (snipname)
            {

```

```

        snippets = snippets.Where(s =>
s.Name.Contains(snipname));
        snippets = snippets.OrderByDescending(s =>
s.LikeCount);
    }
    return View(await snippets.ToListAsync());
}

```

```

[Authorize]
public async Task<IActionResult>
LanguageSnippets(int? languageId)
{
    var snippets = await
_dbc.Codesnippets.Include(s => s.Creator).Include(s =>
s.License).Include(s => s.Language).Include(s =>
s.Team).Include(Comments).Include(Likes).Include(SnippetUpdate).
Include(SViews).Include(SViewCount).Include(LikeCount).Include(C
ommentCount).Include(CreationDateStr).Include(ModificationDateSt
r).Include(CodesnippetConnections).Include(SnippetUpdates);
    snippets = snippets.Where(s => s.LanguageId ==
languageId);
    snippets = snippets.OrderByDescending(s =>
s.LikeCount);
    return View(await snippets.ToListAsync());
}

```

```

[Authorize]
public async Task<IActionResult> OwnTeamsSnippets()
{
    var snippets = await
_dbc.Codesnippets.Include(s => s.Creator).Include(s =>
s.License).Include(s => s.Language).Include(s =>
s.Team).Include(Comments).Include(Likes).Include(SnippetUpdate).
Include(SViews).Include(SViewCount).Include(LikeCount).Include(C
ommentCount).Include(CreationDateStr).Include(ModificationDateSt
r).Include(CodesnippetConnections).Include(SnippetUpdates);
    var developername = User.Identity.Name;
    IQueryable<Developer> developers =
_dbc.Developers;
    var developer =
developers.FirstOrDefaultAsync(d => d.Login == developername);
    var ownsnippets = new List<Codesnippet>();

```

```

        var teams = await _dbc.Teams;
        foreach (var team in teams)
        {
            if (team.Developers.Contains(developer) ||
)
            {
                var rsnippets =
                _dbc.Codesnippets.Where(s => s.TeamId == team.Id)
                ownsippets.Add();
            }
        }
        ownsippets = snippets.OrderByDescending(s =>
s.Id);

        return View(await ownsippets.ToListAsync());
    }

    [Authorize]
    public async Task<IActionResult> Details(int?
snippetId)
    {
        if (snippetId == null)
        {
            return NotFound();
        }
        var snippet = await _dbc.Codesnippets.Include(s
=> s.Creator).Include(s => s.License).Include(s =>
s.Language).Include(s =>
s.Team).Include(Comments).Include(Likes).Include(SnippetUpdate).
Include(SViews).Include(SViewCount).Include(LikeCount).Include(C
ommentCount).Include(CreationDateStr).Include(ModificationDateSt
r).Include(CodesnippetConnections).Include(SnippetUpdates).First
OrDefaultAsync(t => t.Id == snippetId);
        if (snippet == null)
        {
            return NotFound();
        }

        return View(snippet);
    }

    [Authorize]

```

```

        public IActionResult Create()
        {
            ViewData["Teams"] = new SelectList(_dbc.Teams,
            "Id", "Name");
            ViewData["Languages"] = new
            SelectList(_dbc.Languages, "Id", "Name");
            ViewData["Licenses"] = new
            SelectList(_dbc.Licenses, "Id", "Name", "Description");
            ViewData["CodesnippetConnections"] = new
            SelectList(_dbc.Codesnippets, "Id", "Name");
            return View();
        }

        [HttpPost]
        [ValidateAntiForgeryToken]
        public async Task<IActionResult> Create([Bind("Id,
            Name, ShortDescription, Description, Mode, LicenseId,
            LanguageId")] Codesnippet codesnippet)
        {
            if (ModelState.IsValid)
            {
                IQueryable<Developer> developers =
                _dbc.Developers;
                var developername = User.Identity.Name;
                var dev = developers.FirstOrDefault(d
                => d.Login == developername);
                codesnippet.CreatorId = dev.Id;
                codesnippet.ModificationDate =
                DateTime.Now;
                codesnippet.CreationDate = DateTime.Now;
                _dbc.Add(codesnippet);
                await _dbc.SaveChangesAsync();
                return RedirectToAction(nameof(Index));
            }
            return View(codesnippet);
        }
    }
}

```

ДОДАТОК В
Слайди презентації

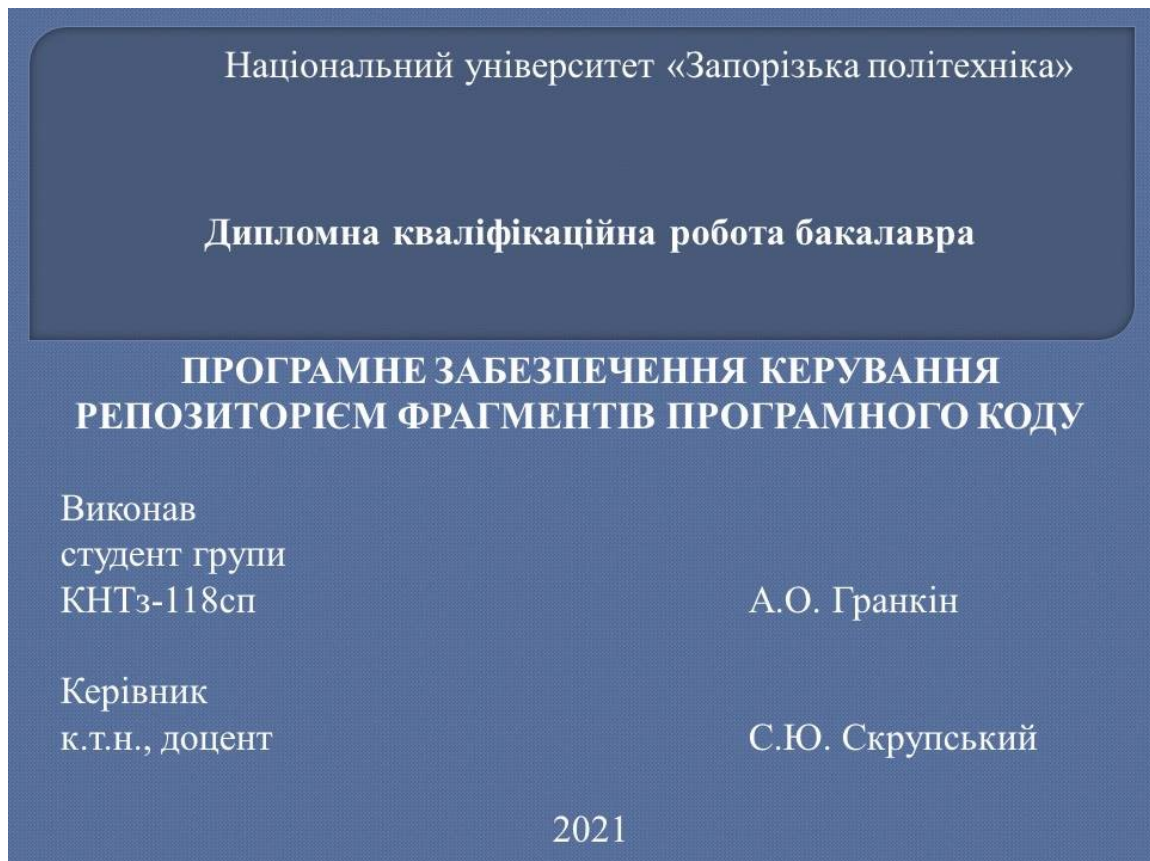


Рисунок В.1 – Слайд № 1

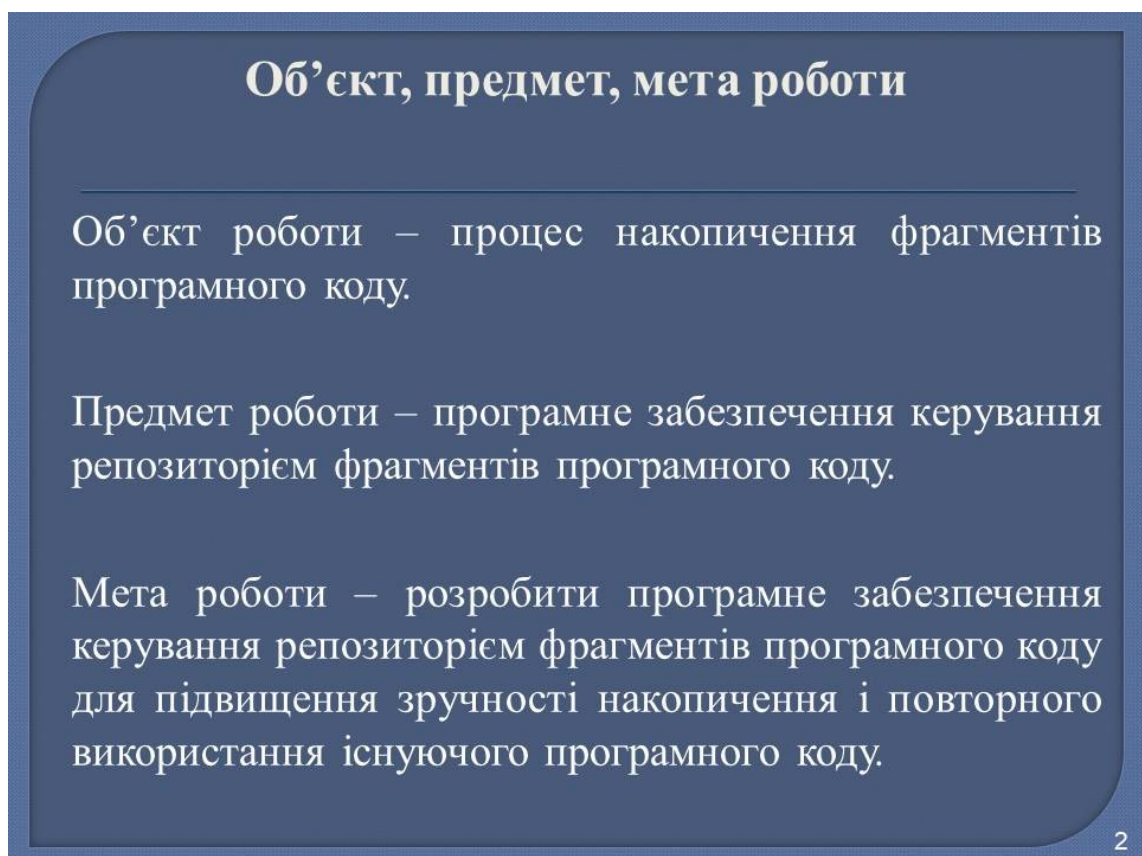


Рисунок В.2 – Слайд № 2

Завдання роботи

- ✓ аналіз предметної області;
- ✓ виділення вимог до програми;
- ✓ проєктування програми;
- ✓ розробка з налагодженням програми;
- ✓ тестування і налагодження програми.

3

Рисунок В.3 – Слайд № 3

Аналіз предметної області

Показник	Snipplr	Djangosnippets
Створення власних фрагментів	+	+
Велика кількість засобів перегляду і пошуку фрагментів	+	+
Створення закритих фрагментів	+	-
Мови програмування	Усі широковідомі	HTML, JavaScript, Python, Shell, SQL
Засоби роботи в команді	-	-
Статистика фрагментів у динаміці	-	-

4

Рисунок В.4 – Слайд № 4



Рисунок В.5 – Слайд № 5

Порівняння фреймворків веброзробки

Показник	ASP.NET	Laravel
Мова програмування	C#	PHP
Підтримка великих проєктів	++	+-
Безпека	++	+-
Підтримка спільноти	Якісна за рахунок Microsoft	Якісна і широка за рахунок вільної спільноти розробників

Рисунок В.6 – Слайд № 6

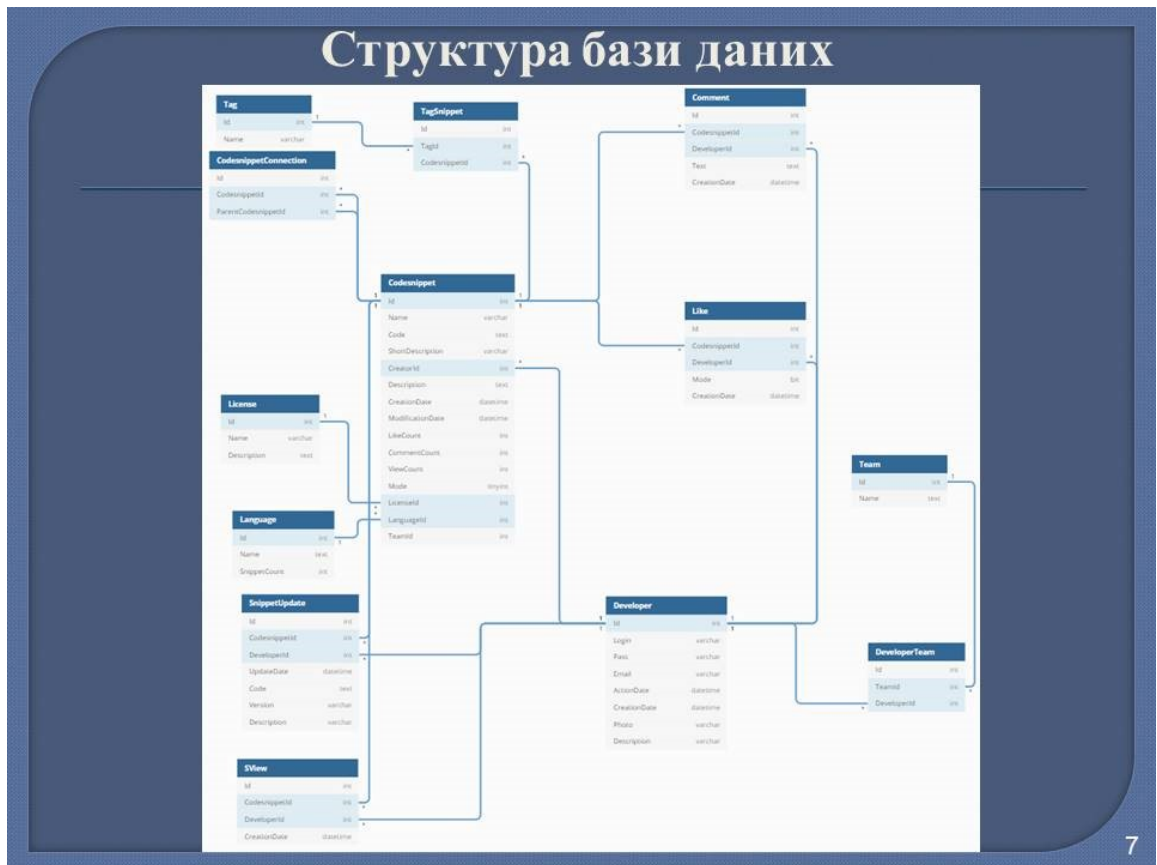


Рисунок В.7 – Слайд № 7



Рисунок В.8 – Слайд № 8



Рисунок В.9 – Слайд № 9

Перегляд фрагментів програмного коду

РЕПОЗИТОРІЙ ФРАГМЕНТІВ ПРОГРАМНОГО КОДУ

Пошук

ФРАГМЕНТИ

ТЕГИ

МОВИ

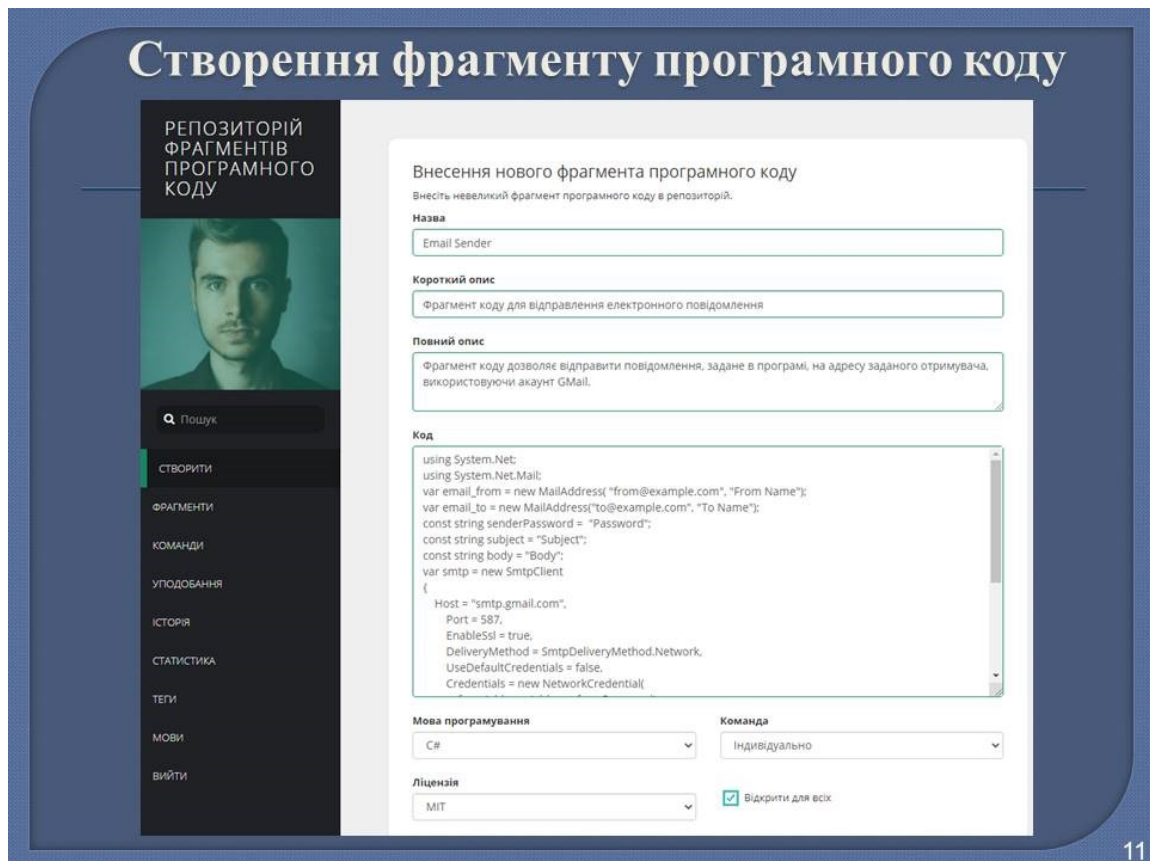
АВТОРИЗАЦІЯ

ПОПУЛЯРНІ | НАЙКРАЩІ

Фрагменти з найбільшою кількістю уподобань

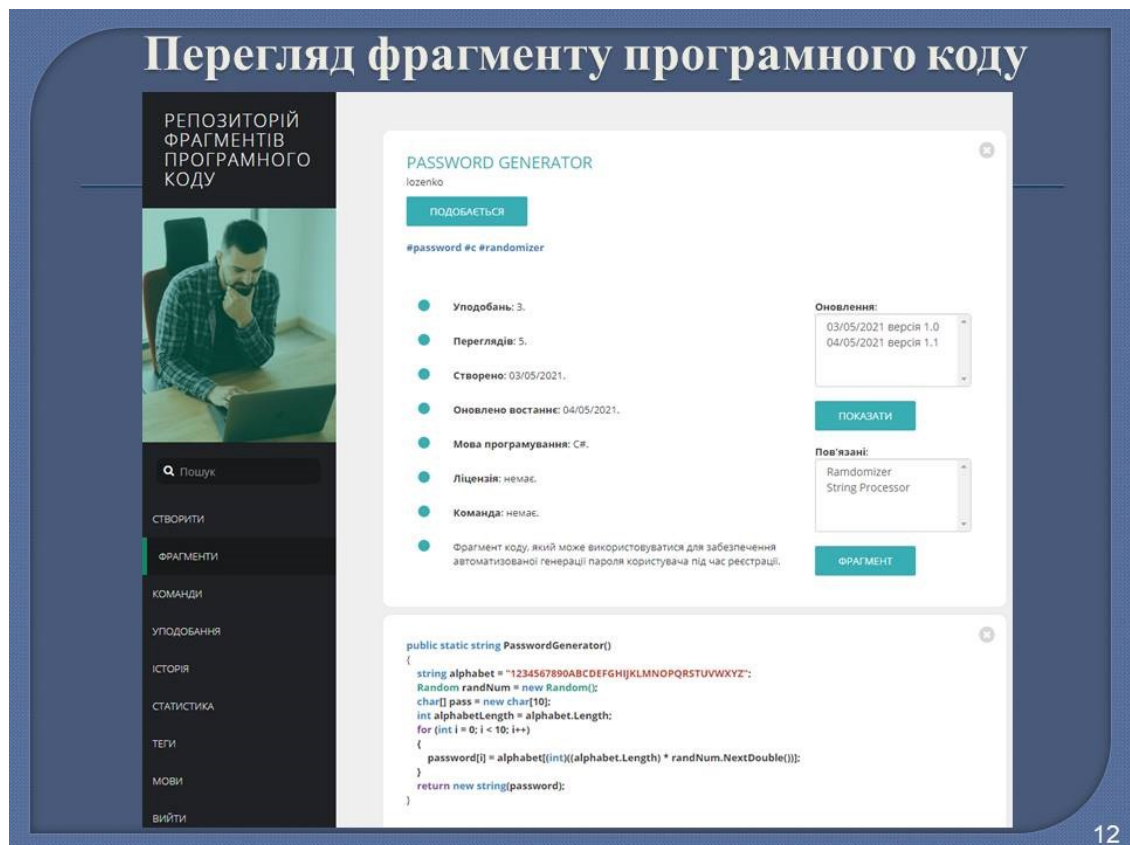
Назва	Автор	Оновлення	Мова	♥	➦	📄
Код створення форми з кнопкою для завантаження файлу на сервер						
File Upload Form	bestdeveloper	28/04/2021	PHP	12	14	2
Скрипт для створення CAPTCHA-зображення у вигляді розв'язання логічної задачі						
CAPTCHA creator	heroku	29/04/2021	PHP	11	15	0
Реалізація алгоритму швидкого сортування						
Quick Sort	chucha	01/05/2021	Java	11	12	3
Код для визначення розташування, включаючи місто і країну або штат, за IP-адресою						
Location detector	developer81	28/04/2021	PHP	11	11	0
Код для генерації дерева за заданим рядком						
Tree Generator	bestdeveloper	28/04/2021	C++	10	10	1
Фрагмент для підрахунку кількості входжень символів у рядок						
Symbol Calculator	sanalberto	30/04/2021	C++	9	10	0
Універсальний парсер дат з конвертором						
Date Parser	bestdeveloper	01/05/2021	Java	9	10	1
Код оптимізованого виконання конкатенації двох і більше рядків у один						
String Concatenation	sanalberto	28/04/2021	C	8	9	0
Код для генерації XML-файлу з заданого шаблону						
XML Creator	ruddi	02/05/2021	PHP	8	8	2
Код для автоматичного відправлення повідомлень в Twitter за заданим у текстовому файлі шаблоном						
Twitter Poster	luka	02/05/2021	Python	8	8	1

Рисунок В.10 – Слайд № 10



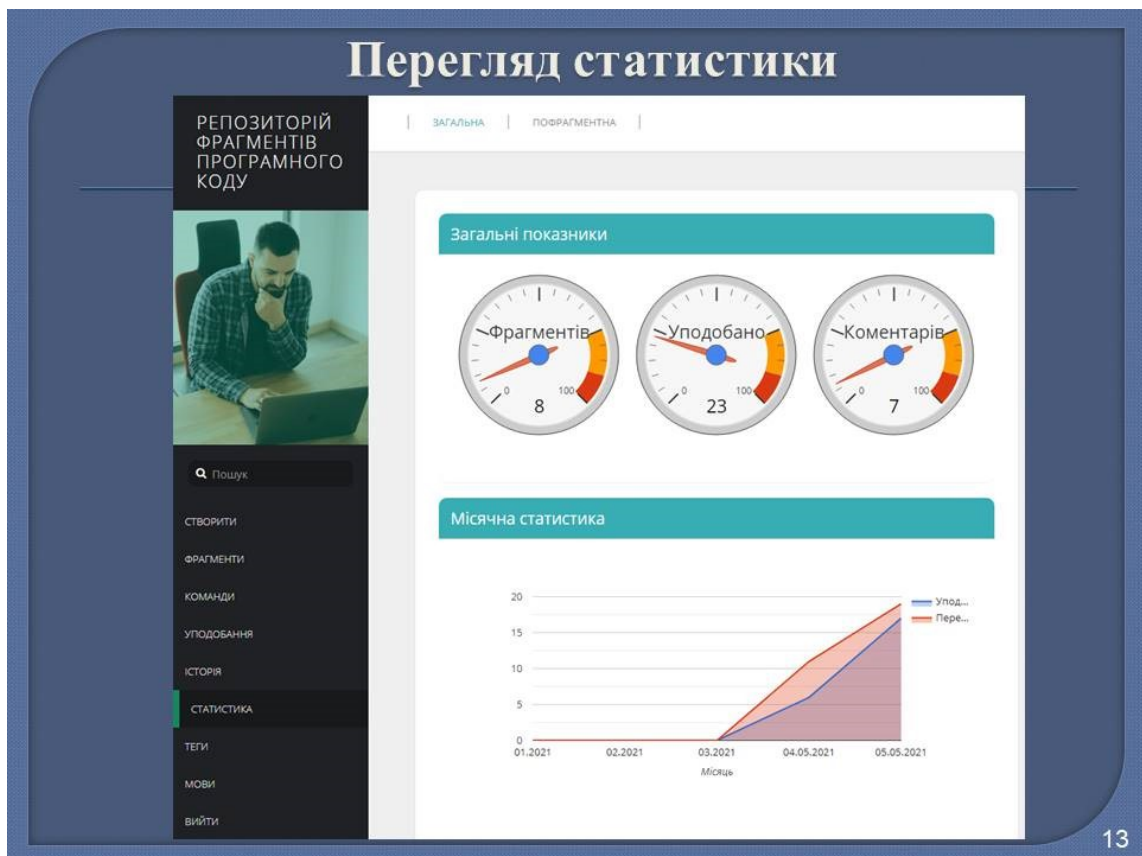
11

Рисунок В.11 – Слайд № 11



12

Рисунок В.12 – Слайд № 12



13

Рисунок В.13 – Слайд № 13

Висновки

У результаті проведеного аналізу предметної області було визначено поняття фрагментів програмного коду, особливості їх використання, основні ліцензії програмного забезпечення, виконано огляд програмних аналогів, включаючи Snipplr, Djangosnippets, для них визначено їх функціональні можливості, переваги і недоліки та виконано порівняння зі створенням відповідної таблиці порівняння. У якості основного програмного аналогу для розробки було обрано Snipplr. Основною відмінністю створеного програмного забезпечення порівняно зі Snipplr є можливість роботи в команді над фрагментами програмного коду та надання статистики про фрагменти програмного коду в динаміці.

Виконана розробка є вебзастосунком, що призначений для підтримки роботи з репозиторієм фрагментів програмного коду та забезпечення доступу до них індивідуальним розробникам і командам розробників.

Виконане проектування програмного забезпечення дозволило прийняти рішення про використання мови програмування C#, фреймворку розробки вебзастосунків ASP.NET, системи керування базами даних Microsoft SQL Server, об'єктно-реляційного відображення Entity Framework, визначити архітектуру програми на основі архітектурного шаблону MVC, виділити функціональні вимоги до програми, визначити структуру бази даних.

Усі завдання дипломної кваліфікаційної роботи повністю виконано.

14

Рисунок В.14 – Слайд № 14