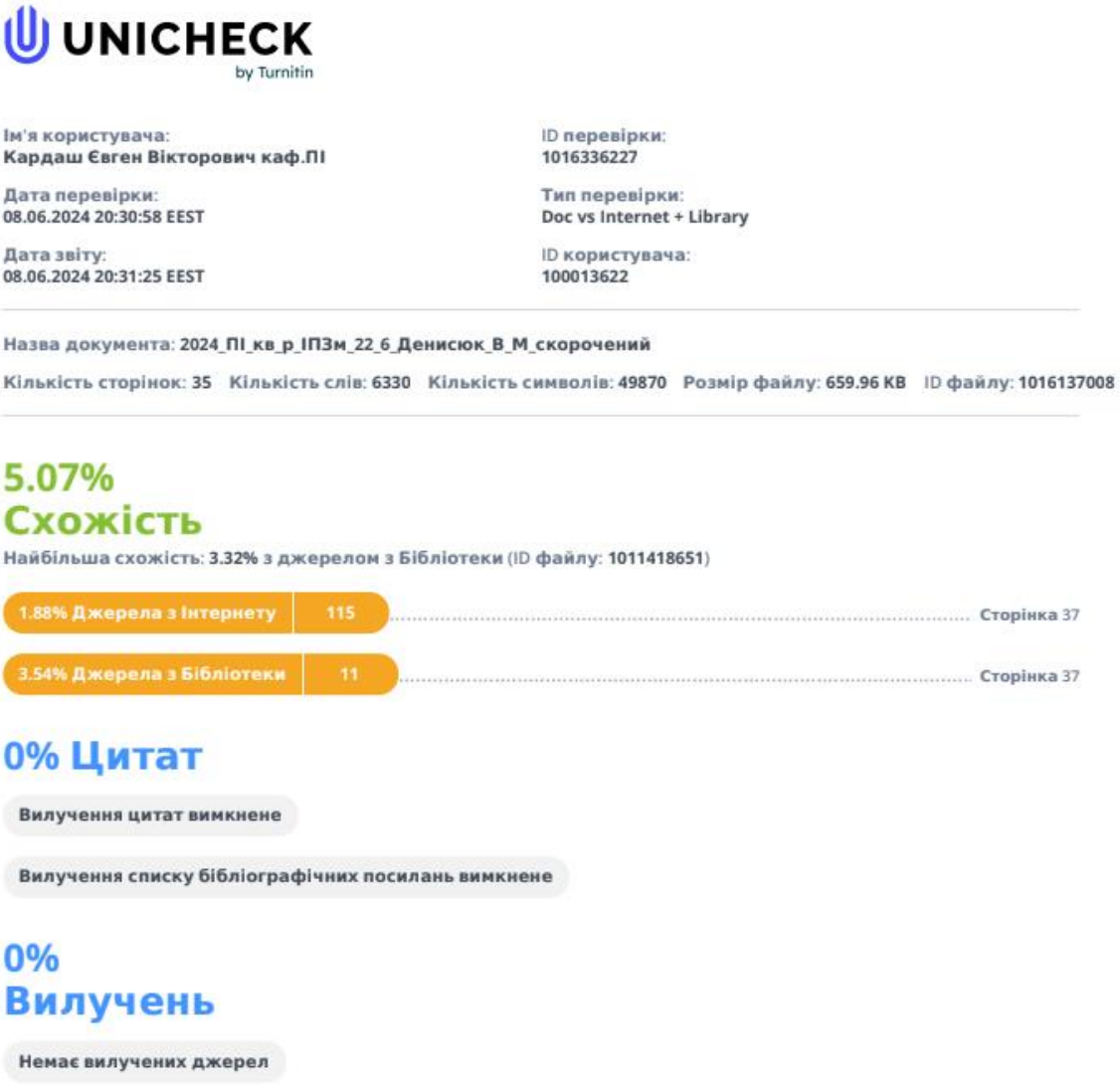
ДОДАТОК А

Результат перевірки на антиплагіат



UNICHECK
by Turnitin

Ім'я користувача:
Кардаш Євген Вікторович каф.ПІ

ID перевірки:
1016336227

Дата перевірки:
08.06.2024 20:30:58 EEST

Тип перевірки:
Doc vs Internet + Library

Дата звіту:
08.06.2024 20:31:25 EEST

ID користувача:
100013622

Назва документа: 2024_ПІ_кв_р_ІПЗм_22_6_Денисюк_В_М_скорочений

Кількість сторінок: 35 Кількість слів: 6330 Кількість символів: 49870 Розмір файлу: 659.96 KB ID файлу: 1016137008

5.07%
Схожість

Найбільша схожість: 3.32% з джерелом з Бібліотеки (ID файлу: 1011418651)

1.88% Джерела з Інтернету 115 ..... Сторінка 37

3.54% Джерела з Бібліотеки 11 ..... Сторінка 37

0% Цитат

Вилучення цитат вимкнене

Вилучення списку бібліографічних посилань вимкнене

0%
Вилучень

Немає вилучених джерел

ДОДАТОК Б

Слайди презентації

Ефективність і практичність інструментів для генерації коду в програмах .NET з використанням графічних представлень

- Харківський Національний Університет Радіоелектроніки
- Виконав: Денисюк Владислав Михайлович ІПЗМ-22-6
- Керівник: Проф. каф. ПІ Четвериков Г.Г

1

Актуальність

- Економія ресурсів компанії
- Актуалізація архітектурних рішень ПЗ
- Візуалізація процесів розробницької діяльності
- Використання генеративних підходів у бізнесі

2

Мета досліждження

- **Аналіз та порівняння інструментів для генерації коду:** Зосередження на автоматичній генерації коду для полегшення розробки додатків .NET за допомогою графічних елементів інтерфейсу.
- **Оцінка ефективності та практичності:** Оцінка різних інструментів на основі продуктивності, зручності використання та якості коду.
- **Розробка стислого програмного забезпечення реалізуючи два підходи**: процедурний та нейромережевий

3

Аналіз предметної області

Кодогенерація – процес при якому одна програмна система генерує вихідний код іншої.

Наявні приклади використання кодогенерації:

- Nexxen – компанія, яка використовує генеративне рішення для вбудови реклами у браузерні сторінки
- Lab digital – компанія яка використовує генерацію вихідного коду на етапі прототипування
- EF CORE – компанія Microsoft, використовує генерацію SQL запитів

nexxen

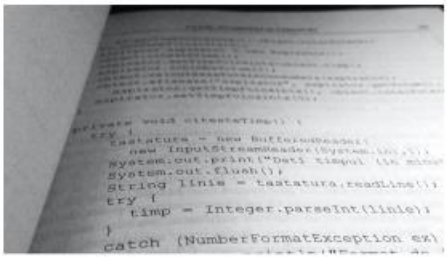
Entity Framework Core

lab digital next level digital

4

Огляд

- **Визначення та важливість генерації коду:** Генерація коду спрощує процес розробки програмного забезпечення шляхом автоматизації створення вихідного коду.
- **Огляд платформи .NET:** .NET є універсальною та потужною платформою, що використовується для розробки масштабованих та ефективних додатків.



5

Інструменти генерації вихідного коду у .NET

- **T4 (Text Template Transformation Toolkit):** Потужний інструмент генерації коду на основі шаблонів, інтегрований з Visual Studio
- **Razor:** Синтаксис, що використовується для вбудовування серверного коду в веб-сторінки, переважно в ASP.NET.
- **Roslyn API:** Набір компіляторів та API для аналізу коду з відкритим кодом для C# та VB.NET.

Photo by Michael Krnac on Unsplash

6

Порівняння T4 та Razor

- **T4 (Text Template Transformation Toolkit):** Гнучкий та універсальний для різних завдань генерації коду, інтегрований з Visual Studio, але може стати складним з великими шаблонами.
- **Razor:** Ідеальний для веб-розробки, легка інтеграція з ASP.NET, але переважно зосереджений на HTML та генерації веб-сторінок.
- У нашому випадку для генерації вихідного коду буде використано Roslyn API

T4

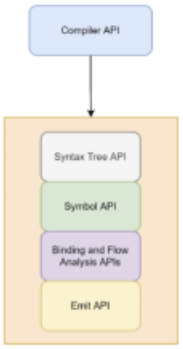
@razor engine

Photo by JC Falcon on Unsplash

7

Використання Roslyn API для генерації коду

- **Вступ до Roslyn API:** Платформа компілятора з відкритим кодом для .NET, що надає багаті API для аналізу коду.
- **Можливості Roslyn API:** Дозволяє розробникам програмно читати, змінювати та писати код C# і VB.NET.

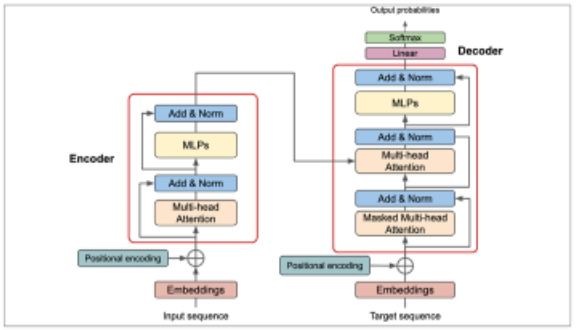


Приклад процесу генерації коду використовуючи Roslyn API

8

Нейромережа Transformer

- **Огляд архітектури Transformer:** Введені в 2017 році, Transformers революціонізували завдання, пов'язані з послідовними даними, такі як переклад мов та генерація тексту.
- **Застосування у генерації коду:** Transformers здатні розуміти складні структури та взаємозв'язки в даних, що робить їх ідеальними для генерації коду з UML-діаграм.

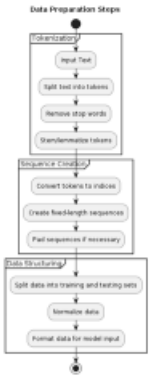


архітектура нейромережі Transformer

9

Підготовка даних

- **Кроки підготовки даних:** Включають токенізацію, створення послідовностей та структурування даних для навчання та тестування



Процес підготовки даних

10

Платформа для проведення дослідження

- **Вибір платформи .NET:** .NET обрана за свою гнучкість, надійність та розширену підтримку бібліотек.
- **Обґрунтування вибору:** Підтримує інтеграцію з різними технологіями, підходить для клієнтських та серверних компонентів.

Microsoft .NET

11

Методології та приготування тесту

- Методології: процедурне генерування та нейромережеве
- Конфігурація: .NET 6.0. Roslyn API, GPT 3.0.

- Випадок 1: Простий клас з властивістю;
- Випадок 2: Клас з кількома властивостями;
- Випадок 3: Клас з методами;
- Випадок 4: Складний клас з властивостями та методами;
- Випадок 5: Великий клас з кількома методами та властивостями.

12

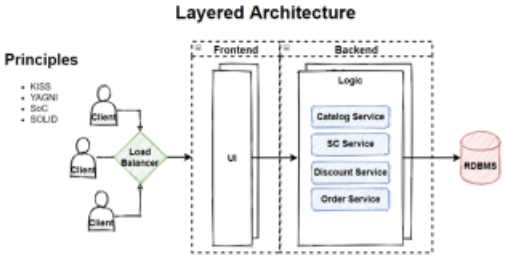
UML моделювання у розробці ПЗ

- Важливість UML у створенні коду: Уніфікована мова моделювання (UML) має вирішальне значення для візуалізації дизайну та архітектури програмних систем.
- Приклади діаграм UML: включає діаграми класів, діаграми послідовності та діаграми варіантів використання для моделювання різних аспектів програмного забезпечення.

Приклад діаграми класів

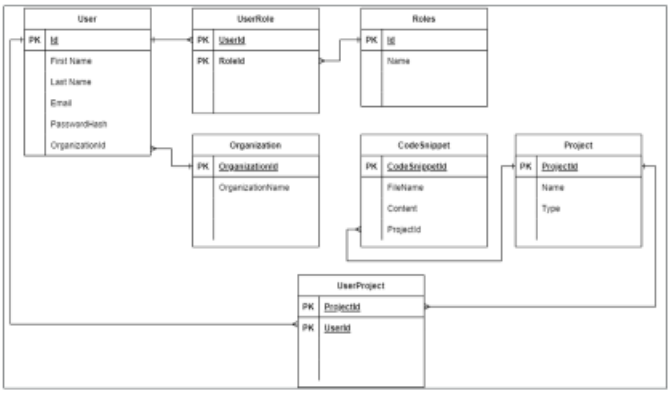
N-layer Архітектура

- Огляд N-рівневої архітектури: шаблон проектування, який розділяє програму на логічні рівні для підвищення модульності та зручності обслуговування.
- Компоненти N-рівневої архітектури: включає рівень презентації, рівень бізнес-логіки, рівень доступу до даних і рівень домену.



Візуальне представлення багатошарової архітектури

Проектування бази даних



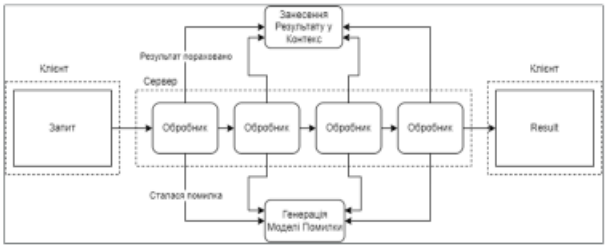
15

Процедурне генерування

Приклади оброблювачів:

- Валідація,
- Керування залежностями
- Зчитування даних класу
- Створення коду за допомогою Roslyn API.

Chain of Responsibility Pattern: Використовується для обробки послідовності етапів обробки, що забезпечує гнучку та розширювану логіку створення коду.



16

Результати бенчмарку

Case	Roslyn T(ms)	Neural T(ms)	Roslyn Memory(mb)	Neural Memory(mb)	Code Quality (Roslyn)	Code Quality (Neural)
Case 1	51.3	151.1	30	200	High	High
Case 2	52.2	152.5	32	210	High	Medium-High
Case 3	61.3	178.2	35	220	High	Medium
Case 4	79.3	201.2	40	250	High	Medium
Case 5	91.2	301.1	50	300	High	Mediun

Оцінка якості коду проводилася використовуючи статичний аналізатор коду від Jenkins SonarQube

17

Analysis of Findings

Roslyn API:
Плюси:
- Швидший,
- Більш ефективний з точки зору пам'яті, генерує високоякісний код.

Мінуси:
- Вимагає знань про Roslyn API та ручного налаштування для різних випадків,
- Неможливо генерувати тіло методів без передчасного втручання у код, або неправильної нотації у діаграмі

Нейронна мережа:
Плюси:
- Легше використовувати для тих, хто не знайомий з Roslyn API,
- більш гнучкий у роботі з різними запитами.
- Можливість генерувати приблизний код методів

Мінуси:
- Повільніший,
- більш пам'яттєвмісний,
- Дещо нижча якість коду.

18

Висновки

- Проведено аналіз предметної галузі
- Проведено дослідження методів генерації вихідного коду
- Запропоновано підхід перетворення візуальної репрезентації у код
- Обрано програмні засоби та технологічні підходи, які задовольняють генеративні підходи
- Проведено порівняльний аналіз процедурного генерування та нейромережевого
- Реалізовано MVP програмної системи для генерації програмного коду

19

ДОДАТОК В

Апробація результатів роботи

Матеріали XXVIII Міжнародного Молодіжного Форуму «Радіоелектроніка та молодь у XXI столітті» 2024 р. том 6 конференція «Інформаційні Інтелектуальні Системи» Information Intelligent Systems с.336-337

ДОДАТОК Г

Експертний висновок результатів перевірки кваліфікаційної роботи на відповідність оформлення вимогам ДСТУ 3008:2015

Експертний висновок результатів перевірки кваліфікаційної роботи

студент (посада) | програмної інженерії (кафедра) | ІПЗм-22-6 (група)

Денисюк Владислав Михайлович
(прізвище, ім'я, по батькові)

Зауваження

Пункт ДСТУ 3008-2015	Зміст пункту	Сторінка кваліфікаційної роботи
1	2	3
	7.1 Загальні положення	
	7.3 Нумерація сторінок звіту	
	7.4 Нумерація розділів, підрозділів, пунктів, підпунктів	
	7.5 Рисунки	
	7.6 Таблиці	
	7.7 Переліки	
	7.8 Примітки	
	7.9 Виноски	
	7.10 Формули та рівняння	
	7.11 Посилання	
	7.13 Список авторів	
	7.14 Скорочення та умовні познаки	
	7.15 Додатки	
Методичні вказівки до виконання кваліфікаційної роботи магістра… ЗАТВЕРДЖЕНО кафедрою ПІ протокол №10 11.01.2022	Відсутній додаток Перелік джерел посилання за науковими напрямами керівника та науковців кафедри програмної інженерії	стор.44

Експерт _____ (підпис) Олена ОЛІЙНИК (прізвище, ініціали)

09.06.2024