ДОДАТОК А

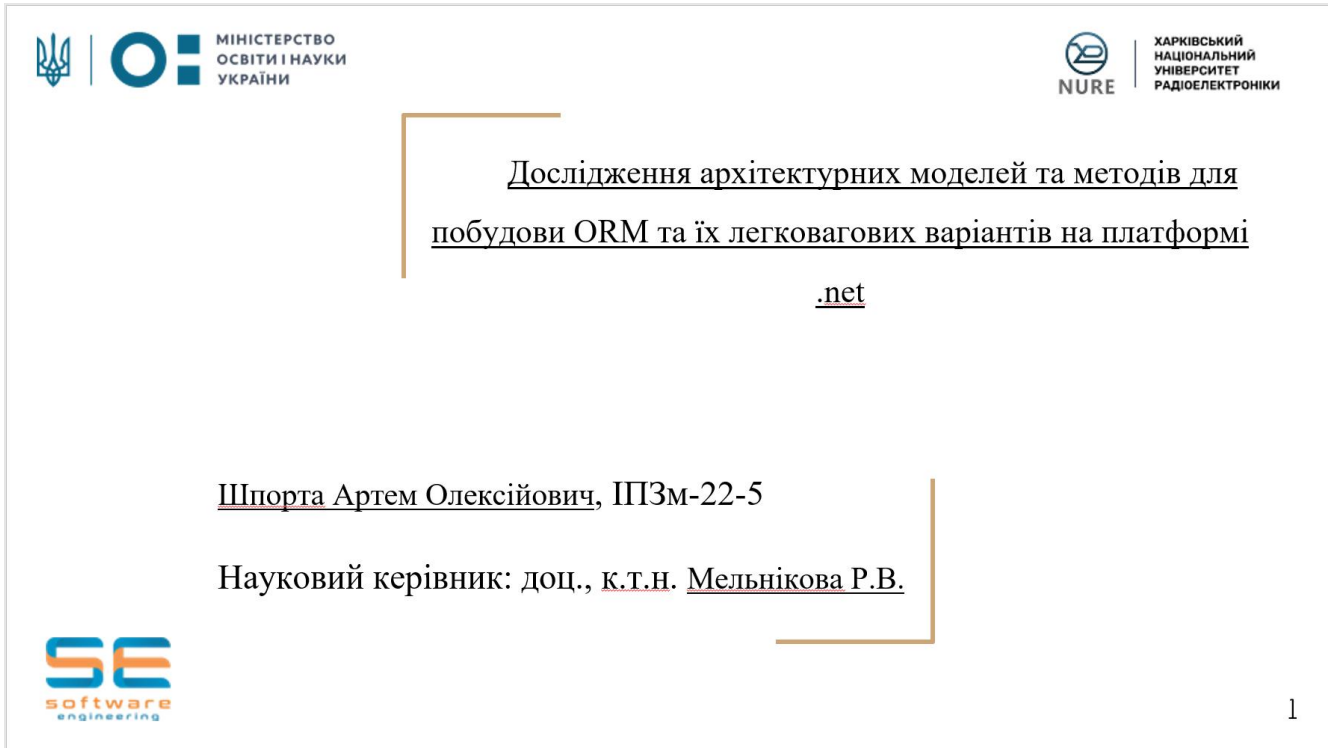
Перелік джерел посилання за науковими напрямами керівника та науковців кафедри програмної інженерії

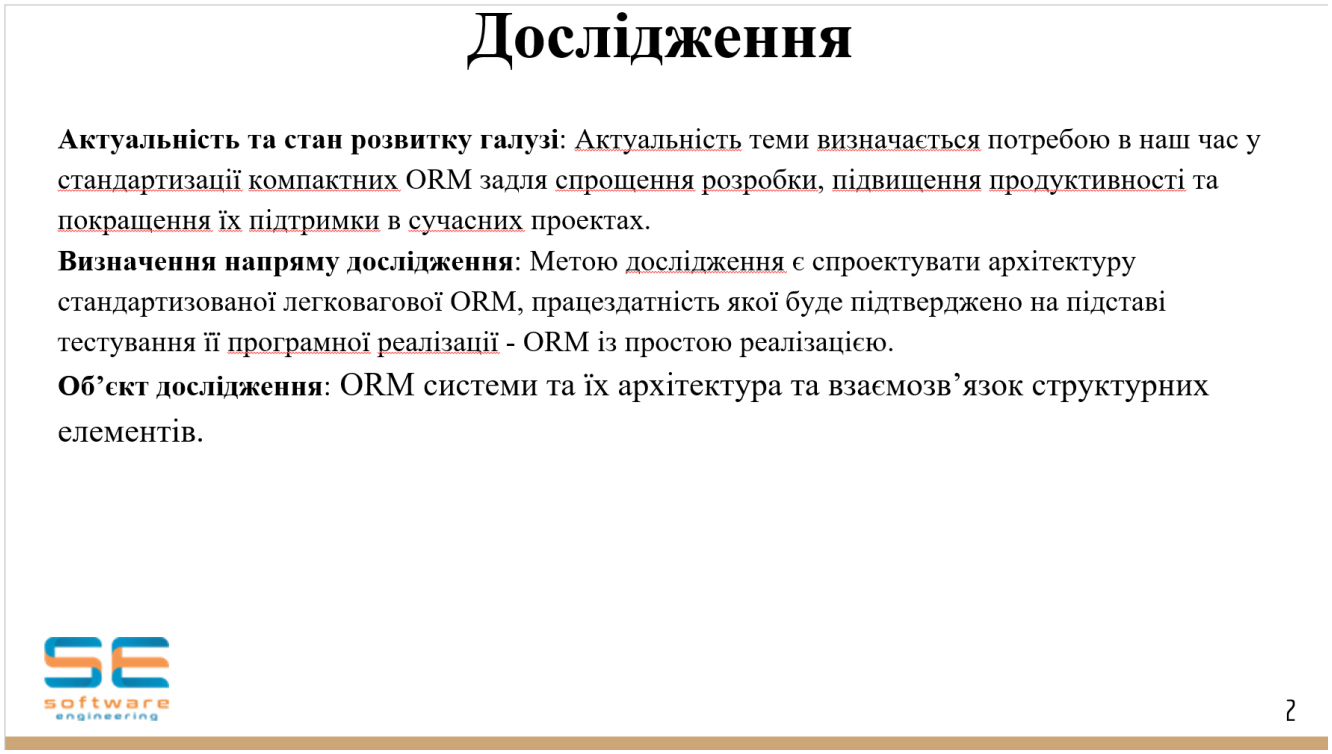
2. Шпорта А.О., Мельнікова Р.В. Дослідження архітектурних моделей та методів для побудови ORM та їх компактних варіантів на платформі .net. 28-й Міжнародний молодіжний форум «Радіоелектроніка та молодь у XXI столітті». Зб. матеріалів форуму. Т. 6., - Харків: ХНУРЕ. 2024. 529-531 с.

8. Falatiuk H., Shirokopetleva M., Dudar Z. Investigation of Architecture and Technology Stack for e-Archive System. 2019 IEEE International Scientific-Practical Conference Problems of Infocommunications, Science and Technology (PIC S&T), Kyiv,Ukraine,8-11October2019.2019.,URL: https://doi.org/10.1109/picst47496.2019.9061407 (дата звернення: 05.05.2024).

ДОДАТОК Б

Слайди презентації





Постановка задачі

- Провести порівняльний аналіз та аналіз призначення кожного архітектурного елементу обраних ORM, створити загальну архітектурну модель, яка буде відображати ідеальну або рекомендовану основу для створення компактних ORM.
- Робота спрямована на виявлення закономірностей в архітектурах сучасних ORM, на основі закономірностей буде сформовано тенденційні патерни для компактних варіантів.
- Реалізація наукового дослідження складається з наступних етапів:
  - аналіз предметної області;
  - аналіз архітектурних особливостей сучасних ORM;
  - знаходження закономірностей та логічних висновків щодо ідеальної архітектури компактних ORM;
  - виведену архітектуру буде проаналізовано та на її основі створено програмний застосунок для її тестування.

Методологія

Опис використаних методів дослідження: Було проведено порівняльний аналіз широковикористовуваних ORM та аналізувались патерни та їх результати, також був задіяний персональний професійний досвід у використанні та модернізації подібних ORM-архітектур.

Інструментарій та технології, використані в роботі:

- Платформи: .Net
- Мова програмування C#

Актуальність обраної тематики

- На сьогоднішній день не існує конкретних стандартних інструкцій або загальної архітектурної моделі для розробки компактних ORM. Кожна бібліотека або фреймворк, які надають такий функціонал, може мати свою власну архітектурну концепцію та інтерфейс.
- Це створює необхідність розробити загальний підхід до архітектури компактних ORM. Дана необхідність зумовлена наступними факторами:

  - відсутність загальних стандартів та рекомендацій з архітектури компактних ORM може призвести до великої різноманітності підходів, що ускладнює вибір правильної бібліотеки або фреймворку.

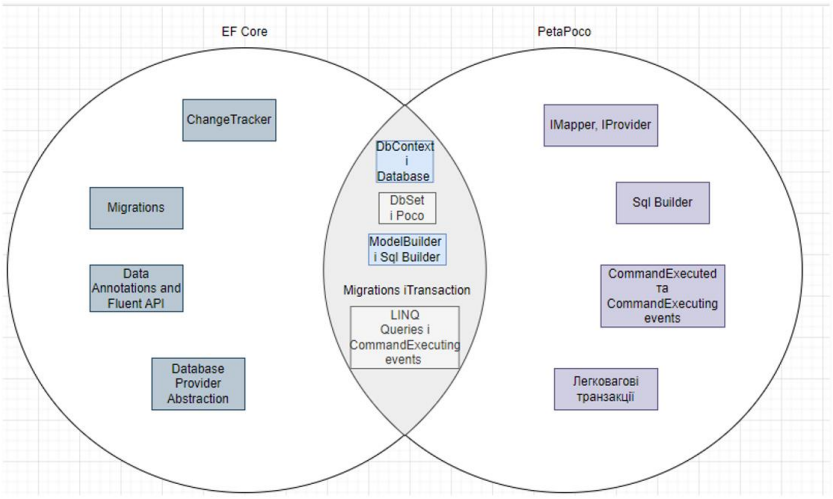
  - стандартизована архітектура може полегшити підтримку і розширення існуючих компактних ORM.

Огляд існуючих рішень

Було розглянуто:

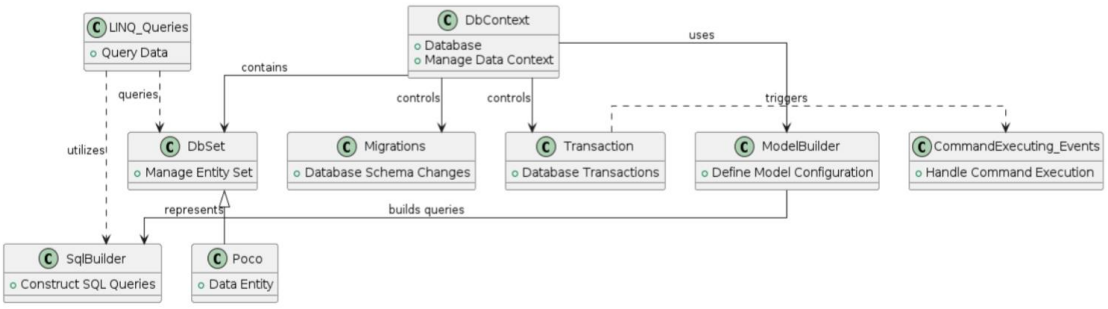
- архітектури Java ORM систем
- архітектурний підхід Hibernate
- архітектурний підхід ORMLite
- архітектурний підхід EF Core
- архітектурний підхід PetaPoco

Схема перетинань ORM PetaPoco та EF Core



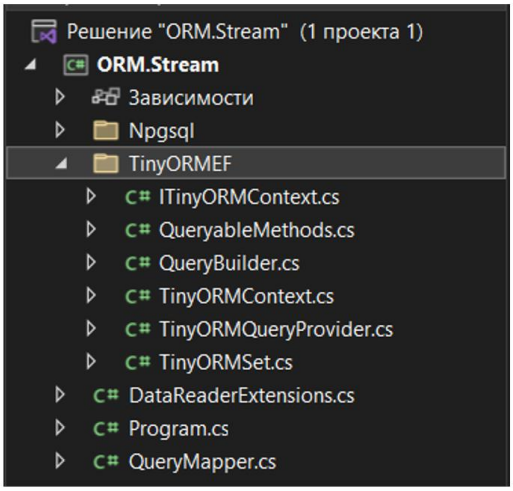
7

Спроєктована архітектура ORM



8

Файлова структура TinyORM



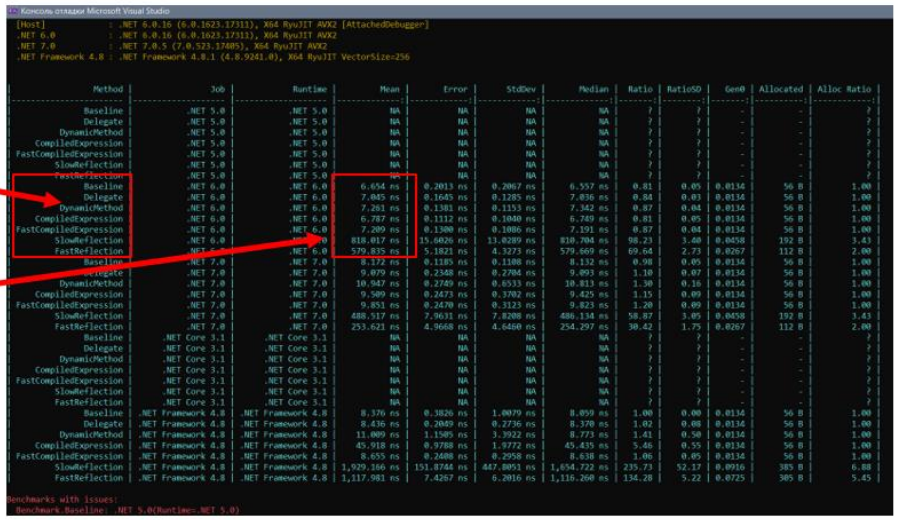
Структурні елементи ORM та їх реалізація

Фундаментальна Функція	Опис	Відповідні Класи/Методи
Управління контекстом бази даних	Центральний контекст для взаємодії з базою даних, створення наборів даних та управління з'єднаннями.	TinyORMContext - TinyORMContext(TinyORMConnection connection) - CreateSet(Type) - ResolveTableName(Type) - QueryAsync<TResult>(FormattableString)
Створення запитів	Динамічне створення запитів на основі LINQ-виразів.	TinyORMQueryProvider - CreateQuery(Expression) - CreateQuery<TElement>(Expression)
Виконання запитів	Виконання запитів та повернення результатів.	TinyORMQueryProvider - Execute(Expression) - Execute<TResult>(Expression) TinyORMContext - QueryAsync<TResult>(FormattableString)
Побудова SQL-запитів	Перетворення дерев виразів у SQL-запити.	QueryBuilder - BuildQuery(Expression) SqlExpressionVisitor - Visit(Expression)
Маппінг результатів запитів	Маппінг рядків бази даних на об'єкти .NET.	DataReaderExtensions - MapTo<T>(IDataReader) QueryMapper - QueryAsync<T>(IDbConnection, string, object) - QueryAsyncType(IDbConnection, FormattableString, Type, object)
Підтримка LINQ	Інтеграція з LINQ для формування запитів.	TinyORMQueryable<T> - Expression - Provider TinyORMQueryProvider - CreateQuery(Expression) - CreateQuery<TElement>(Expression)
Створення наборів даних (Sets)	Створення і управління колекціями об'єктів (сетов) для операцій CRUD.	TinyORMSet<T> - Add(T) - Remove(T) - Find(object) TinyORMContext - CreateSet(Type) - CreateSetInternal<T>()

Порівняння мапінг методів

Тестування підходів до маппінгу.
У процесі підготовки до тесту нами було реалізовано:

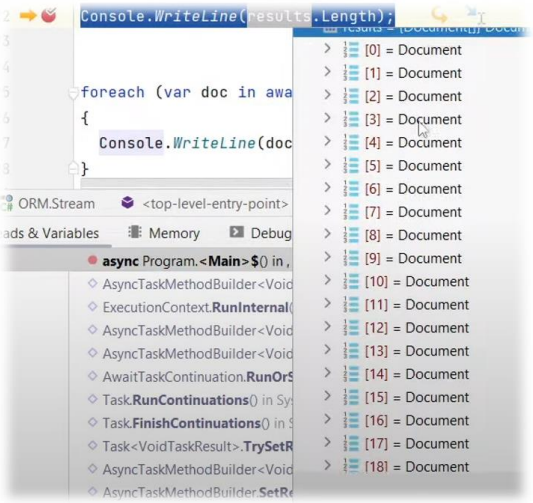
- Розроблено тестові моделі;
- Використано атрибути BenchmarkDotNet;
- Встановлено метрики;
- Проаналізовано результати.



Method	Job	Runtime	Mean	Error	StdDev	Median	Ratio	RatioSD	Gen0	Allocated	Alloc Ratio
Baseline	.NET 5.0	.NET 5.0	NA	NA	NA	NA	?	?	-	-	?
Delegate	.NET 5.0	.NET 5.0	NA	NA	NA	NA	?	?	-	-	?
DynamicMethod	.NET 5.0	.NET 5.0	NA	NA	NA	NA	?	?	-	-	?
CompiledExpression	.NET 5.0	.NET 5.0	NA	NA	NA	NA	?	?	-	-	?
FastCompiledExpression	.NET 5.0	.NET 5.0	NA	NA	NA	NA	?	?	-	-	?
SlowReflection	.NET 5.0	.NET 5.0	NA	NA	NA	NA	?	?	-	-	?
FastReflection	.NET 5.0	.NET 5.0	NA	NA	NA	NA	?	?	-	-	?
Baseline	.NET 6.0	.NET 6.0	6.654 ns	0.2013 ns	0.2067 ns	6.557 ns	0.81	0.05	0.0134	56 B	1.00
Delegate	.NET 6.0	.NET 6.0	7.045 ns	0.1645 ns	0.1285 ns	7.036 ns	0.84	0.03	0.0134	56 B	1.00
DynamicMethod	.NET 6.0	.NET 6.0	7.281 ns	0.1381 ns	0.1153 ns	7.342 ns	0.87	0.04	0.0134	56 B	1.00
CompiledExpression	.NET 6.0	.NET 6.0	6.787 ns	0.1112 ns	0.1040 ns	6.749 ns	0.81	0.05	0.0134	56 B	1.00
FastCompiledExpression	.NET 6.0	.NET 6.0	7.209 ns	0.1300 ns	0.1086 ns	7.191 ns	0.87	0.04	0.0134	56 B	1.00
SlowReflection	.NET 6.0	.NET 6.0	818.017 ns	15.0026 ns	13.8289 ns	810.704 ns	98.23	3.40	0.0458	192 B	3.43
FastReflection	.NET 6.0	.NET 6.0	579.835 ns	5.1821 ns	4.3273 ns	579.669 ns	69.64	2.73	0.0267	112 B	2.00
Baseline	.NET 7.0	.NET 7.0	8.172 ns	0.1185 ns	0.1108 ns	8.132 ns	0.98	0.05	0.0134	56 B	1.00
Delegate	.NET 7.0	.NET 7.0	9.079 ns	0.2348 ns	0.2784 ns	9.093 ns	1.10	0.07	0.0134	56 B	1.00
DynamicMethod	.NET 7.0	.NET 7.0	10.947 ns	0.2749 ns	0.8533 ns	10.813 ns	1.30	0.16	0.0134	56 B	1.00
CompiledExpression	.NET 7.0	.NET 7.0	9.509 ns	0.2473 ns	0.3702 ns	9.425 ns	1.15	0.09	0.0134	56 B	1.00
FastCompiledExpression	.NET 7.0	.NET 7.0	9.851 ns	0.2470 ns	0.3123 ns	9.823 ns	1.20	0.09	0.0134	56 B	1.00
SlowReflection	.NET 7.0	.NET 7.0	488.517 ns	7.9631 ns	7.8208 ns	486.134 ns	58.87	3.05	0.0458	192 B	3.43
FastReflection	.NET 7.0	.NET 7.0	253.621 ns	4.9668 ns	4.6460 ns	254.297 ns	30.42	1.75	0.0267	112 B	2.00
Baseline	.NET Core 3.1	.NET Core 3.1	NA	NA	NA	NA	?	?	-	-	?
Delegate	.NET Core 3.1	.NET Core 3.1	NA	NA	NA	NA	?	?	-	-	?
DynamicMethod	.NET Core 3.1	.NET Core 3.1	NA	NA	NA	NA	?	?	-	-	?
CompiledExpression	.NET Core 3.1	.NET Core 3.1	NA	NA	NA	NA	?	?	-	-	?
FastCompiledExpression	.NET Core 3.1	.NET Core 3.1	NA	NA	NA	NA	?	?	-	-	?
SlowReflection	.NET Core 3.1	.NET Core 3.1	NA	NA	NA	NA	?	?	-	-	?
FastReflection	.NET Core 3.1	.NET Core 3.1	NA	NA	NA	NA	?	?	-	-	?
Baseline	.NET Framework 4.8	.NET Framework 4.8	8.376 ns	0.3826 ns	1.0879 ns	8.059 ns	1.00	0.00	0.0134	56 B	1.00
Delegate	.NET Framework 4.8	.NET Framework 4.8	8.436 ns	0.2049 ns	0.2736 ns	8.370 ns	1.02	0.08	0.0134	56 B	1.00
DynamicMethod	.NET Framework 4.8	.NET Framework 4.8	11.009 ns	1.1505 ns	3.3922 ns	8.773 ns	1.41	0.50	0.0134	56 B	1.00
CompiledExpression	.NET Framework 4.8	.NET Framework 4.8	45.918 ns	0.9788 ns	1.9772 ns	45.435 ns	5.46	0.55	0.0134	56 B	1.00
FastCompiledExpression	.NET Framework 4.8	.NET Framework 4.8	8.655 ns	0.2408 ns	0.2958 ns	8.638 ns	1.06	0.05	0.0134	56 B	1.00
SlowReflection	.NET Framework 4.8	.NET Framework 4.8	1,929.166 ns	151.8744 ns	447.8051 ns	1,654.722 ns	235.73	52.17	0.0916	385 B	6.88
FastReflection	.NET Framework 4.8	.NET Framework 4.8	1,117.981 ns	7.4267 ns	6.2016 ns	1,116.260 ns	134.28	5.22	0.0725	385 B	5.45

Тестування спроєктованої архітектури

- Перевірка результатів в нашій системі здійснюється через асинхронне виконання запитів, маппінг результатів на об'єкти, використання рефлексії для динамічного створення інстансів, юніт-тести для валідації функціональності та обробку винятків для забезпечення надійності системи. Ці методи допомагають переконатися, що система працює коректно і відповідає очікуванням.



Результати дослідження

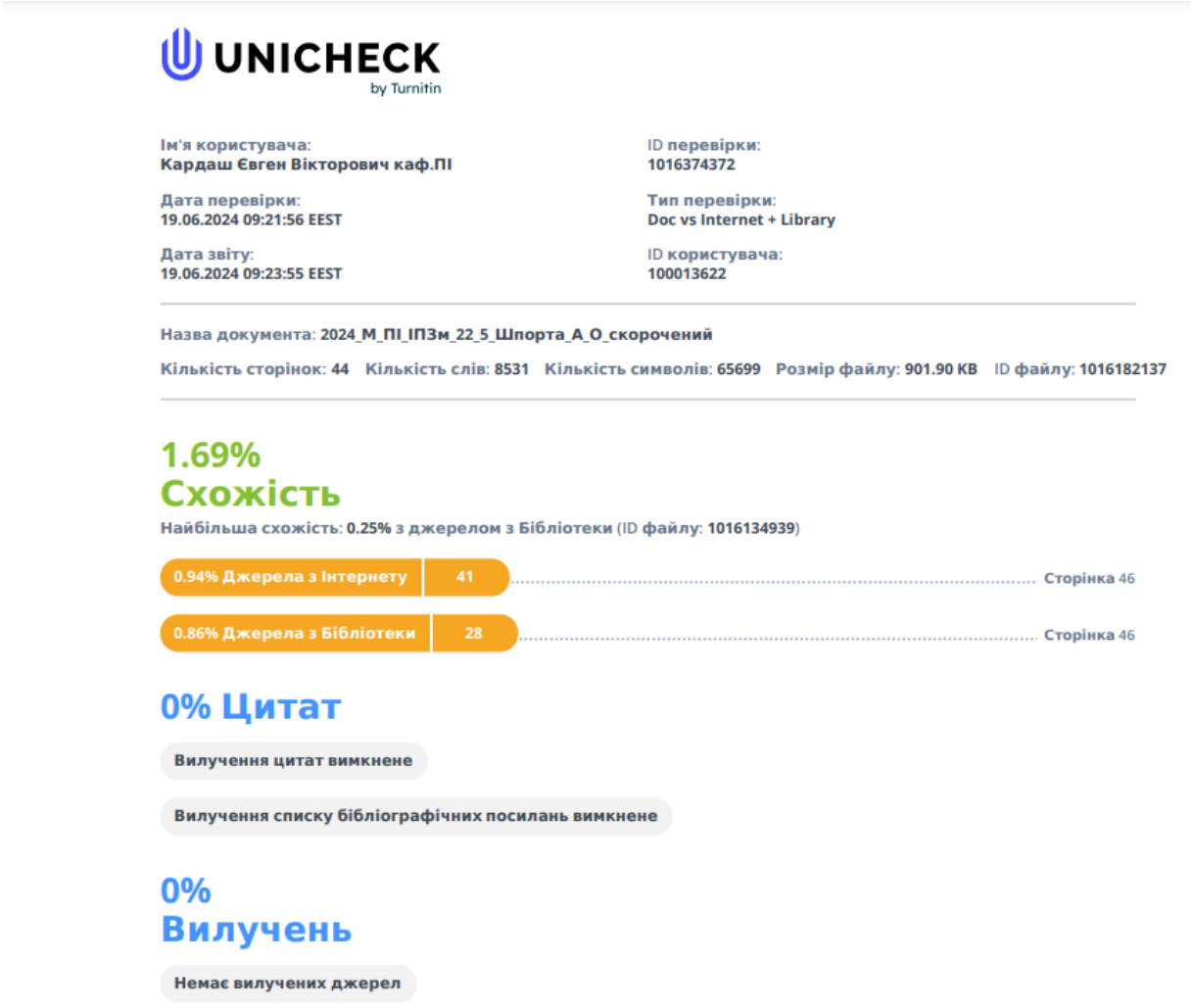
- Провівши дослідження у створеній ORM системі визначено, що навіть легковагова ORM-система, яка розроблена за допомогою запропонованої архітектури, може забезпечити більш ефективну та гнучку роботу з базами даних, використовуючи сучасні підходи, такі як дерева виразів, асинхронне програмування та динамічний маппінг.
- Подальші дослідження та вдосконалення можуть включати розширення функціональності системи, інтеграцію з міграційними інструментами та покращення механізмів оптимізації запитів

Висновки

- Проведено порівняльний аналіз архітектур, який охопив такі загальновідомі ORM як Hibernate, EF Core та PetaPoco.
- Проведено аналіз призначення кожного архітектурного елементу обраних ORM, створено стандартизовану архітектуру, яка відображає рекомендовану основу для створення компактних ORM.
- Були знайдені закономірності в архітектурах сучасних ORM, на основі яких сформовано стандартизовану архітектуру ORM. Вона була протестована за допомогою програмного застосунку.

ДОДАТОК В

Результат проходження на академічний плагіат



UNICHECK
by Turnitin

Ім'я користувача:
Кардаш Євген Вікторович каф.ПІ

ID перевірки:
1016374372

Дата перевірки:
19.06.2024 09:21:56 EEST

Тип перевірки:
Doc vs Internet + Library

Дата звіту:
19.06.2024 09:23:55 EEST

ID користувача:
100013622

Назва документа: 2024_М_ПІ_ІПЗм_22_5_Шпорта_А_О_скорочений

Кількість сторінок: 44 Кількість слів: 8531 Кількість символів: 65699 Розмір файлу: 901.90 KB ID файлу: 1016182137

1.69%
Схожість

Найбільша схожість: 0.25% з джерелом з Бібліотеки (ID файлу: 1016134939)

0.94% Джерела з Інтернету 41 Сторінка 46

0.86% Джерела з Бібліотеки 28 Сторінка 46

0% Цитат

Вилучення цитат вимкнене

Вилучення списку бібліографічних посилань вимкнене

0%
Вилучень

Немає вилучених джерел

ДОДАТОК Г

Апробація результатів роботи

УДК 004.415:004.2 DOI: https://doi.org/10.30837/IYF.IIS.2024.529

ДОСЛІДЖЕННЯ АРХІТЕКТУРНИХ МОДЕЛЕЙ ТА МЕТОДІВ ДЛЯ ПОБУДОВИ ORM ТА ЇХ КОМПАКТНИХ ВАРІАНТІВ НА ПЛАТФОРМІ .NET

Шпорта А. О.
Науковий керівник – к.т.н., доц. Мельнікова Р. В.
Харківський національний університет радіоелектроніки, каф. ПІ
М. Харків, Україна
e-mail: artem.shporta@nure.ua

This research work is aimed at the analysis of architectural models and methods of building compact Object-Relational Mapping (ORM) on the .NET platform. We examine existing approaches to creating compact ORMs, compare their architectural features, and identify key features for developing a common approach to the architecture of such libraries. The main goal of the work is to contribute to the development of more efficient and productive data access tools in the .NET environment.

Область нашої роботи охоплює дослідження та розвиток компактних Object-Relational Mapping (ORM) на платформі .NET. ORM – це методологія, яка дозволяє розробникам працювати з даними бази даних у вигляді об'єктів програмного коду, що спрощує взаємодію з базою даних та полегшує розробку програмного забезпечення.

Дана робота спрямована на аналіз існуючих підходів до побудови компактних ORM, порівняння їхніх архітектурних особливостей та розробку загальних підходів до архітектури таких бібліотек. Ми маємо на меті покращити ефективність та продуктивність розробки програмного забезпечення, що використовує бази даних у .NET-середовищі.

ORM (Object-Relational Mapping) – це технологія програмування, що використовується для зв'язку об'єктно-орієнтованих мов програмування з реляційними базами даних. ORM надає абстракцію над базою даних, що спрощує взаємодію з нею та полегшує розробку програмного забезпечення.

Основні принципи ORM включають відображення класів програмного коду на таблиці бази даних та відображення атрибутів класів на колонки таблиць[1].

ORM також надає можливість виконувати операції з базою даних (створення, читання, оновлення, видалення) з використанням об'єктів програмного коду, що забезпечує більш зрозумілу та підтримувану розробку.

Компактні ORM (Object-Relational Mapping) є важливим компонентом розробки програмного забезпечення, особливо в сучасному світі, де доступ до даних із баз даних є однією з найбільш поширених задач.

Подібні системи ORM створюють можливість розробникам працювати з даними у вигляді об'єктів програмного коду, а не SQL-запитів та рядків даних. Вони спрощують взаємодію з базами даних та дозволяють створювати більш читабельний та підтримуваний код.

Проте, на сьогоднішній день не існує конкретних стандартних інструкцій або загальної архітектурної моделі для розробки компактних ORM. Кожна бібліотека або фреймворк, які надають такий функціонал, може мати свою власну архітектурну концепцію та інтерфейс.

Ця ситуація створює необхідність розробити загальний підхід до архітектури компактних ORM. Дана необхідність зумовлена наступними факторами:

– відсутність загальних стандартів та рекомендацій з архітектури компактних ORM може призвести до великої різноманітності підходів, що ускладнює вибір правильної бібліотеки або фреймворку для проекту;

– розробка загальної архітектурної моделі може допомогти спростити розробку компактних ORM, забезпечуючи консистентність та стандартизацію;

– загальний підхід може полегшити підтримку і розширення існуючих компактних ORM. Розробники можуть бути більш обізнані з архітектурними концепціями та краще розуміти, як працює компонент;

– створення загального підходу до архітектури компактних ORM може сприяти розвитку спільноти та розширенню знань та ресурсів для цього напрямку розробки[2].

Отже, необхідність розробки загального підходу до архітектури компактних ORM визначається потребою в стандартизації, спрощенні розробки, підвищенні продуктивності та покращенні підтримки та розширення таких бібліотек.

Дана робота спрямована на виявлення закономірностей в архітектурах сучасних ORM, на основі закономірностей було сформовано тенденційні патерни для компактних варіантів.

Реалізація наукового дослідження складається з наступних етапів:

– аналіз предметної області;

– аналіз архітектурних особливостей сучасних ORM;

– знаходження закономірностей та логічних висновків що до ідеальної архітектури компактних ORM.

Виведену архітектуру буде повторно проаналізовано та на її основі створено невелику програмну реалізацію.

За результатами роботи було проаналізовано предметну область, та сферу застосування певних ORM, потім була поставлена задача та розроблено стратегію виявлення оптимальних архітектурних патернів для компактних ORM.

Для досягнення цієї мети було розглянуто та проаналізовано архітектури існуючих ORM фреймворків, які широко використовуються в

Java та .NET середовищах. Дослідження охопило як великі, так і компактні ORM системи, включаючи Entity Framework Core для .NET та PetaPoco – як приклад компактного ORM. Після поглибленого аналізу було розроблено приблизну архітектурну схему.

Візуальний SWOT-аналіз перетинів та закономірностей порівняння цих двох ORM зображено на рисунку 1.

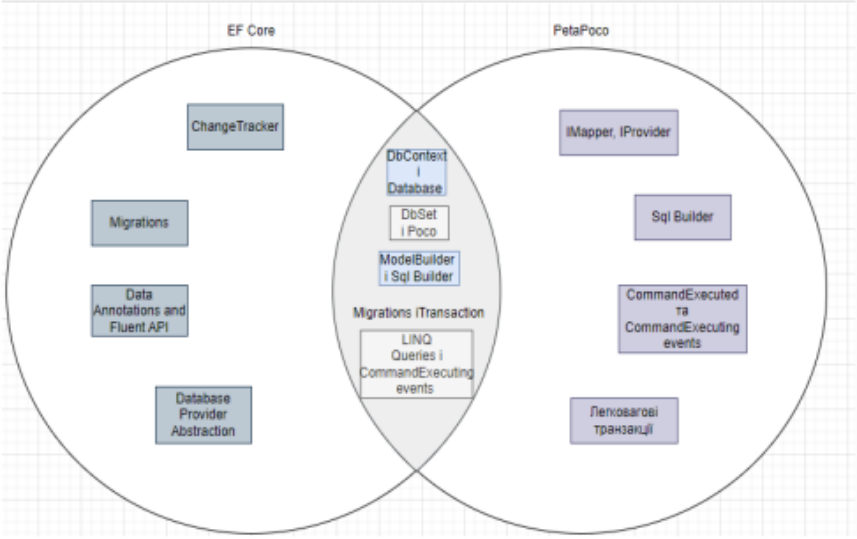


Рисунок 1 – Схема перетину ORM PetaPoco та EF Core

Отже, ці елементи і підходи вирішують завдання, які є центральними для компактних ORM, але можуть бути недостатньо комплексними або занадто простими для повнофункціональних ORM, які вимагають більш розширеного управління даними та відносинами.

Ці структурні патерни разом створюють функціональну основу для компактних ORM, вони роблять їх підходящим для проектів, де необхідний простий у реалізації та ефективний доступ до даних.

На основі аналізу було виявлено певні закономірності які характерні для систем не передбачаючи великих навантажень та довгого часу розробки, надалі ці напрацювання будуть доповнені новими елементами для розробки оптимального архітектурного патерну.

Список використаних джерел:

1. Michał Markiewicz. Repository architecture with OrmLite, 2017. : https://medium.com/@masztalski/repository-architecture-with-ormlite-fcf7e08ad23e (дата звернення: 16.05.2023).

2. Mazurova, O., Naboka, A., Shirokopetleva, M. (2021). Research of ACID transaction implementation methods for distributed databases using replication technology. Innovative technologies and scientific solutions for industries, (2 (16)), 19-31. DOI: 10.30837/ITSSI.2021.16.019.

ДОДАТОК Д

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-22-5 (група)

Шпорта Артем Олексійович

(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	

зауважень немає

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

19.06.2024

ДОДАТОК Е

Структура класів