ХЕРСОНСЬКИЙ НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ
(повне найменування вищого навчального закладу)
ФАКУЛЬТЕТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ ТА ДИЗАЙНУ
(повне найменування інституту, назва факультету (відділення))
КАФЕДРА ПРОГРАМНИХ ЗАСОБІВ І ТЕХНОЛОГІЙ
(повна назва кафедри (предметної, циклової комісії))

Пояснювальна записка

до кваліфікаційної роботи

бакалавра
(освітньо-кваліфікаційний рівень)

на тему: «Розробка компактного сервісу об'єктно-реляційного маппінгу для невеликих програмних проєктів»

Виконав: здобувач *4* курсу, групи *2ПРс*
напряму підготовки (спеціальності)
121 «Інженерія програмного забезпечення»
(шифр і назва напряму підготовки, спеціальності)
Долбня І.С.
(прізвище та ініціали)
Керівник Доровьска І.О.
(прізвище та ініціали)
Рецензент
(прізвище та ініціали)

Хмельницький – 2025

ХЕРСОНСЬКИЙ НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРЗВТЕТ

Інститут, факультет, відділення *Інформаційних технологій та дизайну*
Кафедра, циклова комісія *Програмних засобів і технологій*
Освітньо-кваліфікаційний рівень *бакалавр*
ОПП *Програмна інженерія*
(шифр і назва)
Спеціальність 121 – Інженерія програмного забезпечення
(шифр і назва)

ЗАТВЕРДЖУЮ
Завідувач кафедри,
програмних засобів і технологій
_____ к.т.н., доцент О.Е. Огнєва
«___» _____ 2025 року

З А В Д А Н Н Я

НА КВАЛІФІКАЦІЙНУ РОБОТУ ЗДОБУВАЧА

Долбня Ілля Сергійович
(прізвище, ім'я, по батькові)

1. Тема проекту (роботи) «Розробка компактного сервісу об'єктно-реляційного Маппінгу для невеликих програмних проєктів»

керівник проекту (роботи) к.т.н., доц. Доровська Ірина Олександрівна
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом вищого навчального закладу від «_20_» січня 2025 року №_99-с_

2. Строк подання студентом проекту(роботи) 17.06.2025

3. Вихідні дані до проекту (роботи)

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)
Аналітичний огляд предметної області, постановка задачі, проєктування програмного забезпечення, реалізація програмного забезпечення, тестування та верифікація, висновки

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень)
Комп'ютерна презентація

6. Консультанти розділів проекту (роботи)

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв

7. Дата видачі завдання _____

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів атестаційного проекту (роботи)	Строк виконання етапів проекту (роботи)	Примітка
1	Отримання завдання	09.02.2025	Виконано
2	Збір і аналіз літературних джерел	10.02.2025 – 25.02.2025	Виконано
3	Дослідження предметної області	28.02.2025 – 10.03.2025	Виконано
4	Побудова концептуальної моделі компактного ORM-сервісу	12.03.2025 – 22.03.2025	Виконано
5	Проєктування архітектури сервісу об'єктно-реляційного мапінгу	25.03.2025 – 30.03.2025	Виконано
6	Моделювання бази даних та визначення мапінг-стратегій	02.04.2025 – 10.04.2025	Виконано
7	Реалізація ключових елементів ORM-сервісу	12.04.2025 – 20.04.2025	Виконано
8	Написання коду програми	21.04.2025 – 20.05.2025	Виконано
9	Підготовка та оформлення розрахунково-пояснювальної записки	21.05.2025- 01.06.2025	Виконано
10	Проведення захисту кваліфікаційної роботи	18.06.2025	Виконано

Здобувач _____ І.С.Долбня
(підпис) (прізвище та ініціали)

Керівник проекту (роботи) _____ І.О.Доровьска
(підпис) (прізвище та ініціали)

РЕФЕРАТ

Кваліфікаційна робота бакалавра: 95 сторінок, 21 рисунка, 2 додатки, 43 джерел, 1 таблиця.

Мета роботи – розробити компактний ORM-сервіс для ефективної взаємодії програмного забезпечення з реляційними базами даних, інтегрований із NestJS та оптимізований для невеликих проєктів. Проєкт зосереджений на створенні інструменту, що забезпечує зручне управління схемами баз даних та автоматичне генерування SQL-запитів.

Об'єкт дослідження – системи Object-Relational Mapping (ORM) для взаємодії з реляційними базами даних.

Предмет дослідження – розробка ORM-сервісу на базі Node.js/TypeScript із підтримкою PostgreSQL, валідацією через JSON-схеми та автоматичним генеруванням міграцій.

Методи дослідження – аналіз існуючих ORM-фреймворків (TypeORM, Prisma, Sequelize), вивчення літератури, побудова UML-діаграм, розробка на TypeScript, тестування з PostgreSQL.

Наукова новизна: створено легковаговий ORM-сервіс із унікальним підходом до валідації JSON-схем через Zod та інтеграцією з NestJS, що підвищує гнучкість і швидкість розробки для невеликих проєктів.

Практичне значення – ORM-сервіс спрощує серверну розробку, зменшує кількість рутинного коду, мінімізує помилки й пришвидшує створення ПЗ. Може використовуватись у навчальних і комерційних проєктах малого та середнього масштабу.

Ключові слова: Object-Relational Mapping, ORM, SQL, реляційні бази даних, NestJS, PostgreSQL, JSON-схеми, міграції, TypeScript, валідація.

АНОТАЦІЯ

У роботі розглянуто процес розробки компактного сервісу об’єктно-реляційного мапінгу (ORM) для взаємодії з реляційними базами даних у невеликих програмних проєктах, з метою забезпечення зручного управління даними та автоматизації SQL-запитів. Загальний обсяг роботи складає 95 сторінки, 21 рисунків, 1 таблицю, 43 літературних джерел, 2 додатки.

Перший розділ «Дослідження та аналіз предметної області» присвячено вивченню об’єктно-реляційного мапінгу (ORM), аналізу його ролі у взаємодії з реляційними базами даних та проблематики створення легких ORM-рішень для невеликих проєктів. Проведено порівняльний аналіз популярних ORM-фреймворків (TypeORM, Prisma, Sequelize, Hibernate), визначено їхні переваги, недоліки та придатність для використання в умовах обмежених ресурсів.

Другий розділ «Проєктування об’єктно-реляційного мапінгу» охоплює розробку концептуальної архітектури ORM-сервісу для роботи з PostgreSQL. Сформовано вимоги до системи, зокрема підтримку автоматичної генерації SQL-запитів, CRUD-операцій, зв’язків між сутностями (1:1, 1:N, M:N), транзакцій та міграцій.

Третій розділ «Програмна реалізація» описує створення ORM-сервісу на базі TypeScript та Node.js. Реалізовано модульну архітектуру з менеджерами підключень, схем, генератором SQL-запитів, системою міграцій та CLI-інтерфейсом для спрощення управління базою даних.

Робота демонструє створення компактного, гнучкого та продуктивного ORM-сервісу, який спрощує взаємодію з PostgreSQL у невеликих програмних проєктах. Рішення відповідає сучасним стандартам бекенд-розробки, є придатним для використання в навчальних і комерційних цілях та може бути основою для подальшого розвитку складніших ORM-систем.

ABSTRACT

The work examines the process of developing a compact Object-Relational Mapping (ORM) service for interacting with relational databases in small software projects, aiming to ensure convenient data management and automation of SQL queries. The total scope of the work comprises 95 pages, 21 figures, 1 table, 43 literary sources, and 2 appendices.

The first section, "Research and Analysis of the Subject Area," is dedicated to studying Object-Relational Mapping (ORM), analyzing its role in interacting with relational databases, and addressing the challenges of creating lightweight ORM solutions for small projects. A comparative analysis of popular ORM frameworks (TypeORM, Prisma, Sequelize, Hibernate) was conducted, identifying their advantages, disadvantages, and suitability for use under limited resource conditions.

The second section, "Design of Object-Relational Mapping," covers the development of a conceptual architecture for an ORM service designed to work with PostgreSQL. Requirements for the system were formulated, including support for automatic SQL query generation, CRUD operations, entity relationships (1:1, 1:N, M:N), transactions, and migrations.

The third section, "Software Implementation," describes the creation of an ORM service based on TypeScript and Node.js. A modular architecture was implemented, including connection managers, schema managers, an SQL query generator, a migration system, and a CLI interface to simplify database management.

The work demonstrates the creation of a compact, flexible, and efficient ORM service that simplifies interaction with PostgreSQL in small software projects. The solution meets modern backend development standards, is suitable for educational and commercial purposes, and can serve as a foundation for further development of more complex ORM systems.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ СКОРОЧЕНЬ

ORM	Object-Relational Mapping (об'єктно-реляційне відображення)
SQL	Structured Query Language (мова структурованих запитів)
DBMS	Database Management System (система керування базами даних)
RDBMS	Relational Database Management System (реляційна система керування базами даних)
NoSQL	Not Only SQL (нереляційна база даних)
DDL	Data Definition Language (мова визначення даних у SQL)
DML	Data Manipulation Language (мова маніпуляції даними у SQL)
ACID	Atomicity, Consistency, Isolation, Durability (властивості транзакцій у базах даних)
CRUD	Create, Read, Update, Delete (базові операції з даними)
UoW	Unit of Work (патерн для роботи з множинними змінами у базі)
Repository	Шаблон проєктування для роботи з даними через ORM
Eager Loading	Завантаження пов’язаних об’єктів одразу при виконанні запиту
Lazy Loading	Завантаження пов’язаних об’єктів лише при необхідності
Migration	Механізм зміни структури бази даних
Query Builder	Інструмент для створення SQL-запитів у коді
Cache L1/L2	Кешування першого та другого рівня у ORM
Transaction	Група SQL-операцій, які виконуються як єдине ціле
Index	Механізм оптимізації пошуку у базі даних

Foreign Key	Зовнішній ключ, який забезпечує зв’язки між таблицями
Primary Key	Первинний ключ, що ідентифікує унікальний запис
Soft Delete	Видалення запису без фактичного видалення з бази
Connection Pool	Пул підключень до бази даних для зменшення накладних витрат
N+1 Problem	Проблема продуктивності при лінивому завантаженні пов’язаних об’єктів
Sharding	Розподіл бази даних на частини для масштабування
Replication	Копіювання бази даних на кілька серверів для резервування або масштабування
Multi-Tenancy	Архітектура, що дозволяє зберігати дані різних клієнтів у одній базі
ETL	Extract, Transform, Load (процеси обробки та перенесення даних)
Full-Text Search	Пошук по тексту у базі даних з урахуванням лексичного аналізу

ВСТУП

Актуальність теми. У сучасному світі інформаційних технологій ефективне управління базами даних є ключовим викликом бекенд-розробки, особливо в умовах зростання складності веб-застосунків, мікросервісних архітектур і хмарних технологій. Об’єктно-реляційний мапінг (ORM) відіграє важливу роль у спрощенні взаємодії між програмним кодом і реляційними базами даних, дозволяючи розробникам зосередитися на бізнес-логіці замість написання складних SQL-запитів. Сучасні ORM-інструменти, такі як TypeORM, Prisma, Sequelize та Hibernate, значно полегшують розробку, але часто мають обмеження, пов’язані з надмірною складністю конфігурації або накладними витратами на продуктивність. Це створює потребу в розробці легких, гнучких і продуктивних ORM-рішень, оптимізованих для невеликих проєктів, що особливо актуально для стартапів і компаній з обмеженими ресурсами, зокрема в українській IT-індустрії, яка активно розвивається в умовах глобальної цифрової трансформації.

Популярність хмарних сервісів, таких як AWS RDS, Google Cloud SQL чи Azure SQL, а також зростання обсягів даних у системах штучного інтелекту та аналітики підкреслюють необхідність ORM-систем, які забезпечують швидке розгортання, автоматизацію управління схемами баз даних і оптимізацію запитів. Крім того, потреба в технологічній незалежності та створенні власних рішень для українських проєктів робить розробку компактних ORM-систем важливим напрямком як для практичного застосування, так і для наукових досліджень у сфері управління даними.

Мета роботи. Метою роботи є створення базової об’єктно-реляційної моделі (ORM) для взаємодії з реляційними базами даних, що дозволяє автоматизувати основні операції збереження, вибірки, оновлення та видалення даних у базі без необхідності написання сирих SQL-запитів.

Завдання дослідження. Для досягнення поставленої мети необхідно виконати такі завдання:

1 Проаналізувати існуючі ORM-рішення (TypeORM, Prisma, Sequelize, Hibernate) та визначити їхні сильні та слабкі сторони.

2 Розробити концептуальну архітектуру ORM, що включає механізми роботи з моделями, генерації SQL-запитів, валідації даних та міграцій.

3 Реалізувати механізм парсингу моделей користувача, використовуючи Zod або аналогічні бібліотеки для типізації та валідації схем.

4 Створити механізм автоматичної генерації SQL-запитів на основі визначених моделей, включаючи підтримку CRUD-операцій.

5 Розробити систему роботи зі зв’язками між таблицями (One-to-One, One-to-Many, Many-to-Many) для забезпечення коректної взаємодії між сутностями в базі даних.

6 Запровадити механізм міграцій для динамічного оновлення структури бази даних на основі змін у моделях.

7 Оптимізувати виконання SQL-запитів шляхом впровадження механізмів кешування, індексації та аналізу продуктивності запитів.

8 Забезпечити роботу ORM у середовищі PostgreSQL, перевірити сумісність із базою даних та забезпечити коректну взаємодію через пул підключень.

9 Написати модульні та інтеграційні тести для перевірки коректності роботи ORM, використовуючи Jest або аналогічні тестові фреймворки.

10 Створити документацію для ORM, включаючи приклади використання, конфігурацію та пояснення внутрішніх механізмів роботи.

Виконання цих завдань дозволить реалізувати базову ORM-систему, яка може бути використана у розробці серверних застосунків, а також слугуватиме основою для подальшого розширення та оптимізації у більш складних проєктах.

Об’єкт дослідження. Об’єктом дослідження є процес взаємодії програмного забезпечення з реляційними базами даних через механізми

об’єктно-реляційного мапінгу (ORM). Це включає в себе загальні принципи побудови ORM-систем, їхню роль у сучасних інформаційних системах та способи реалізації ефективної роботи з базами даних.

Предмет дослідження. Предметом дослідження є методи та підходи до розробки базового ORM-рішення для роботи з PostgreSQL, зокрема механізми генерації SQL-запитів, управління транзакціями, реалізація CRUD-операцій, підтримка зв’язків між сутностями та оптимізація продуктивності взаємодії з базами даних.

Наукова новизна одержаних результатів. Наукова новизна дослідження полягає у наступному:

- Вперше здійснено комплексний аналіз ефективності існуючих ORM-рішень у контексті оптимізації взаємодії з реляційними базами даних та визначено їхні сильні та слабкі сторони.
- Розроблено новий підхід до автоматизованої генерації SQL-запитів на основі типізованих моделей, що дозволяє зменшити кількість помилок при роботі з базою даних.
- Запропоновано механізм інтеграції ORM із системами валідації схем (наприклад, Zod), що дозволяє підвищити рівень коректності даних ще на рівні моделювання структури БД.
- Визначено оптимальні стратегії управління зв’язками між сутностями в ORM-системі, які дозволяють мінімізувати проблему N+1 запитів та підвищити продуктивність взаємодії з даними.
- Розроблено та реалізовано базову ORM-систему з підтримкою автоматичних міграцій, що дозволяє розробникам динамічно змінювати структуру бази даних без ручного написання SQL-скриптів.
- Вперше формалізовано алгоритми роботи з кешуванням у рамках ORM, що дозволяє оптимізувати роботу з часто використовуваними запитами та зменшити навантаження на сервер бази даних.

- Створено концепцію гнучкого ORM-рішення, яке може бути використане у реальних комерційних проєктах із можливістю подальшого масштабування та інтеграції з іншими технологіями.

Результати цього дослідження сприяють підвищенню ефективності роботи з реляційними базами даних, що є важливим для оптимізації серверних застосунків, особливо в умовах високих навантажень та роботи з великими обсягами інформації.

Практичне значення одержаних результатів. Розроблена у межах дослідження базова ORM-система для роботи з PostgreSQL має практичне значення у сфері серверної розробки та взаємодії з реляційними базами даних. Основні аспекти практичного використання отриманих результатів:

- Автоматизація роботи з базою даних – ORM дозволяє взаємодіяти з реляційними сховищами без необхідності написання сирих SQL-запитів, що значно зменшує ризик помилок та підвищує швидкість розробки програмного забезпечення.
- Застосування у реальних проєктах – реалізоване ORM-рішення може бути використане у веб- та серверних застосунках, де необхідна структурована робота з даними, включаючи автоматизацію CRUD-операцій та підтримку складних запитів.
- Використання у навчальних цілях – ORM може бути використане як навчальний інструмент для студентів, що вивчають принципи роботи з базами даних та бекенд-технологіями, особливо у контексті розробки веб-сервісів та мікросервісної архітектури.
- Гнучкість у застосуванні – реалізоване рішення дозволяє легко адаптувати ORM до різних проєктів, зокрема у сферах e-commerce, фінансових технологій, управління контентом та автоматизації бізнес-процесів.

Використання розробленого ORM-рішення у реальних комерційних проєктах може суттєво спростити процес управління даними, підвищити швидкість розробки та мінімізувати помилки, пов’язані з некоректною взаємодією з базою даних.

Теоретичне значення одержаних результатів.

1 Розширення знань про об’єктно-реляційний мапінг – у дослідженні здійснено аналіз існуючих ORM-рішень та їхніх архітектурних підходів, що дозволяє систематизувати знання про ефективні стратегії роботи з реляційними базами даних.

2 Формалізація алгоритмів автоматизованої генерації SQL-запитів – розроблено модель створення SQL-запитів на основі об’єктних схем, що може бути використано у подальших дослідженнях, пов’язаних з оптимізацією взаємодії між кодом та базою даних.

3 Розробка підходів до оптимізації взаємодії між серверним кодом та базою даних – досліджено механізми Lazy Loading, Eager Loading, кешування та індексування, які дозволяють підвищити продуктивність ORM-систем.

4 Застосування моделей валідації схем даних – досліджено можливість інтеграції ORM з бібліотеками для валідації моделей (Zod, Joi, Yup), що підвищує безпеку роботи з базами даних та сприяє зниженню ризиків некоректного збереження даних.

Таким чином, дослідження робить внесок як у практичну, так і в науково-теоретичну сферу, сприяючи розвитку методів управління реляційними базами даних та їхньої інтеграції у сучасні веб- та серверні технології.

Наукове використання результатів дослідження та рекомендації щодо їх застосування. Результати цього дослідження можуть бути використані у наукових дослідженнях та розробках у сфері інформаційних технологій, а також у практичних застосуваннях, пов’язаних із взаємодією програмного забезпечення з реляційними базами даних.