МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ДЕРЖАВНИЙ УНІВЕРСИТЕТ «ЖИТОМИРСЬКА ПОЛІТЕХНІКА»
ФАКУЛЬТЕТ ІНФОРМАЦІЙНО-КОМП'ЮТЕРНИХ ТЕХНОЛОГІЙ
КАФЕДРА ІНЖЕНЕРІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

ПОЯСНЮВАЛЬНА ЗАПИСКА

до кваліфікаційної роботи освітнього ступеня «бакалавр»
за спеціальністю 121 «Інженерія програмного забезпечення»
(освітня програма «Веб технології»)
на тему:

«Бібліотека для взаємодії з реляційними базами даних»

Виконав студент групи ВТ-21-1
ШУМСЬКИЙ Олександр Вячеславович

Керівник роботи:
САВІЦЬКИЙ Роман Святославович

Рецензент:
ФАНТ Микола Вікторович

Житомир – 2025

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ДЕРЖАВНИЙ УНІВЕРСИТЕТ «ЖИТОМИРСЬКА ПОЛІТЕХНІКА»
ФАКУЛЬТЕТ ІНФОРМАЦІЙНО-КОМП'ЮТЕРНИХ ТЕХНОЛОГІЙ
КАФЕДРА ІНЖЕНЕРІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

«ЗАТВЕРДЖУЮ»
Зав. кафедри інженерії
програмного забезпечення
Тетяна Вакалюк
«14» лютого 2025 р.

ЗАВДАННЯ
на кваліфікаційну роботу

Здобувач вищої освіти: **ШУМСЬКИЙ Олександр Вячеславович**
Керівник роботи: **САВІЦЬКИЙ Роман Святославович**
Тема роботи: **«Платформа для управління IT-проєктами»**,
затверджена Наказом закладу вищої освіти від **«14» лютого 2025 р., №61/с**
Вихідні дані для роботи: **Об'єктом дослідження є використання об'єктно-реляційного підходу при роботі з базою даних. Предметом дослідження є використання технологій розробки для спрощення роботи з реляційними базами даних.**
Консультанти з бакалаврської кваліфікаційної роботи із зазначенням розділів, що їх стосуються:

Розділ	**Консультант**	Завдання видав	Завдання прийняв
1	Савіцький Р. С.	27.02.2024р.	27.02.2024р.
2	Савіцький Р. С.	27.02.2024р.	27.02.2024р.
3	Савіцький Р. С.	27.02.2024р.	27.02.2024р.

РЕФЕРАТ

Випускна кваліфікаційна робота бакалавра складається з бібліотеки для роботи з базами даних та пояснювальної записки. Пояснювальна записка до випускної роботи містить 61 сторінок, 53 ілюстрацій та 11 таблиць.

У роботі поставлені основні завдання на розробку ORM системи на Golang. Система включає модулі для операцій з базою даних, адміністративної панелі, логування та аутентифікації Також було проведено детальний порівняльний аналіз аналогічних систем. Було обрано підхід монорепозиторію з клієнт-серверною архітектурою. Наведені основні сценарії використання системи, логіка міжмодульної взаємодії. Клієнтська частина розроблена за допомогою бібліотеки HTMX а серверна частина за допомогою фреймворку Fiber. Для консольного клієнту використана бібліотека Cobra.

Ключові слова: ORM, БІБЛІОТЕКА, БАЗА ДАНИХ, GOLANG, SQL, СУТНІСТЬ, CRUD

ABSTRACT

The bachelor's final qualification work consists of an ORM Golang library and an explanatory note. The explanatory note to the final work contains 61 pages, 53 illustrations and 11 tables.

The work sets the main tasks for the development of an ORM system in Golang. The system includes modules for database operations, an administrative panel, logging and authentication. A detailed comparative analysis of similar systems was also conducted. A monorepository approach with a client-server architecture was chosen. The main scenarios for using the system and the logic of intermodule interaction are presented. The client part is developed using the HTMX library, and the server part is developed using the Fiber framework. The Cobra library is used for the console client.

Keywords: ORM, LIBRARY, DATABASE, GOLANG, SQL, ENTITY, CRUD

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

ПЗ – Програмне забезпечення

БД – База даних

ORM – Object-Relational Mapping

JWT – JSON Web Token

JS – JavaScript

TS – TypeScript

CRUD – Create Read Update Delete

ВСТУП

Актуальність теми пов’язана з активним розвитком програмування і потребою до гнучких інструментів для створення сучасних програмних продуктів. Одним із ключових напрямів є використання ORM (Object-Relational Mapping) систем, що дозволяють спрощено та структуровано взаємодіяти з базами даних. ORM дає змогу програмістам працювати з даними через об'єктно-орієнтований підхід, що полегшує розробку та підтримку програм.

Використання ORM системи дозволяє уникнути рутинної роботи з SQL-запитами, підвищити продуктивність та зменшити кількість помилок під час маніпуляції даними.

Метою дипломної роботи є дослідження використання ORM систем у Golang для спрощення роботи з базами даних та підвищення ефективності розробки.

Завдання дипломної роботи:

аналіз теоретичних засад ORM систем та їх можливостей;

дослідження особливостей ORM у Golang;

розробка моделі програмного продукту з використанням ORM;

створення прикладного додатку для демонстрації роботи ORM;

Предметом дослідження є особливості розробки ORM систем.

Об’єктом дослідження є технології об'єктно-реляційного відображення та методи їх впровадження у програмних продуктах.

За темою випускної кваліфікаційної роботи бакалавра було опубліковано тези:

Савіцький Р. С. ВІДМІННОСТІ РЕЛЯЦІЙНИХ БАЗ ДАНИХ. Тези доповідей X Міжнародна науково-технічна конференція Інформаційно-комп’ютерні технології: інновації, проблеми, рішення, 01-02 грудня 2022 року. Житомир: «Житомирська політехніка», 2025. С.164-165;

Савіцький Р. С. ACID-ВЛАСТИВОСТІ ТА РІВНІ ІЗОЛЯЦІЇ ТРАНЗАКЦІЙ. Тези доповідей X Міжнародна науково-технічна конференція Інформаційно-

комп’ютерні технології: інновації, проблеми, рішення, 01-02 грудня 2022 року. Житомир: «Житомирська політехніка», 2025. С.164-165;

ОСНОВНА ЧАСТИНА

1. Визначення та постановка задачі

Основна мета роботи є розробка ORM системи на Golang, задля підвищення ефективності роботи з базами даних розробниками програмного забезпечення, збільшення надійності написаного коду а також легкого адміністрування. ORM система має складатися з деяких ключових програмних модулів, а саме:

Модуль ядра має основний функціонал системи, який може використовуватись у всіх інших.

Модуль адаптерів має реалізації адаптерів для реляційних баз даних, готових для використання розробниками. Всі взаємодії з проектом відрізняються за рахунок адаптерів. Розробники не зобов'язані використовувати адаптери, адже вони мають можливість написати його самостійно якщо ні одне з готових рішень їм не підходить.

Модуль ORM служить проміжною ланкою між розробником та базою даних, має в собі функціонал абстракції над базою даних, дозволяє робити отримання та зміну даних залежно від потреб розробника, працює за допомогою вказаного адаптеру.

Модуль адміністративної панелі дозволяє проводити роботу з базою даних у веб-інтерфейсі, він служить для адміністраторів баз даних і дає змогу швидко маніпулювати даними.

Модуль консольного додатку дає змогу розробникам працювати з когенерацією а також міграціями через зручний консольний інтерфейс.

Основними етапами розробки платформи є:

- Написання перевикористовувальної логіки запитів до бази даних.
- Розробка готових імплементацій адаптерів для використання розробниками
- Розробка адміністративної панелі

- Розробка консольного інтерфейсу
- Тестування додатку

Результатом реалізації поставленого завдання є бібліотека ORM системи, яка включає в себе основну логіку роботи з базами даних, готові адаптери для популярних реляційних баз даних, адміністративну панель та консольний інтерфейс.

2. Аналіз аналогів програмного продукту

Основна мета ORM системи полягає в збільшенні продуктивності роботи розробників для виконання операцій з базою даних. Система підвищує надійність системи за допомогою чіткої схеми бази даних а також захисту від SQL-ін’єкцій. Також система має зручний інтуїтивний інтерфейс, який дасть змогу редагувати дані додатків нетехнічним людям. Відносно цих критерій розглянемо аналоги системи для дослідження ринку в порівняльній таблиці 1.1. Розглянемо аналоги детальніше:

Таблиця 1.1

Перелік аналогів системи

Назва	Підтримувальні БД	Особливості	Плюси	Мінуси
GORM	MySQL, PostgreSQL, SQLite, SQL Server, TiDB, Clickhouse	Найпопулярніша ORM для Golang, активна розробка, авто-міграції, хуки	Простий у використанні, велика спільнота, гнучкість	Повільний у порівнянні з "чистим" SQL, складний для кастомізації
Ent	MySQL, PostgreSQL, SQLite	Генерація схем, типобезпечність, граф-орієнтований API	Висока продуктивність, потужний генератор коду, хороший DX	Високий поріг входу, потребує попередньої генерації коду

Продовження таблиці 1.1

XORM	MySQL, PostgreSQL, CockroachDB, SQLite, MS SQL, Oracle, TiDB	Простий API, кешування, автоматичні міграції	Простий синтаксис, ефективне кешування, хороша підтримка	Мало різноманітних можливостей
SQLBoiler	MySQL, PostgreSQL, SQLite, MS SQL, CockroachDB	Генерація ORM-коду з SQL-схем, строгий типобезпечний API	Висока продуктивність, прозорість на основі чистого SQL	Не підтримує динамічні запити, складніший у налаштуванні
Beego ORM	MySQL, PostgreSQL, SQLite	Вбудований в Beego framework, підтримка зв'язків	Легко інтегрується з Beego, простота використання	Мала гнучкість, неактивний розвиток

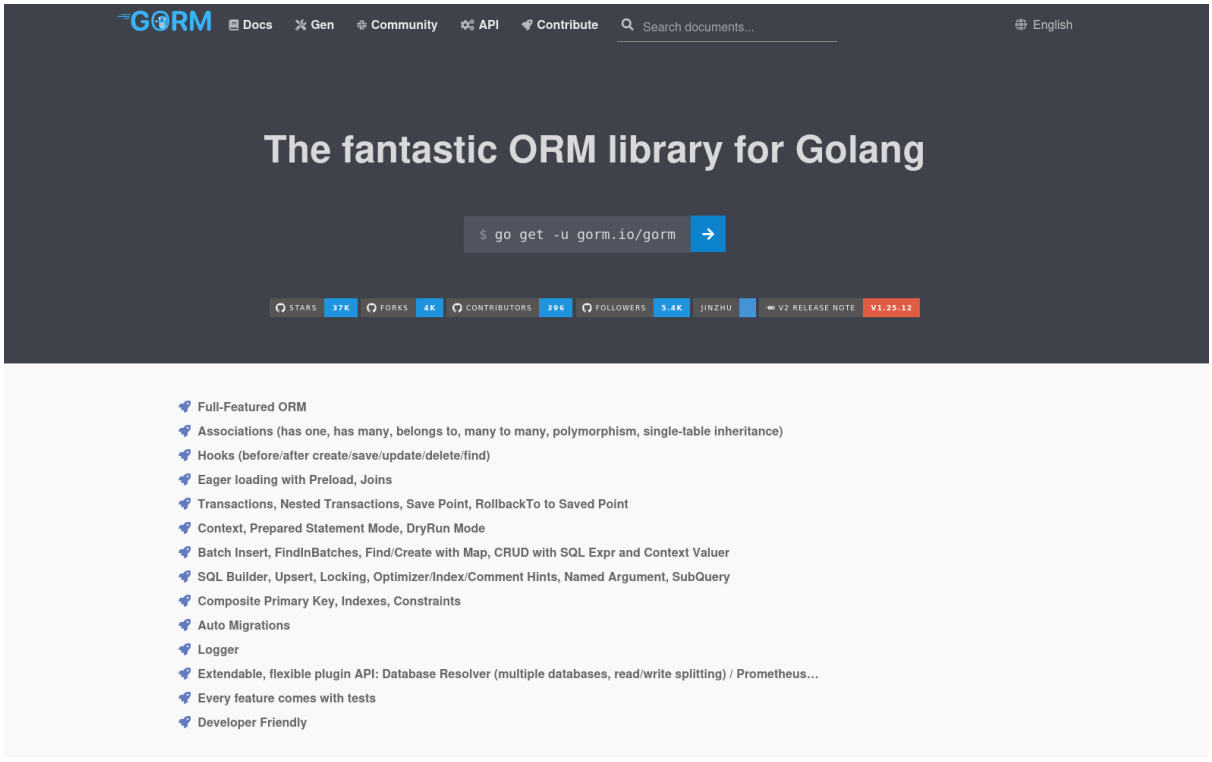

Рис. 1.1. Сайт бібліотеки «GORM»

GORM – це найпопулярніша ORM система [1]. Вона відома своєю зручністю та обширним функціоналом. Серед різноманітних баз даних вона підтримує MySQL, PostgreSQL, SQLite, SQL Server, TiDB та Clickhouse (див. рис.1.1).

Переваги:

- Інтуїтивна робота за допомогою будівельника запитів
- Автоматичні міграції, які дають змогу для синхронізації структури БД з моделями Go.
- Присутні розширені функції: хуки, "м'яке" видалення (soft delete), транзакції.
- Детальна документація та найбільша спільнота серед всіх ORM систем.

Недоліки:

- Продуктивність: Це може стати ключовим мінусом у виборі цієї ORM системи для високонавантажених додатків.
- Не повна підтримка складних запитів: Для деяких випадків доводиться вдаватися до використання чистого SQL.
- Проблеми з пошуком проблем: Помилки можуть бути недостатньо інформативними.

Дана ORM система є хорошим вибором для простих веб-додатків, де швидкість розробки є пріоритетом.

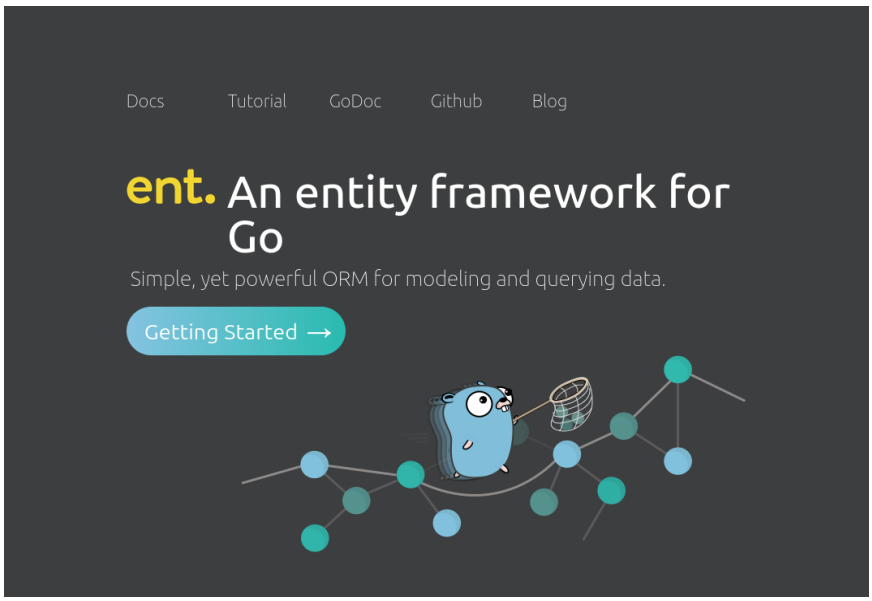

Рис. 1.2. Сайт бібліотеки «Ent»

Ent – це ORM від Facebook з кодогенерацією, орієнтована на типобезпеку та високу продуктивність (див. рис.1.2) [2].

Переваги:

- Генерує типобезпечний код на основі схем, що запобігає помилкам під час виконання.
- Висока продуктивність.
- Наявні можливості для створення складних запитів.
- Підтримка міграцій і плагінів для розширення функціоналу.

Недоліки:

- Складність налаштування, вимагає попередньої генерації коду.
- Мала популярність. Мало прикладів і менш активна спільнота порівняно з іншими ORM.

Дана ORM система є хорошим вибором для проектів з критичними вимогами до безпеки та продуктивності

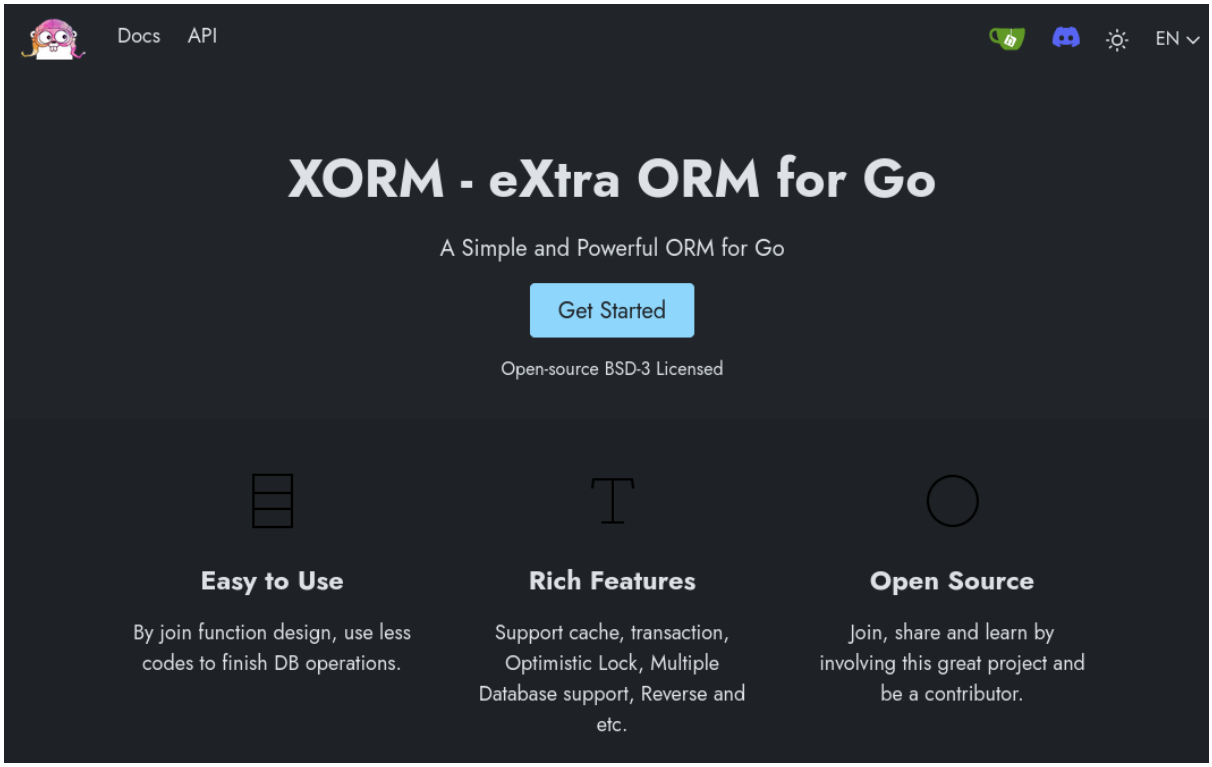

Рис. 1.3. Сайт бібліотеки «XORM»

XORM – це легка, але потужна ORM система з пріоритетом продуктивності (див. рис.1.3). Підтримує широкий спектр баз даних, включаючи Oracle та TiDB.

Переваги:

- Правильно згенеровані запити в поєднанні з вбудованим кешуванням допомагають у розробці високонавантажених систем.

- Добре генерує SQL з урахуванням специфік СУБД.
- Є вбудована можливість використовувати автоматичні міграції БД.
- Розширена підтримка транзакцій і розподілених систем.

Недоліки:

- Потрібно багато різних кроків для комплексних запитів.
- Обмежена документація. Часто можуть виникнути різні проблеми, які вимагають .

Дана ORM система є чудовим інструментом для додатків з великими обсягами даних або високим навантаженням.

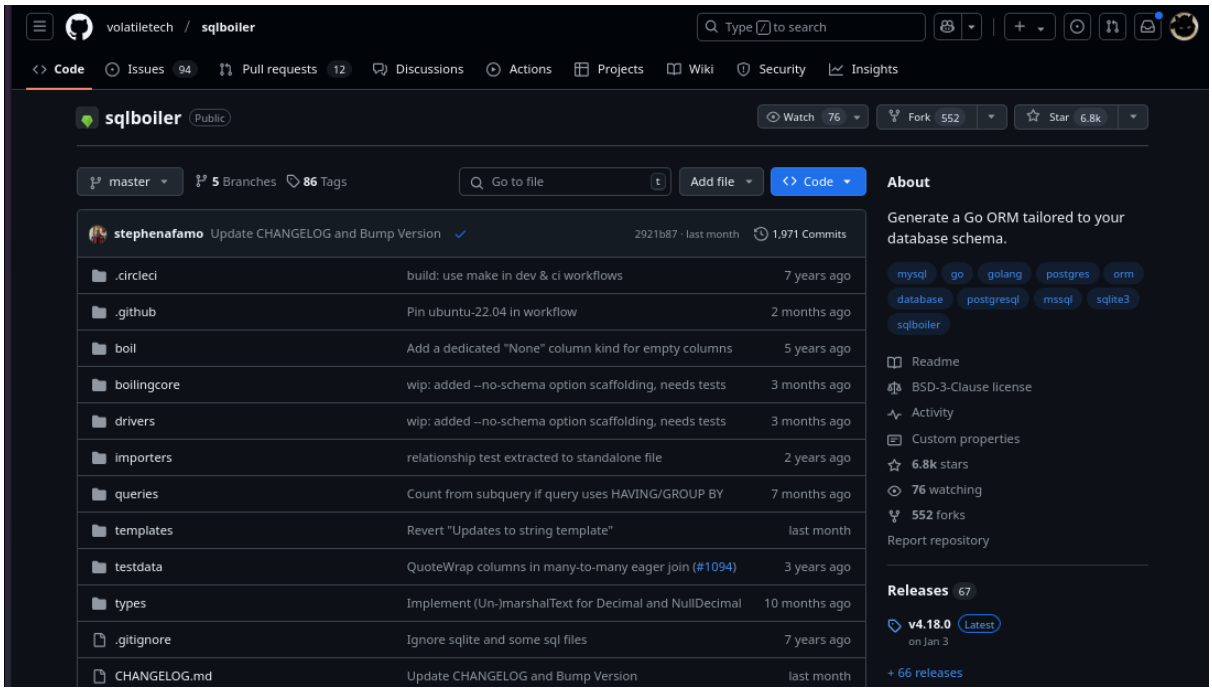

Рис. 1.4. Сторінка бібліотеки «sqlboiler» на сайті «github.com»

SQLBoiler – це ORM система, яка відрізняється від інших підходом «code-first»: вона генерує моделі з існуючої схеми бази даних, створюючи повністю типізований API [3]. Завдяки цьому підходу, SQLBoiler забезпечує високу швидкодію(див. рис.1.4).

Переваги:

- Більшість помилок виявляється ще на етапі компіляції.
- Висока продуктивність.

- Генерація моделей безпосередньо з існуючої схеми мінімізує розбіжності між кодом і структурою бази даних та підвищує надійність системи.

Недоліки:

- Необхідність повторного запуску коду при кожній зміні схеми бази даних. Це може спричиняти незручності при розробці додатків.
- Мала гнучкість. Підхід з статичний кодом, може набагато менш адаптивним для змін схеми бази даних.

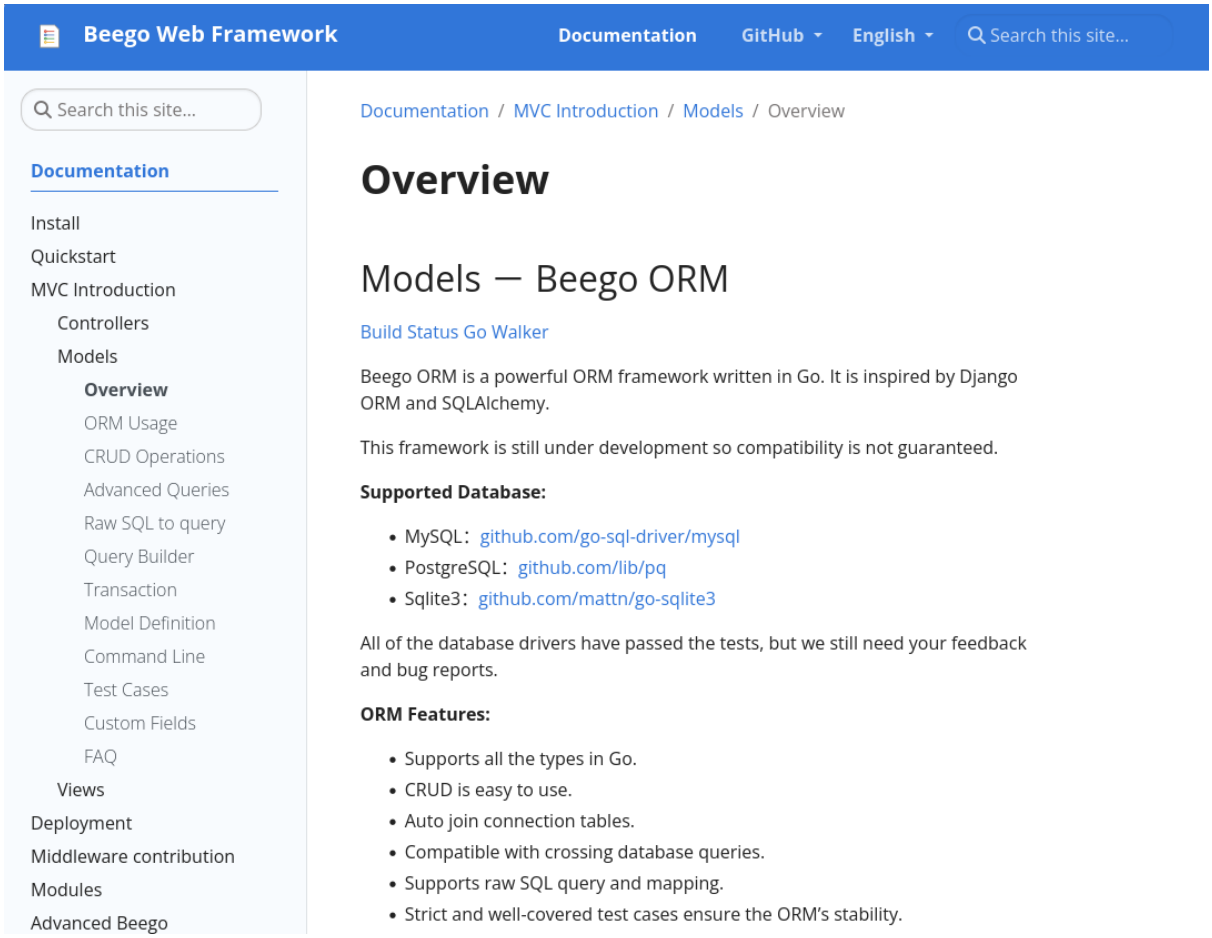

Рис. 1.5. Сайт бібліотеки «Beego»

Beego ORM це частина фреймворку Beego [4], орієнтована на простоту та інтеграцію з іншими модулями Beego (див. рис.1.5). Також вона є агностиком і може бути застосована в будь якому оточенні незалежно від фреймворку.

Переваги:

- Потрібно мінімальне налаштування для базових операцій.
- Інтеграція з Beego.

- Підтримка транзакцій і базових міграцій.

Недоліки:

- Обмежений функціонал. Не підходить для складних запитів і нестандартних СУБД.
- Мала спільнота.

Beego ORM це хороший вибір для проектів на фреймворці Beego або дуже малих проектів

Аналіз аналогічних платформ допоміг визначити сильні та слабкі сторони існуючих рішень. Як можна побачити з аналогів, всі наведені ORM системи мають підтримку MySQL, PostgreSQL та SQLite але на жаль вони майже не мають в собі вбудованих інструментів, які могли б спростити процес розробки. Таким чином, аналіз аналогів допоміг остаточно визначити необхідний набір функціоналу та розставити пріоритетність функціоналу у розробці системи.

3. Визначення архітектури ПЗ

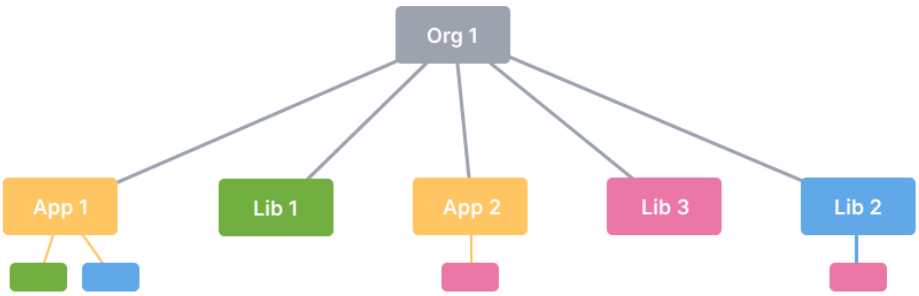

Рис. 1.6 Схема роботи монорепозиторія

Для розробки ORM системи було обрано підхід модульного монорепозиторія (див. рис.1.6) де різні частини можуть спілкуватись одна з одною за допомогою різних протоколів зв'язку. Такий підхід дає змогу користувачу дуже гнучко використовувати систему і підключати тільки ті частини які йому дійсно потрібні. Прикладом цього є різні адаптери баз даних, користувач може обрати один або декілька адаптерів і на основі цього використовувати систему. Також це дає змогу заміняти готові частини системи на свої власні імплементуючи взаємодію з інтерфейсами інших частин системи. Також цей підхід зводить дублювання коду

до мінімуму, так як спільний код можна переносити в окремі бібліотеки і використовувати в різних місцях.

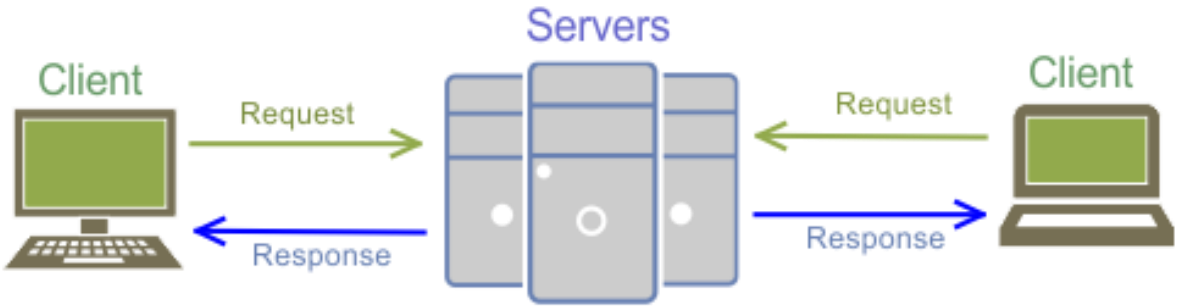

Рис. 1.7 Клієнт-серверна архітектура

В деяких випадках використовується клієнт-серверна архітектура (див. рис.1.7), де одні модулі є клієнтами а інші серверами.

Серверне представлення слугує для реалізації бізнес логіки системи. Всі важливі дані зберігаються тільки на сервері. В системі передбачені різні серверні частини для подальшої взаємодії з клієнтськими.

Клієнтське представлення слугує для відображення даних в зручному для користувача форматі. В системи передбачена варіативність різних клієнтів. Веб-клієнт повинен бути сумісним з різними веб-браузерами, таких як Mozilla FireFox та Google Chrome, і працювати на різних платформах, забезпечуючи зручний та надійний доступ до основних функцій. Інтерфейс користувача має бути інтуїтивним та зручним, з використанням сучасного дизайну. При розробці важливо забезпечити адаптивність для зручного використання на пристроях з різними розмірами екрану. Також важлива можливість використання сенсорних пристроїв для зручного використання. Консольний клієнт повинен мати вбудовані підказки для простого користування.

4. Обґрунтування вибору інструментальних засобів

При розробці було використано декілька інструментів для ефективної та зручної роботи. В якості основних інтегрованих середовищ розробки використовувалися Webstorm та Goland разом з необхідними плагінами для використаних бібліотек. Ці середовища розробки дуже добре себе показують в складних проектах завдяки індексації всього проекту і можуть виявити критичні помилки ще до запуску коду.

Для тестування REST API було обрано Postman. Він має в собі весь необхідний функціонал для тестування серверної частини а також має купу можливостей для автоматизації команд, які необхідні для коректної роботи системи. Він має зручний графічний інтерфейс який прискорює тестування системи і робить пошук проблем більш зручним.

Таблиця 1.2

Порівняння бібліотек для створення веб-інтерфейсів

	HTMX	React	Alpine.js
Складність	Низька	Висока	Низька
Підхід до рендерингу	Серверний (HTML over the wire)	Клієнтський (Virtual DOM)	Клієнтський (нативний DOM)
Інтерактивність	через AJAX/HTML	через компоненти	через директиви
Зв'язок із сервером	Вбудований (через атрибути)	Потрібен додатковий код (fetch)	Потрібен додатковий код (fetch)
Використання з Golang	Ідеально для серверного рендерингу	Потрібен окремий API (REST/GraphQL)	Потрібен окремий API (REST/GraphQL)
Розмір бібліотеки	~10 KB	~40 KB (React) + ~5 KB (ReactDOM)	~10 KB
Крива навчання	Дуже низька	Середня	Низька

Проаналізувавши бібліотеки для простих інтерактивних веб-інтерфейсів для клієнтської частини була обрана бібліотека HTMX [5] завдяки своїй простоті і підтримці серверного рендеру, яка не буде вимагати ще одного веб-сервера. Функціоналу HTMX цілком вистачить для відмальовування інтерактивного інтерфейсу адміністративної панелі.

Для серверної частини були проаналізовані найпопулярніші фреймворки на Golang. Для потреб проекту було обрано Fiber. Він має чудову продуктивність та вбудовану підтримку різноманітних технологій. Для консольної частини обрана бібліотека Cobra [6]

Таблиця 1.3

Порівняння фреймворків для Golang

	Fiber	Chi	Echo
Продуктивність	Дуже висока (побудований на fasthttp)	Висока (базується на net/http)	Висока (базується на net/http)
Підтримка WebSockets	Так	Ні	Так
Підтримка шаблонів	Так	Ні	Так
Готові рішення	Багато вбудованих функцій	Мінімалістичний, потребує додаткових бібліотек	Багато вбудованих функцій
Вбудована Підтримка JWT	Так	Ні	Так
Документація	Детальна та зрозуміла	Хороша, але менш структурована	Хороша, але менш детальна
Підтримка контексту	Власна реалізація	Через стандартний context.Context	Через стандартний context.Context

Вибір інструментів був обумовлений їх відповідністю поставленим вимогам. Використання сучасних технологій дозволяє покращити ефективність розробки системи.

Висновки до основної частини

У першому розділі було проведено детальний аналіз існуючих аналогів, обґрунтовано вибір інструментальних засобів, сформульовано основні завдання та визначено архітектуру програмного забезпечення. Детальний аналіз аналогів дозволив визначити необхідні функціональні вимоги до системи. Для реалізації проекту було обрано підхід модульного монорепозиторія з клієнт-серверною архітектурою, яка забезпечує масштабованість і гнучкість. Клієнтська частина створена на основі HTMX, що дозволяє зробити зручний користувацький веб-інтерфейс. Серверна частина створена на фреймворці Fiber, що забезпечує ефективну обробку запитів. При виборі технологічного стеку враховувалися потреби системи.

Загалом, описана архітектура та обрані технології забезпечують високу продуктивність, надійність, пришвидшення швидкості розробки додатків і можливість легкого розширення функціоналу системи.

ВИСНОВКИ

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Офіційна документація React [Електронний ресурс] – Режим доступу до ресурсу: https://react.dev/reference/react
2. Офіційна документація JavaScript [Електронний ресурс] – Режим доступу до ресурсу: https://developer.mozilla.org/en-US/docs/Web/JavaScript
3. Офіційна документація Node.js [Електронний ресурс] – Режим доступу до ресурсу: https://nodejs.org/docs/latest/api/
4. Офіційна документація Next.js [Електронний ресурс] – Режим доступу до ресурсу: https://nextjs.org/docs
5. Офіційна документація PostgreSQL [Електронний ресурс] – Режим доступу до ресурсу: https://www.postgresql.org/docs/
6. Linear [Електронний ресурс] – Режим доступу до ресурсу: https://linear.app/
7. GitHub Projects [Електронний ресурс] – Режим доступу до ресурсу: https://docs.github.com/en/issues/planning-and-tracking-with-projects/learning-about-projects/about-projects
8. Jira [Електронний ресурс] – Режим доступу до ресурсу: https://www.atlassian.com/software/jira