НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ ІМЕНІ ІГОРЯ СІКОРСЬКОГО»

Навчально-науковий інститут атомної та теплової енергетики

Кафедра цифрових технологій в енергетиці

"На правах рукопису"
УДК 341.951

«До захисту допущено»
Завідувач кафедри
_____ Наталія АУШЕВА
“___”_____2022р.

Магістерська дисертація

За освітньою програмою "Комп’ютерний моніторинг та геометричне моделювання процесів та систем"
Спеціальності 122 Комп’ютерні науки
на тему Компоненти візуальних середовищ для баз даних через дані ODBC

Виконав студент 2 курсу, групи ТР-12мп
Архипенко Дмитро Павлович _____
(прізвище, ім’я, по батькові) (підпис)

Науковий керівник к.е.н., доцент Ірина СЕГЕДА _____
(посада, вчене звання, науковий ступінь, прізвище та ініціали) (підпис)

Рецензент д.т.н., професор
Директор НТЦ «ЕКОТЕЗ» Генадій ВАРЛАМОВ _____
(посада, вчене звання, науковий ступінь, прізвище та ініціали) (підпис)

Засвідчую, що у цій магістерській дисертації немає запозичень з праць інших авторів без відповідних посилань.

Студент _____
(підпис)

Київ – 2022

Національний технічний університет України

«Київський політехнічний інститут ім. Ігоря Сікорського»

Факультет Навчально-науковий інститут атомної та теплової енергетики
Кафедра цифрових технологій в енергетиці
Рівень вищої освіти другий, магістерський
За освітньою програмою "Комп'ютерний моніторинг та геометричне моделювання процесів та систем"

Спеціальності 122 Комп'ютерні науки

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ Наталія АУШЕВА
(підпис)

«____» _____ 2022р.

З А В Д А Н Н Я

НА МАГІСТЕРСЬКУ ДИСЕРТАЦІЮ СТУДЕНТУ

Архипенку Дмитру Павловичу
(прізвище, ім'я, по батькові)

1. Тема дисертації Компоненти візуальних середовищ для БД через дані ODBC

Науковий керівник к.е.н., доцент Ірина СЕГЕДА
(науковий ступінь, вчене звання, ім'я, прізвище)

затверджена наказом вищого навчального закладу від "__" _____ 2022р. №3629-С
2. Строк подання студентом роботи "_" _____ 2022р.
3. Вихідні дані до роботи: мова програмування – JavaScript, середовище розробки –Webstorm, середовище побудови діаграм — Microsoft Visio
4. Перелік питань, які потрібно розробити: провести аналіз ринку та актуальності вибору відповідних технологій, спроектувати додаток, базу даних та бізнес-логіку, зробити висновки по виконаній роботі
5. Орієнтований перелік ілюстративного матеріалу: опис роботи, короткі відомості про СУБД, аналіз існуючих систем, обрані інструменти розробки, ролі в системі та прецеденти, життєвий цикл замовлення, база даних, структура проекту, приклад автоматично згенерованої документації, результати роботи програмного інтерфейсу взаємодії (каталог), результати роботи програмного інтерфейсу взаємодії (завершені пристрої), висновки
6. Орієнтований перелік публікацій:
7. Дата видачі завдання «__» _____ 2022 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів виконання магістерської дисертації	Строки виконання етапів магістерської дисертації	Примітки
1.	Вивчення та аналіз задачі	15.09.2021-01.10.2021	
2	Розробка архітектури та загальної структури системи	02.10.2022-23.10.2022	
3.	Розробка структур окремих підсистем	23.10.2022-24.11.22	
4.	Програмна реалізація системи	24.11.2022-30.12.2022	
5.	Оформлення пояснювальної записки	30.12.2022-08.12.2022	
6.	Передзахист	24.11.2022	
7.	Захист	19.12.2022	

Студент _____ Дмитро АРХИПЕНКО
(підпис) (прізвище та ініціали,)

Керівник роботи _____ Ірина СЕГЕДА
(підпис) (прізвище та ініціали,)

РЕФЕРАТ

Структура і обсяг. Дисертація містить вступ, 5 розділів, висновки. Надано 22 джерела у вигляді переліку використаних джерел, 51 ілюстрацій, 18 таблиць. Основна частина роботи викладена на 63 сторінках.

Актуальність. У дисертації описано комп'ютерну систему управління та оновлення даних, що знаходяться на віддалених SQL серверах. Результат роботи надає можливість швидко та інтерактивно переглядати і редагувати дані, що знаходяться на SQL сервері в базі певних даних.

Мета роботи. Проведення аналізу існуючих систем на ринку, моделювання та проектування процесів в системі управління базами даних та подальша імплементація системи.

Задачі:

- виконати аналіз відомих систем;
- змоделювати прецеденти;
- створити архітектуру;
- запропонувати методи для керування процесами;
- спроектувати базу даних, імплементувати бізнес-логіку, та інтерфейс взаємодії;
- виконати тестування системи.

Практичне значення одержаних результатів. Спроектовано та імплементовано комп'ютерний додаток для перегляду та редагування даних з бази, який здатний працювати з великою кількістю даних а також є кросс-платформенним. Описано діаграми та схеми роботи додатку, описано розроблені процеси, а також обґрунтований вибір стеку технологій для реалізації. В записці також продемонстровано роботу додатку.

Ключові слова: SQL, база даних, Electron, React, система керування базами даних, REST, UI.

ABSTRACT

The structure and volume of dissertation. The dissertation consists of introduction, 5 chapters, conclusions. 22 links, 51 figures, 18 tables are present. Main information is shown on the 63 pages.

The relevance. The essay contains research of automated system for observing and manging data stored in SQL DB. The practical usage of this system is shown by example of created cross platform application and further process modeling.

The goal. To execute research of existing and similar systems on the market, design processes in the system, database with further automated system creation.

Tasks. To achieve the goal the list of tasks is below:

- execute and analysis of known systems;
- create use interaction models (use cases);
- create an architecture;
- offer management of processes;
- design database, code business logic, design and code interaction interfaces;
- perform the testing of the system.

The practical significance of the obtained results. The automated system for observing and editing data in DBs has been designed and developed. The system is able to work with a big amount of data on different platforms and has an opportunity for further scalling. Use cases diagrams, created processes and relevance of used technologies are described. The system's work flow is demonstrated as well.

Key words: SQL, DB, Electron, React, DBMS, REST, UI.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

SQL – це мова програмування для роботи з наборами фактів і зв'язками між ними. У програмах керування реляційними базами даних, як-от Microsoft Office Access, мова SQL використовується для роботи з даними

База даних – це масив спеціальним чином організованої інформації, поданої у вигляді впорядкованого набору елементів однакової структури.

СУБД – це спеціальний пакет програм, що забезпечує створення, супроводження і використання баз даних багатьма користувачами. [14]

Сервер — програма яка працює в режимі реального часу і здатна взаємодіяти з користувачем, базою даних та реалізовувати логіку.

Реляційна база даних — база даних, заснована на реляційній моделі даних. Слово «реляційний» походить від англ. *relation* (відношення). Для роботи з реляційними БД застосовують реляційні СКБД. Інакше кажучи, реляційна база даних — це база даних, яка сприймається користувачем як набір нормалізованих відношень різного ступеня.

ПЗ – програмне забезпечення.

Десктоп – комп'ютер.

ODBC (англ. *Open DataBase Connectivity*) — це відкритий прикладний програмний інтерфейс доступу до баз даних, розроблений консорціумом X/Open.

ВСТУП

На сьогоднішній день для багатьох компаній є необхідними численні системи, які використовуються для автоматизації роботи в бізнесі, підтримки клієнтів, вивчення ринку, реклами тощо. Як правило, ці системи призначені для обробки корпоративної інформації, масштаби якої, як правило, досить великі. Відповідно, необхідно докладати багато зусиль для вирішення питань зберігання та маніпулювання даними. З цієї причини існує широкий вибір систем управління базами даних, однією з яких є реляційна система управління базами даних (СУБД).

Системи управління базами даних мають складну конструкцію. Їм необхідно дотримуватися реляційної моделі, яка вимагає багато можливостей, в тому числі впорядковувати всі дані в таблицях, зберігати вже встановлені параметри обмеження в стовпцях, первинних і зовнішніх ключах, здійснювати транзакції і зберігати властивості ACID. Оскільки дані зберігаються на фізичному носії, який має обмеження по швидкості та ємності, необхідно оптимальним чином організувати дані, а також реалізувати підходи до них. Найбільш звичними структурами є купи та індекси. Купа - це невпорядкований набір даних, тоді як індекс - це структуроване дерево, яке було збалансовано. Використання індексу може зменшити кількість операцій читання, необхідних для пошуку даних. Однак є й недоліки, такі як збільшення розміру бази даних та додаткові зусилля, необхідні для розробки та підтримки індексу. Крім того, база даних може містити додаткові функції, такі як статистичні дані, які необхідно підтримувати в актуальному стані.

Як правило, сервер здатний самостійно піклуватися про продуктивність своїх компонентів, таких як оновлення різноманітної статистики, підтримання логічної структури індексу та максимальне використання ресурсів. Однак для того, щоб сервер працював належним чином і підтримував свою продуктивність, необхідне його технічне обслуговування. Це досить складне та трудомістке завдання, яке потребує відповідних знань та інструментів, тому навіть фахівцям знадобиться певна апаратура.

Сьогодні для проектів будь-якого масштабу характерно використання декількох баз даних для зберігання специфічних даних. Саме тому розробка ефективного та дієвого додатку для управління цими даними та доступу до них є як ніколи актуальною; при використанні декількох баз даних нам необхідно встановлювати окремий програмний продукт для кожного типу SQL Server.

Для досягнення цієї мети необхідно здійснити такі основні кроки: аналіз існуючих систем; побудова архітектури; проектування бази даних; розробка інтерфейсу для полегшення взаємодії (API); створення десктопної програми, яка надає доступ до декількох серверів за допомогою параметрів ODBC.

У першому розділі описано схему роботи базового SQL сервера, наведено порівняльну характеристику найбільш поширених серверів ринку та описано засоби реалізації додатку.

У другому розділі описано вибрані технології що будуть використані для реалізації настільного додатку.

У третьому розділі описано архітектуру, користувацький інтерфейс розробленої системи.

У четвертому розділі наведено приклади роботи API у зв’язку з користувацьким інтерфейсом.

У п’ятому розділі розглянуто додаток, як потенційний стартап та описана його можливість виведення на ринок інформаційних технологій.

1 ЗАДАЧА РОЗРОБКИ СИСТЕМИ ВІЗУАЛІЗАЦІЇ КОМПОНЕНТІВ БАЗ ДАНИХ ЧЕРЕЗ ДАНІ ODBC

1.1 Актуальність роботи та постановка задачі розробки настільного додатку для візуалізації компонентів бази даних

Сьогодні багато підприємств покладаються на складні системи для автоматизації таких завдань, як обслуговування клієнтів, аналіз ринку та маркетинг. Ці системи призначені для обробки великих обсягів бізнес-даних, тому важливо мати хорошу систему управління базами даних для зберігання та обробки всіх цих даних. Реляційні системи управління базами даних (СКБД) є одними з найпоширеніших систем, що використовуються для цієї мети. Коли ми говоримо про вибір БД перш за все ми маємо розуміти, що між ними є велика різниця і кожна з них має свої переваги та недоліки, тому часто для розбиття продукту на мікросервіси розробники приходять до того, що їм необхідно використовувати різні бази для різних задач, наприклад, для роботи з надвеликими об’ємами даних краще використовувати Postgres за рахунок роботи індексів під капотом, з іншого боку, коли структура проекту не передбачає роботи з Big Data то достатньо буде MS SQL за рахунок швидкості обробки простих запитів до БД.

Саме тому розробка додатку, який міг би підключатись до різних серверів з БД міг би позбавити контент менеджера або самого розробника від необхідності встановлення індивідуального програмного забезпечення під кожен тип реляційної бази, адже підключення можна здійснювати по ODBC параметрах в рамках одного додатку до різних серверів

У мережі Інтернет ми можемо знайти різних дистриб’юторів БД, таких як: Postgres, MySQL та інші, підключення до серверів, яких здійснюється по параметрам ODBC. Для подальшого розуміння процесів закладених в основу візуалізації компонентів бази даних розглянемо схему того як працює SQL сервер [13].

1.2 Схема роботи SQL сервера на прикладі MS SQL Server

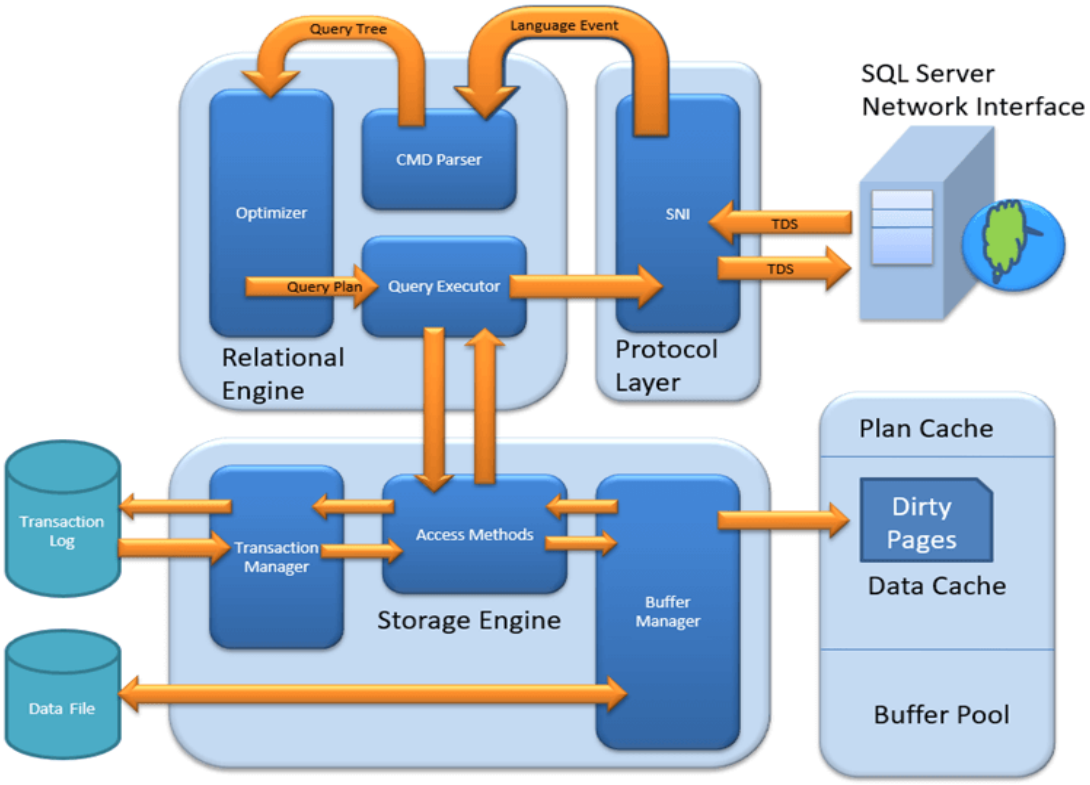


Рисунок 1.1 - Структура MS SQL Server

Структура MS SQL Server наведена на рисунку 1.1 Далі розглянемо основні модулі детально [2].

1.2.1 Рівень протоколу (Protocol Layer)

MS SQL Server підтримує чотири протоколи: Спільна пам'ять, TCP/IP, Іменовані конвеєри та Адаптер віртуального інтерфейсу. Спільна пам'ять використовується, коли клієнт і сервер розташовані на одній машині, в той час як TCP/IP є основним протоколом для з'єднання через мережу. Іменовані конвеєри життєздатні, коли дані обмінюються між двома системами в локальній мережі. Нарешті, адаптер віртуального інтерфейсу необхідний для високошвидкісного зв'язку між двома машинами - для цього протоколу потрібне спеціальне обладнання [8].

1.2.2 Обробник запитів (Query Processor)

Завданням обробника є визначення найкращого способу виконання та оптимізації запиту. Він приймає вхідні дані від користувача, запитує дані зі сховища даних та обробляє результати. Обробник складається з трьох компонентів: синтаксичного аналізатора команд, який отримує вхідні дані запиту та здійснює синтаксичний і семантичний аналіз для формування дерева запиту, та оптимізатора, який створює найкращий план виконання запиту. Виконавець виконує запити. Він викликає методи доступу для створення плану виконання для отримання даних. Після того, як дані з механізму зберігання отримані, вони доставляються за допомогою протоколів до передбачуваного одержувача.

1.2.3 Система збереження або Storage Engine

Система зберігання виконує завдання розміщення та отримання даних з пристроїв зберігання або мереж зберігання даних (SAN) [12]. Ця система включає наступні елементи: Access Methods, який працює як зв'язок між виконавцем запиту та менеджером буферів/журналів транзакцій; менеджер буферів, який опікується нечистими сторінками, каталогом планів виконання та інтерпретацією даних; менеджер транзакцій, який виконує обов'язки, пов'язані з межами транзакцій, наглядом за проведенням транзакцій, відновленням після збоїв; менеджер журналів, який архівує виконання та скасування транзакцій; менеджер блокування, який опікується блокуванням даних, що відносяться до транзакції. Це забезпечує ізоляцію та узгодженість даних.

1.3 Структура SQL сервера та збереження даних на сервері.

SQL Server прив'язує базу даних до набору файлів операційної системи, в цих файлах зберігаються об'єкти, таблиці та індекси. База даних SQL складається з двох або більше файлів операційної системи. Кожен файл бази даних має два імені -

логічне та фізичне. Логічне ім'я згадується в операторах T-SQL, а фізичне - це назва файлу в операційній системі.

Файли бази даних можуть зберігатися на файловій системі FAT або NTFS.

Існує три різних типи файлів бази даних, які можуть бути створені. Основним файлом є файл .mdf, який містить інформацію про стиль бази даних, посилання на інші документи та деталі про об'єкти бази даних. Крім того, файли .ndf є вторинними файлами даних, які використовуються для зберігання об'єктів користувача. Вони можуть розташовуватися на декількох окремих дисках і використовуватися як додаткове сховище бази даних. Файли .ldf - це файли журналу транзакцій, в яких реєструються всі зміни в базі даних, і які використовуються у разі необхідності відновлення бази даних. Одна база даних може містити один або декілька файлів журналу транзакцій.

1.4 Аналіз та порівняльна характеристика існуючих рішень серверів баз даних

На ринку сьогодні є багато рішень для імплементації SQL серверів, однак, якщо більш детально зануритись у вибір правильного, то варто детальніше розглянути особливості кожного з них. Далі у розділі наведено порівняльну характеристику різних дистрибутивів для розуміння їх особливостей

1.4.1 Особливості MySQL серверу та його характеристика

MySQL часто є основною СУБД для веб-додатків [3]. Вона є стандартом для операційних систем Linux і може бути отримана безкоштовно або придбана за окрему плату. Безкоштовна версія робить акцент на швидкості та надійності і може не мати певних функціональних можливостей - перевага чи недолік цього залежить від того, для чого вона буде використовуватися. Нові версії випускаються часто з метою підвищення безпеки та розширення функцій.

Корпорація Oracle відповідає за розробку та підтримку MySQL. Коли компанія Sun Microsystems була куплена Oracle, вона отримала права на торгову марку. MySQL

доступна за ліцензією GNU General Public License, а також за комерційною ліцензією. Багато функцій MySQL з'явилися в результаті виконання завдань на замовлення ліцензованими користувачами. Чудовим прикладом цього є механізм реплікації, який був реалізований майже у всіх ранніх версіях.

Ця база даних пропонує різноманітні опції механізму, які оптимізують можливості інструменту та обробку даних для численних типів таблиць. MySQL дуже гнучко підходить до засобів зберігання даних, надаючи користувачам як таблиці MyISAM з повнотекстовим пошуком, так і таблиці InnoDB з підтримкою транзакцій на рівні окремих записів. Крім того, таблиця типу EXAMPLE дозволяє новачкам в MySQL зрозуміти, як створювати нові типи таблиць. Крім того, MySQL може легко додавати нові типи таблиць на свою платформу з відкритою ліцензією GPL. І останнє, але не менш важливе: вона має простий користувальницький інтерфейс і пакетні команди для зручного виконання процесів. Інтерфейс зручний для користувача, а пакетні команди допомагають швидко обробляти величезні обсяги даних. Система є надійною та ефективно використовує всі апаратні ресурси.

Перевагою MySQL є те, що вона безкоштовна, добре документована і має широкий спектр функцій, доступних у її безкоштовній версії. Крім того, її легко встановити на багато з найпоширеніших операційних систем Linux. Крім того, доступно багато різних користувацьких інтерфейсів, а також сумісність з базами даних, такими як DB2 і Oracle. Однак, MySQL може забирати багато часу для виконання простих завдань, порівняно з системами, які повністю автоматизують ці функції, наприклад, створення автоматичних резервних копій. Крім того, функції XML та OLAP не доступні у безкоштовній версії, а пропонується лише платна підтримка.

Добре підходить для малих та великих компаній – стартапів за рахунок безкоштовного доступу що пропонується самимою компанією. Однак якщо потрібна підтримка то за це прийдется платити.

1.4.2 Аналіз та характеристика Microsoft SQL серверу

Програмний продукт Microsoft SQL Server є однією з найпоширеніших СУБД. Для його роботи використовуються як хмарні, так і локальні сервери, причому обидва варіанти можуть використовуватися в комбінації. Одразу після запуску Microsoft SQL Server 2016 компанія Microsoft адаптувала його для системи Linux, а першочергове призначення - для платформи Windows [8].

Версія 2016 Microsoft SQL Server пропонує ексклюзивну функцію підтримки темпоральних даних, що дозволяє відслідковувати будь-які зміни в даних за певний період часу. Крім того, нова версія має можливість динамічного маскування даних, тобто доступ до конфіденційних даних матимуть лише авторизовані користувачі.

Продукт простий у використанні, працює швидко і стабільно, надає можливість налаштовувати та контролювати продуктивність, що допомагає обмежувати ресурси. Крім того, він підтримує доступ з мобільних пристроїв та інтеграцію з іншими продуктами Microsoft. Одним з недоліків є висока вартість для бізнесу, і адміністратори повинні бути обережними з налаштуванням продуктивності, оскільки це може споживати ресурси. Також повідомлялося про проблеми при використанні сервісу інтеграції для імпорту файлів. Якщо компанія вже використовує продукти Microsoft, то придбання ліцензії може бути гарною ідеєю. Нарешті, цей продукт найкраще підійде великим компаніям.

Варто відмітити, що дуже гарно підійде для великих Enterprise проектів у разі, якщо компанія вже використорує продукти ти мікросервіси від Microsoft.

1.4.3 PostgreSQL сервер та його характеристика

PostgreSQL - це відома та безкоштовна СУБД, яка використовується для зберігання даних веб-сайтів. Це зріла система, що дозволяє користувачам управляти як структурованою, так і неструктурованою інформацією. Вона працює на багатьох основних операційних системах, з особливо хорошою продуктивністю на Linux. Імпорт даних з інших баз даних може бути легко здійснений за допомогою власних

інструментів. Вона може бути встановлена в різних конфігураціях, наприклад, на фізичних, віртуальних і хмарних платформах. Остання версія, PostgreSQL 9.5, пропонує підвищену швидкість обробки даних та кількість одночасних користувачів а за допомогою додавання спеціального модулю DBMS_SESSION була покращена секьюрність даних на сервері.

Це рішення ідеально підходить для організацій з обмеженим бюджетом, які потребують підтримки системи, що працює з JSON. Основними перевагами є масштабованість, заздалегідь визначені функції та безліч інтерфейсів. Однак, документи не надають достатньо інформації, а конфігурація може бути складною. Крім того, він може сповільнюватися під час пакетних операцій або запитів на читання. У таких випадках може знадобитися залучення експертів для правильного налаштування системи та вибору певного інтерфейсу.

1.4.4 MariaDB сервер та його характеристика

Ця СУБД доступна безкоштовно, однак, як і інші безкоштовні СУБД, оновлення доступні за окрему плату. Доступна велика кількість плагінів розширення, і це, безумовно, найбільш швидкозростаюча СУБД на даний момент.

MariaDB була створена у зв'язку з необхідністю захисту вільного статусу бази даних на противагу системі ліцензування Oracle для MySQL. Для додавання коду в основну гілку бази даних, автори повинні передати свої авторські права фонду MariaDB Foundation, щоб підтримувати ліцензію і дозволяти робити необхідні патчі для MySQL. Вона заснована на MySQL, випускається під ліцензією GNU GPL, підтримується і розвивається MariaDB Corporation Ab і MariaDB Foundation.

Різноманітність систем зберігання даних, доступних в ядрі бази даних, сприяє більш оптимізованому використанню ресурсів та швидшій обробці запитів та виконання. На заміну InnoDB як основної системи зберігання даних, MariaDB має підсистеми зберігання даних XtraDB.

Цей проект забезпечує значну сумісність з MySQL, включаючи заміну функцій на двійкові еквіваленти бібліотек та збереження відповідності з існуючими API та

командами MySQL. Перша захищена версія бази даних була включно з номером 5.2, ніби наслідуючи архітектуру MySQL. MariaDB включає в себе два сильно просунутих движка баз даних: Aria, раніше Maria, для зміни MyISAM і XtraDB для заміни InnoDB (до версії 10.1 включно). Колекція MariaDB 5.2.x сумісна з точки зору функціональності з MySQL 5.1.x і базується на тій самій кодовій базі, що дозволяє мігрувати з MySQL на MariaDB без проблем із сумісністю. Зберігаючи зворотну сумісність, MariaDB включає в себе підтримку декількох свіжих сховищ (FederatedX, PBXT, XtraDB, Aria, OQGRAPH, S).

2 ЗАСОБИ РЕАЛІЗАЦІЇ ПРОГРАМНОЇ СИСТЕМИ

Першим кроком є вибір інструментів програмування, які будуть використовуватися, їх дослідження та зважування їх переваг та недоліків для того, щоб втілити програму в життя.

2.1 Опис Javascript як засобу для створення програмних додатків

Аналізуючи засоби розробки, дослідники зупинилися на мові програмування JavaScript. До переваг вибору JavaScript можна віднести:

1. Життєздатність у веб-розробці, оскільки це технологія "go-to" для клієнт-серверних моделей та веб-програмування.

2. Сумісність з дизайном сторінок і хостингом ще більше спрощує використання.

3. Ефективність і зручність використання - прості завдання можуть бути виконані за лічені хвилини, а синтаксис простий для розуміння.

4. Speed-JS сприяє отриманню більш швидкого, більш повного кінцевого продукту.

6. Комфорт користувальницького інтерфейсу з власним потужним середовищем.

Наявність власної потужної екосистеми (інфраструктури). Особливо помітно останніми роками. Приклад – поява величезної кількості корисних фреймворків, які підійдуть будь-якому випадку.

2.2 Порівняльна характеристика бібліотек для розробки користувацького інтерфейсу

Наразі найбільш популярними бібліотеками на ринку є React, Vue, Angular. Однак для більш детального розуміння переваг та недоліків ми розглянемо як вони себе поводять по основних параметрах, на які перш за все звертають увагу розробники [9].

2.2.1 Особливості рендерингу компонентів інтерфейсних бібліотек

JavaScript фреймворки насамперед варто порівняти за рендерингом сторінки. Сучасна архітектура допускає два види: за клієнта (сторінка відмальовується з допомогою потужностей ПК користувача) чи за сервера.

Модель об'єктів документа (Document Object Model, DOM) забезпечує інтерфейс для читання і маніпулювання вмістом, представленням і структурою HTML-документів. І Vue.js, і React створюють репліку DOM, обробляють її і переглядають відмінності між оригінальною і обробленою формою. Те, що з'явиться на екрані користувача, буде комбінацією оригінального і обробленого результатів, які замінюють будь-які відмінності між ними [7].

Angular версії 1.x або вище прискорює завантаження та відображення, що є важливим фактором для людей, які використовують портативні пристрої. Це досягається за рахунок поділу завдань для DOM між двома потоками, де браузер (клієнтська частина) відповідає за створення представлення DOM, а інший потік (серверна частина) піклується про директиви, код і сервіси. Незважаючи на це, фактичне відображення сторінки все одно буде здійснюватися на сервері, що робить SEO-оптимізацію простою для роботів пошукових систем.

2.2.2 Архітектура інтерфейсних компонентів при роботі з фреймворками

React - це щось середнє між бібліотекою та фреймворком; він розроблений з урахуванням структури MVC (Model-View-Controller), де модель надає інформацію, представлення відображає ці дані користувачеві, а контролер інтерпретує взаємодію з користувачем і викликає зміни в моделі.

Angular та Vue.js посилаються на відповідні фреймворки. Якщо архітектура побудована на React, це означає: пошук і використання додаткових бібліотек для виконання різних завдань; коригування функціональності програми відповідно до конкретної бібліотеки; і потенційно більш складне залучення інших розробників до проекту через розбіжності в бібліотеках, що використовуються кожною програмою.

Це вимагатиме більше часу для виконання архітектури. Низькорівневий API передбачає занадто багато налаштувань. При використанні усталених фреймворків, таких як Vue.js та Angular, не виникає проблем з підбором та налаштуванням бібліотек під різноманітні завдання. Високорівневий API забезпечує зворотну сумісність всіх бібліотек, а це означає, що сторонній програміст може підключитися до виконання завдання без необхідності вичерпного дослідження архітектури додатку. Привабливість комплексних фреймворків походить від стандартизації діяльності.

2.2.3 Назначення та особливості інтерфейсних фреймворків

React та Vue.js підтримують лише односторонню передачу даних. При цьому React об'єкти інкапсульовані. Простіше кажучи, кожен програмний об'єкт - це, по суті, останній крок в автоматизованому процесі, який не потребує подальшого втручання користувача до його завершення. React дозволяє створювати резервні копії та переносити стан програми, тобто її параметри можуть бути продубльовані на інший пристрій, якщо додаток запускається з нього. Таким чином, на обох екранах з'явиться однакова "картинка". Vue.js працює дещо інакше, його компоненти працюють на основі шаблонів і створюють звичайний HTML, а також підтримується JSX для полегшення переходу з React тощо. У фреймворках логіка процесу автоматизована завдяки побудові інструкцій. Для додаткового функціоналу достатньо додати необхідні директиви. Наприклад, потрібно зробити блоковий елемент, який рухається по сторінці. Для цього достатньо додати у властивості об'єкта директиву draggable, після чого його можна буде перетягнути на екран за допомогою миші.

Робота з Angular відрізняється. Він передачає все, що властиве Vue.js, проте опис взаємодії об'єктів відбувається у службах, що є складовими частинами модулів.

Модульна архітектура більш зручна при створенні великих додатків. Цей модуль був створений для вирішення багатьох завдань для цієї функції. Таким чином, кінцевий розмір коду стає меншим, а швидкість обробки стає вищою.

Підтримка представлення моделі дозволяє використовувати один набір даних для вирішення різних завдань в одній частині програми. Функціональні залежності визначають двонаправленість передачі. Кожна програма може запускати інший процес.

2.3 Вибір React як бібліотеки для створення інтерфейсу програми

Звісно, порівняти фреймворки абсолютно об'єктивно досить важко, але аналіз дозволить розробнику підібрати платформу. Підведемо підсумки:

- Якщо потрібно швидко вивчити середовище то розробники більше схиляють вибір до React за рахунок його архітектурних особливостей та низького порогу входу.
- Якщо з часом до команди будуть залучатись інші програмісти, то Vue.js стане найкращим вибором. Адже бібліотека дуже легка у вивченні та для комфортної розробки інженеру не буде потрібно багато часу щоб освоїти технологію.
- Якщо для проекту передбачається багатоступінчасте оновлення та до кінця не визначений функціональний набір інструментів або передбачуваних сервісів то варто зупинити свій вибір на React за рахунок його гнучкості.

Свій вибір ми робимо на користь React, адже за рахунок його гнучкості у використанні ми маємо на багато більшу кількість можливостей при рішенні нашої задачі.

2.4 Опис технології Electron як засобу для розробки кросплатформенних настільних застосунків

Electron – це фреймворк для розробки настільних програм з використанням HTML, CSS та JavaScript. Такі програми можуть працювати на різних платформах. Серед них – Windows, Mac та Linux.

У основі Electron лежать проекти Chromium та Node.js, об'єднані в єдине середовище, що забезпечує роботу програм. Це дає можливість використовувати веб-технології при розробці настільних програм.

Electron - серйозний проект, який використаний при створенні багатьох популярних додатків. Серед них – месенджери Skype та Discord, редактори для коду Visual Studio Code та Atom, а також ще більше 700 додатків, відомості про які опубліковані на сайті Electron.

Для розробки програми з використанням Electron цей фреймворк потрібно налаштувати. Це стосується й тих випадків, коли в додатку планується застосовувати інші фреймворки або бібліотеки, наприклад, Angular, React, Vue, або щось інше.

Інструмент командного рядка Electron Forge дозволяє серйозно спростити процес налаштування Electron. Він дає у розпорядження розробника шаблони додатків, заснованих на Angular, React, Vue, та інших фреймворках. Це позбавляє програміста необхідності налаштовувати все вручну.

Крім того, Electron Forge спрощує складання та упаковку програми. Насправді, цей засіб має і багато інших корисних можливостей, дізнатися про які можна з його документації.

Спроектована архітектура розробленого додатку виглядає наступним чином на Рисунку 2.1 у вигляді схеми.

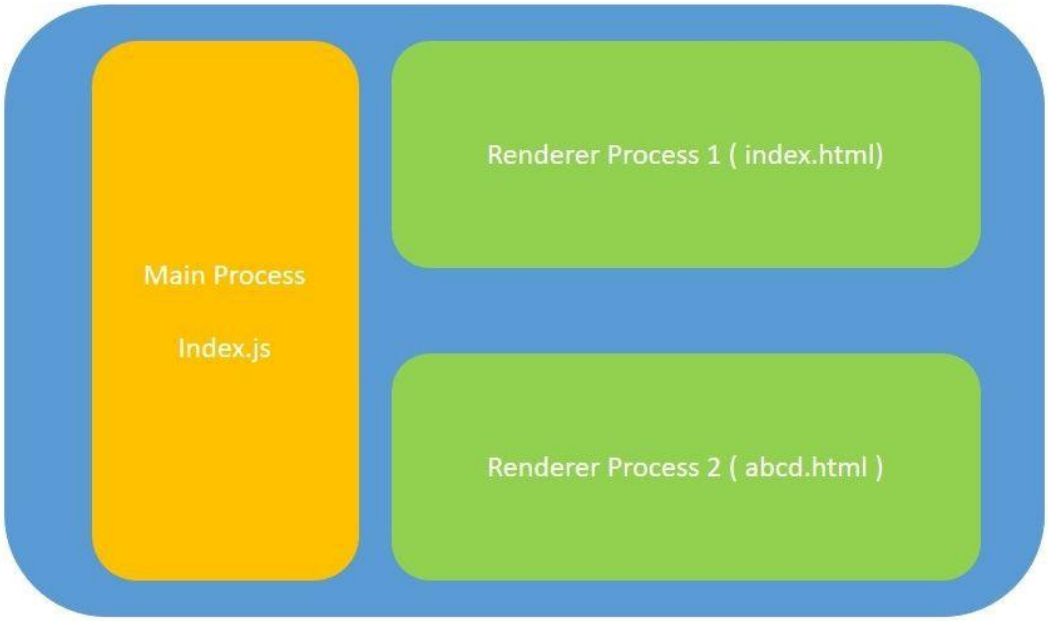


Рисунок 2.1 – Архітектурний шаблон Electron застосунку

На цій схемі показано дві веб-сторінки – index.html та abcd.html. У прикладі буде використовуватися лише одна сторінка, представлена файлом index.html, адже фронтенд буде написаний єдиним фронтендом написаним на React.

3 ПРОЕКТУВАННЯ ПРОГРАМНОЇ РЕАЛІЗАЦІЇ

У наступному розділі деталізовано план, згідно якого ми будемо розробляти встановлений у попередніх розділах настільний додаток, що являє собою клієнт для управління базами даних.

3.1 Функціонал

При роботі з сервером реляційної бази даних SQL для маніпуляцій з даними розробниками був розроблений спеціальний синтаксис T-SQL. Цей синтаксис передбачає набір інструкцій за допомогою яких розробник може з легкістю виконувати запити до сервера, щоб отримати або редагувати первні дані а також мати змогу редагувати саму структуру бази враховуючи реляційні зв`язки.

Передбачуваний функціонал:

- Підключення до декількох серверів баз даних;
- Налаштування підключення;
- Редагування підключення;
- Розробка зручних налаштувань додатку;
- Можливість виконувати SQL запити до серверу;
- Перегляд даних в таблицях;
- Редагування даних таблиць;
- Копіювання результатів виконання скриптів;
- Імпорт/експорт даних виконаного скрипта.

3.2 Керування скриптами

Для того щоб працювати з T-SQL скриптами нам необхідно буде розробити модуль з полем для вводу, редагування та збереження скриптів для подальшого використання у вигляді SQL файлів.

На Рисунку 3.1 зображена діаграма прецедентів при роботі користувача з вище описаним модулем керування T-SQL скриптами

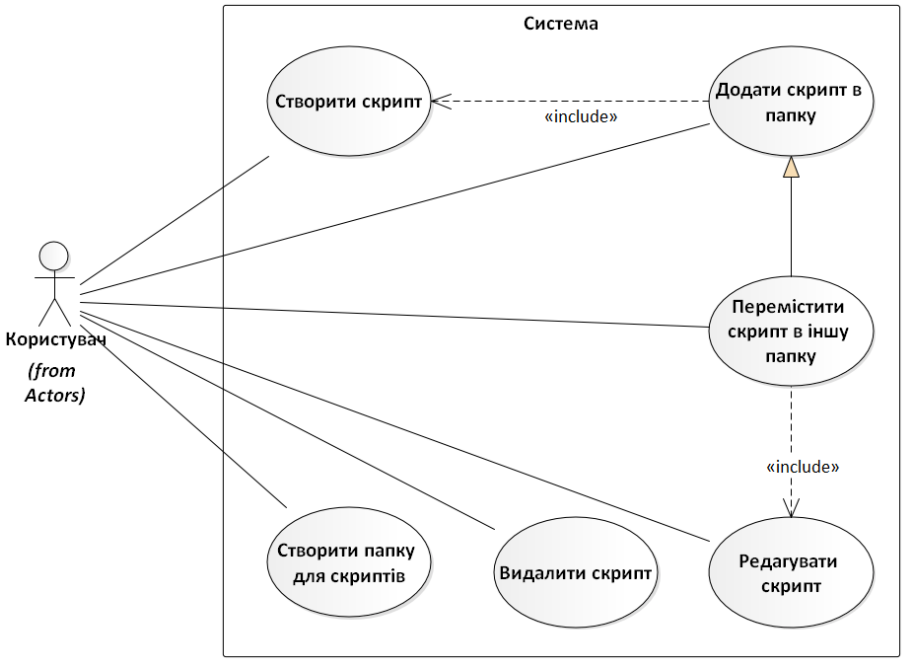


Рисунок 3.1 – Прецедент керування T-SQL скриптами

Виконання T-SQL скриптів є однією з найважливіших частин системи. На його основі працюватимуть основні функції обслуговування та моніторингу сервера баз даних. Користувач також зможе запускати скрипти на виконання вручну, обравши з'єднання та вказавши параметри. З'єднання вибирається з тих, що зберігаються в системі, а база даних - з тих, що існують на підключеному сервері. Параметри вказуються відповідно до тих, що визначені в моделі сценарію, при цьому автоматично будуть застосовані значення за замовчуванням.

Результати роботи сценаріїв, що виконуються користувачем вручну, будуть відображатися на графічному інтерфейсі у вигляді таблиць. Стовпці таблиць будуть відповідати набору даних, що повертається сервером. Також результат може складатися з декількох таблиць і користувач зможе працювати з кожною з них.

Якщо під час виконання скрипта сервер повертає помилку, система повинна відобразити помилку користувачеві.

Нижче детально описаний прецедент виконання скрипту всередині додатку(див. Таблиця 3.1).

Таблиця 3.1

Прецедент виконання скрипту

Користувач	Система
1. Користувач обирає зі списку доступних серверів та виконує запит на підключення до нього.	2. Система відправляє запит по параметрам ODBC та повертає відповідь на клієнт з статусом запиту.
3. Обирає базу даних і далі працює з даними, отриманими шляхом подальших запитів на читання бази.	4. Виконує подальші запиту. Створює відповіді на відповідні запити до сервера.

3.3 Підключення сервера SQL

ODBC (англ. Open Database Connectivity) — це програмний інтерфейс (API) доступу до баз даних, розроблений компанією Microsoft у співпраці з Simba Technologies на основі специфікацій Call Level Interface (CLI), який розробляли організації SQL Access Group, X/Open і Microsoft . Згодом CLI було стандартизовано ISO. Стандарт CLI покликаний уніфікувати програмну взаємодію з СУБД, зробити її незалежною від постачальника СУБД та програмно-апаратної платформи.

На початку 1990 р. існувало кілька постачальників баз даних, кожен із яких мав власний інтерфейс. Якщо програмі потрібно було спілкуватися з кількома джерелами даних, для взаємодії з кожною з баз даних необхідно було написати свій код. Для вирішення проблеми Microsoft і ряд інших компаній створили стандартний інтерфейс для отримання та відправки джерелам даних різних типів. Цей інтерфейс був названий Open Database Connectivity або відкритий механізм взаємодії з базами даних.

За допомогою ODBC прикладні програмісти могли розробляти програми для використання одного інтерфейсу доступу до даних, не турбуючись про тонкощі взаємодії з кількома джерелами.

Це досягається завдяки тому, що постачальники різних баз даних створюють драйвери, що реалізують конкретне наповнення стандартних функцій з ODBC API з урахуванням особливостей їхнього продукту. MFC удосконалила ODBC для розробників додатків. Справжній інтерфейс ODBC є звичайним процедурним API. Замість створення простої оболонки процедурного API, розробники MFC створили набір абстрактних класів, що представляють логічні сутності в базі даних.

Далі на Рисунку 3.2 буде відображена діаграма прецедентів відносно керування підключенням до серверів

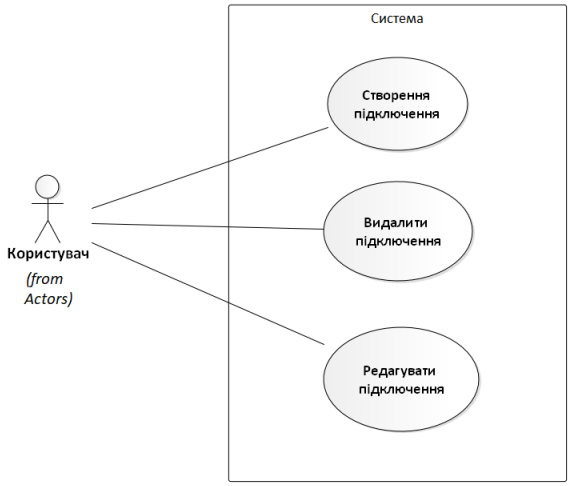


Рисунок 3.2 – Прецедент керування підключенням серверу через ODBC

3.4 Системна модель

Найкращим типом системи для цього є веб-архітектура. Таким чином, до неї можуть отримати доступ користувачі в мережі. Наявність одного набору скриптів, до яких може отримати доступ багато користувачів, економить час і енергію. Крім того, наявність лише одного екземпляру програми гарантує, що доступ до цільового сервера бази даних буде здійснюватися точно і без будь-яких конфліктів. У такий спосіб може бути забезпечений високий рівень безпеки доступу до підключених серверів.

Веб-інтерфейс надасть користувачам доступ до функцій системи через веб-браузер. Компоненти включатимуть розмітку сторінок та методи доступу до сервера сервісної системи, які дозволять користувачам керувати системою.

Сервер системи обслуговування включатиме методи доступу до даних системи, логіку підключення до бази даних, що обслуговується, та виконання сценаріїв. Він також буде планувати виконання скриптів за заданими користувачем параметрами.

Сервер бази даних буде зберігати системні дані, у тому числі скрипти, параметри підключення та налаштування планування завдань. Вище описана модель зображена на Рисунку 3.3.

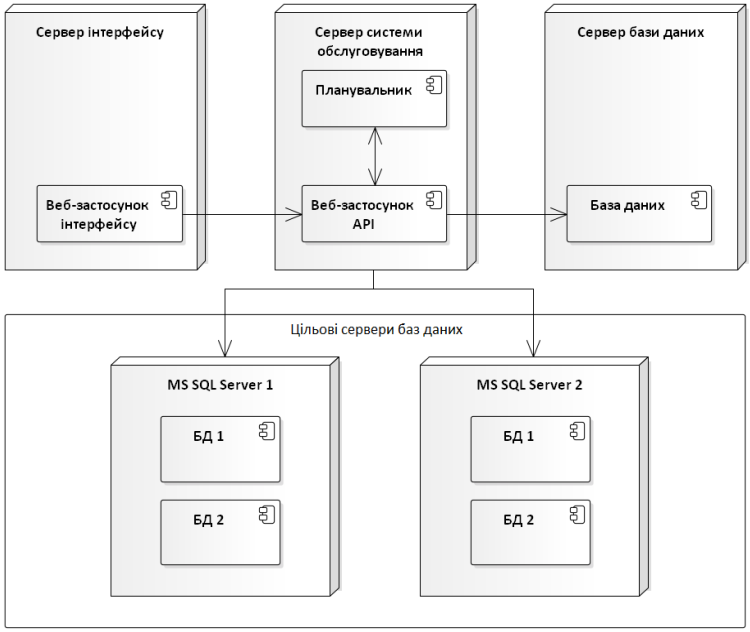


Рисунок 3.3 – Модель системи

3.5 Керування параметрами додатку

За допомогою бібліотеки фреймворку electron-json-storage будемо зберігати дані користувача про підключення у файлі розширення JSON (так званий storage), адже для такого простого інтерфейсу даного інструменту нам буде достатньо.

З точки зору програмної реалізації бібліотека пропонує нам наступний набір можливостей відображений на Рисунку 3.4 [2].

- storage
  - .getDefaultDataPath() ⇒ `String` | `Null`
  - .setDataPath(directory)
  - .getDataPath() ⇒ `String`
  - .get(key, [options], callback)
  - .getSync(key, [options])
  - .getMany(keys, [options], callback)
  - .getAll([options], callback)
  - .set(key, json, [options], callback)
  - .has(key, [options], callback)
  - .keys([options], callback)
  - .remove(key, [options], callback)
  - .clear([options], callback)

Рисунок 3.4 – Інструменти бібліотеки electron-json-storage

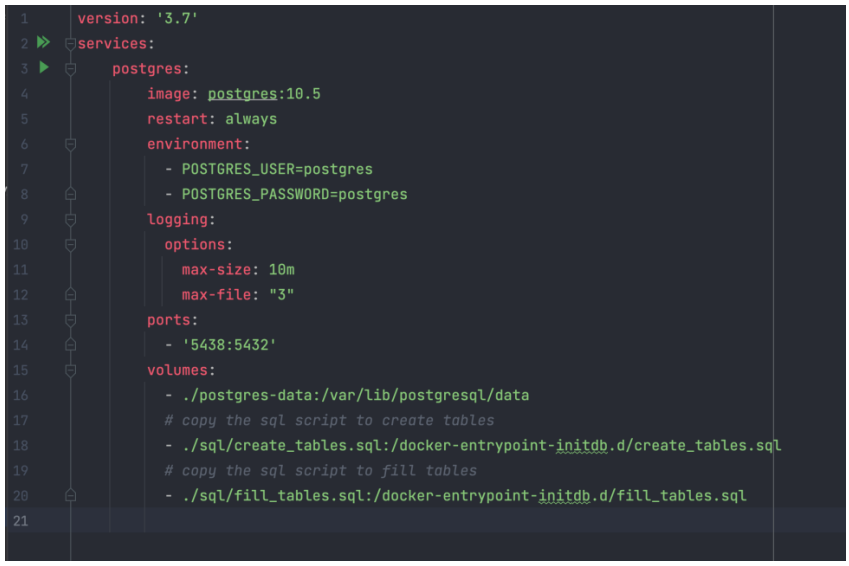
Серед функцій виділяємо ключові:

- .getDefaultDataPath() ⇒ String | Null – отримання шляху, де знаходиться конфіг;
- .setDataPath(directory) – встановлення шляху конфігу;
- .get(key, [options], callback) – отримання даних з сховища по ключу;
- .getSync(key, [options]) – оновлення стану даних конфігу (використовується при ініціалізації додатку;
- .set(key, json, [options], callback) – встановлення атрибуту з даними закріпленими за ключем;
- .remove(key, [options], callback) – видалення даних по ключу та самого ключа з конфіга;

3.6 Розгортання тестових серверів баз даних

Для того щоб тестувати підключення та виконувати маніпуляції з базами даних, використовуючи Docker, ми можемо запустити декілька локальних серверів і в подальшому використовувати їх.

Сервери Docker містять готові шаблони для запуску серверу, тож створивши Docker Compose файл з інструкціями на Рисунку 3.5. запускаємо сервер у себе на локальній машині.



```
version: '3.7'
services:
    postgres:
        image: postgres:10.5
        restart: always
        environment:
          - POSTGRES_USER=postgres
          - POSTGRES_PASSWORD=postgres
        logging:
          options:
            max-size: 10m
            max-file: "3"
        ports:
          - '5438:5432'
        volumes:
          - ./postgres-data:/var/lib/postgresql/data
          # copy the sql script to create tables
          - ./sql/create_tables.sql:/docker-entrypoint-initdb.d/create_tables.sql
          # copy the sql script to fill tables
          - ./sql/fill_tables.sql:/docker-entrypoint-initdb.d/fill_tables.sql

```

Рисунок 3.5 – Docker файл для запуску серверу Postgres

У конфігураційному файлі визначені змінні середовища, де зконфігуровано користувача та порти запущеного сервера, таким чином після цього по DNS нашої локальної машини ми зможемо виконати підключення до сервера через параметри ODBC за посиланням 0.0.0.0:5438 з USER=postgres та PASSWORD=postgres.

Після виконання скрипта, відкривши настільний застосунок Docker, можемо переконатися, що контейнер з сервером створено та запущено (див. Рисунок 3.6)

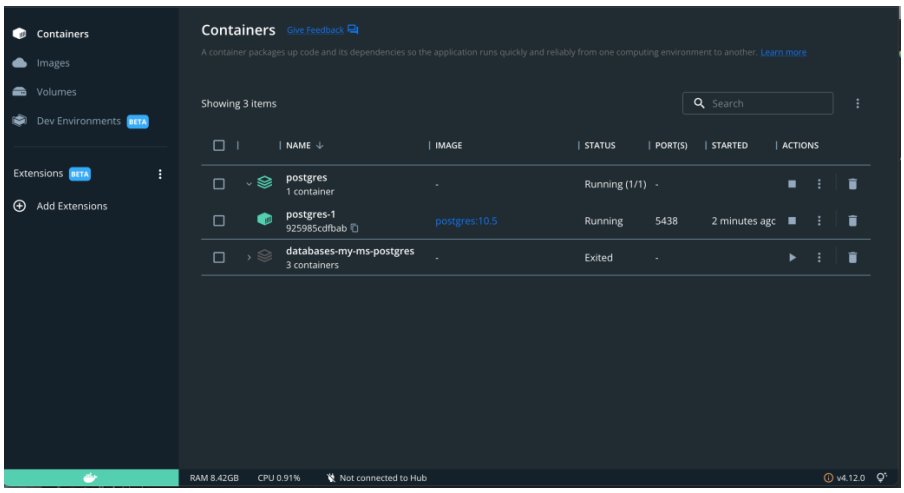


Рисунок 3.6 – Статус запущеного контейнера з базою даних у застосунку Docker

Далі створимо тестову базу для роботи з нею. На Рисунку 3.7 зображена база даних для роботи з сервером, яка складається з 8 таблиць.

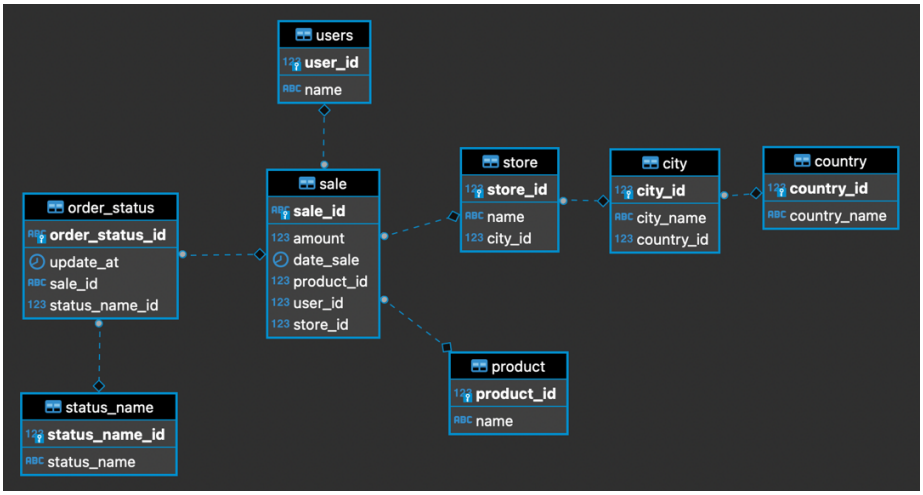


Рисунок 3.7 – База даних тестування

Приклади створення основних таблиць відображено на Рисунках 3.8-3.14.

```sql
-- Creation of product table
CREATE TABLE IF NOT EXISTS product (
  product_id INT NOT NULL,
  name varchar(250) NOT NULL,
  PRIMARY KEY (product_id)
);
```

Рисунок 3.8 – Створення таблиці продуктів

```sql
-- Creation of country table
CREATE TABLE IF NOT EXISTS country (
  country_id INT NOT NULL,
  country_name varchar(450) NOT NULL,
  PRIMARY KEY (country_id)
);
```

Рисунок 3.9 – Створення таблиці країн

```sql
-- Creation of city table
CREATE TABLE IF NOT EXISTS city (
  city_id INT NOT NULL,
  city_name varchar(450) NOT NULL,
  country_id INT NOT NULL,
  PRIMARY KEY (city_id),
  CONSTRAINT fk_country
      FOREIGN KEY(country_id)
      REFERENCES country(country_id)
);
```

Рисунок 3.10 – Створення таблиці міст

```sql
-- Creation of store table
CREATE TABLE IF NOT EXISTS store (
  store_id INT NOT NULL,
  name varchar(250) NOT NULL,
  city_id INT NOT NULL,
  PRIMARY KEY (store_id),
  CONSTRAINT fk_city
      FOREIGN KEY(city_id)
      REFERENCES city(city_id)
);
```

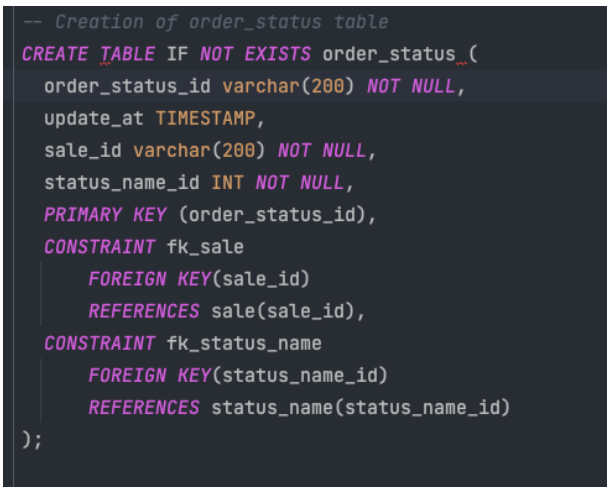
Рисунок 3.11 – Створення таблиці магазину

```sql
-- Creation of status_name table
CREATE TABLE IF NOT EXISTS status_name (
  status_name_id INT NOT NULL,
  status_name varchar(450) NOT NULL,
  PRIMARY KEY (status_name_id)
);
```

Рисунок 3.12 – Створення таблиці статусів замовлення

```sql
-- Creation of sale table
CREATE TABLE IF NOT EXISTS sale (
  sale_id varchar(200) NOT NULL,
  amount DECIMAL(20,3) NOT NULL,
  date_sale TIMESTAMP,
  product_id INT NOT NULL,
  user_id INT NOT NULL,
  store_id INT NOT NULL,
  PRIMARY KEY (sale_id),
  CONSTRAINT fk_product
      FOREIGN KEY(product_id)
      REFERENCES product(product_id),
  CONSTRAINT fk_user
      FOREIGN KEY(user_id)
      REFERENCES users(user_id),
  CONSTRAINT fk_store
      FOREIGN KEY(store_id)
      REFERENCES store(store_id)
);
```

Рисунок 3.13 – Створення таблиці замовлення



```sql
-- Creation of order_status table
CREATE TABLE IF NOT EXISTS order_status (
  order_status_id varchar(200) NOT NULL,
  update_at TIMESTAMP,
  sale_id varchar(200) NOT NULL,
  status_name_id INT NOT NULL,
  PRIMARY KEY (order_status_id),
  CONSTRAINT fk_sale
      FOREIGN KEY(sale_id)
      REFERENCES sale(sale_id),
  CONSTRAINT fk_status_name
      FOREIGN KEY(status_name_id)
      REFERENCES status_name(status_name_id)
);
```

Рисунок 3.14 – Створення таблиці статусу замовлення

4 РОЗРОБКА СПРОЕКТОВАНОГО НАСТІЛЬНОГО ДОДАТКУ

У попередньому розділі було розглянуто основний функціонал, передбачуваний технічним завданням, де ми визначили основний функціонал додатку та перечислили основні функції згідно з вимогами до кінцевого продукту.

У даному розділі запропоновано описаний вище функціонал та користувацький інтерфейс настільного додатку та його реалізацію з використанням технологій описаних попередньо.

На Рисунку 4.1 зображено головний екран додатку.

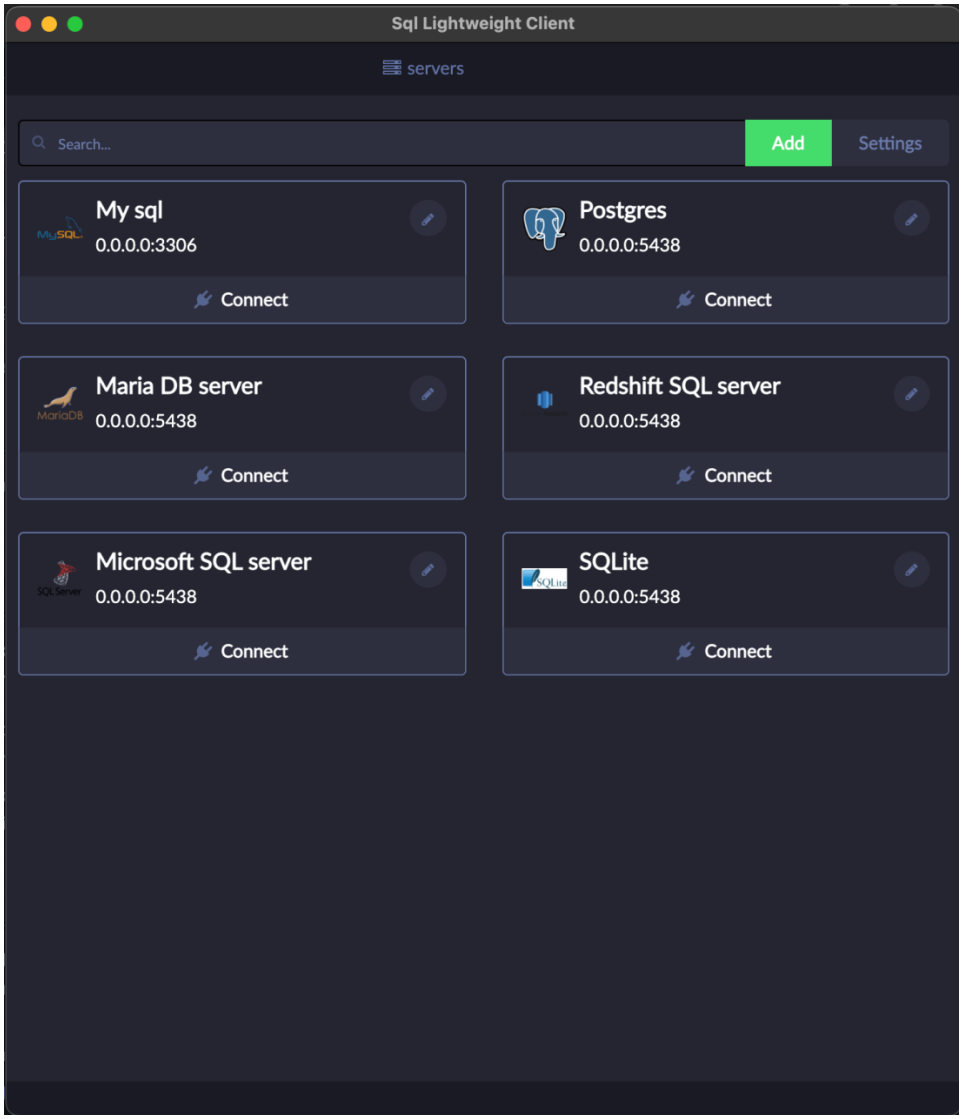


Рисунок 4.1 - Головний екран додатку

На головному екрані у верхній частині є поле пошуку, яке дозволяє користувачу швидко відфільтрувати та знайти необхідний сервер з переліку доданих у конфігураційний файл electron-local-storage (див. Рисунок 4.2).

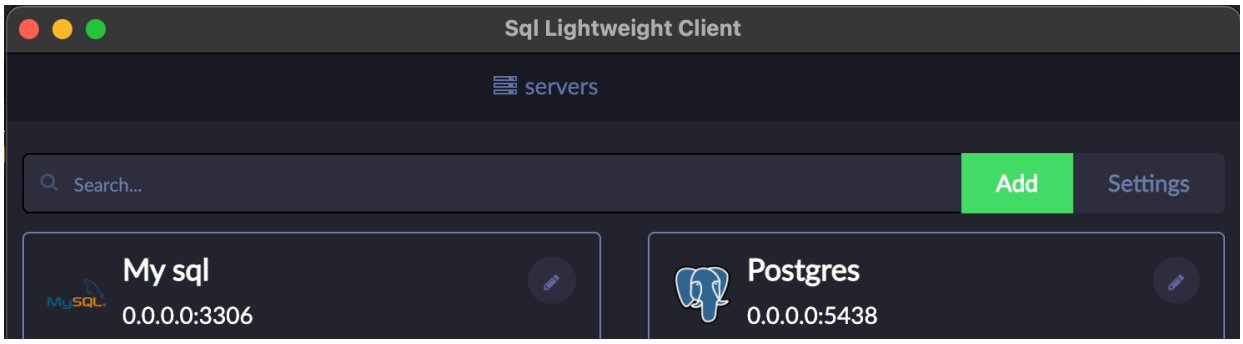


Рисунок 4.2 – Поле пошуку доданих серверів

Справа від поля пошуку розташована кнопка додавання нового серверу до списку для подальшого швидкого підключення до нього (див. Рисунок 4.3).

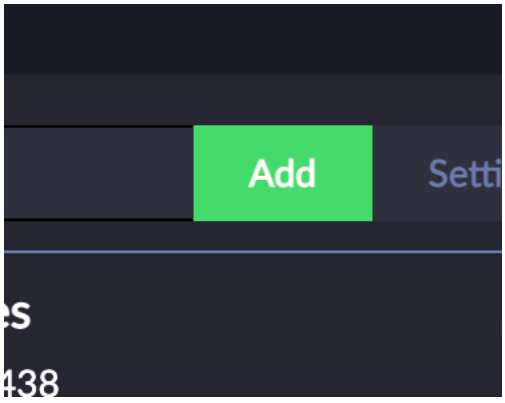


Рисунок 4.3 – Кнопка додавання нового підключення до серверу

При натисканні на кнопку перед користувачем відображається вікно, зображене на Рисунку 4.4 – створення нового підключення до серверу з набором усіх обов`язкових та додаткових параметрів підключення до SQL серверу по ODBC параметрах.

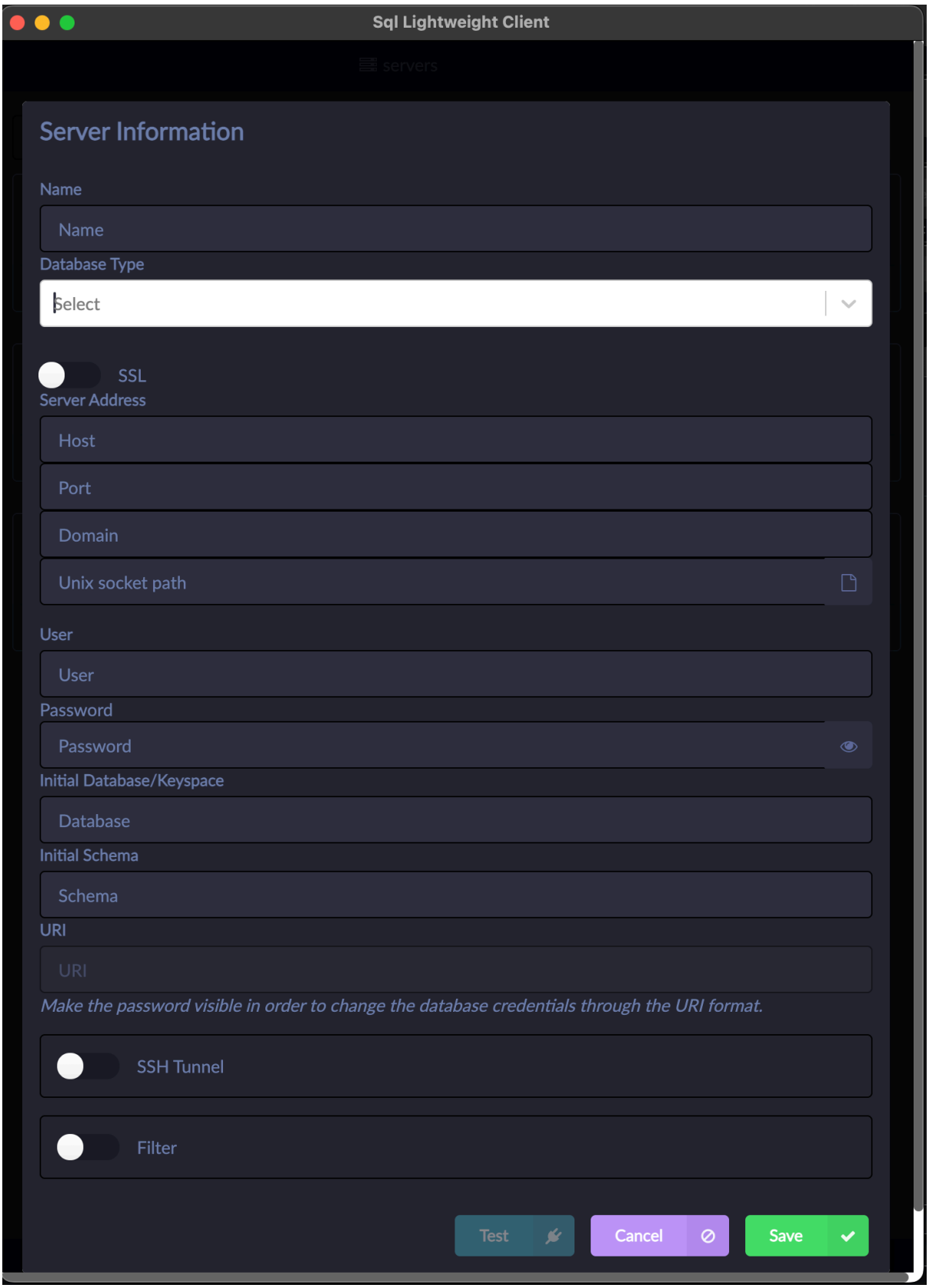


Рисунок 4.4 – Форма створення нового підключення до серверу

Після заповнення форми даними, як на Рисунку 4.5, розблоковується кнопка Test для перевірки підключення до серверу по заповненим атрибутам форми, а саме:

користувач, пароль, порт серверу, та публічна адреса (так як сервер запущено локально і DNS машини = 0.0.0.0 то його і вказуємо як Server Address.).

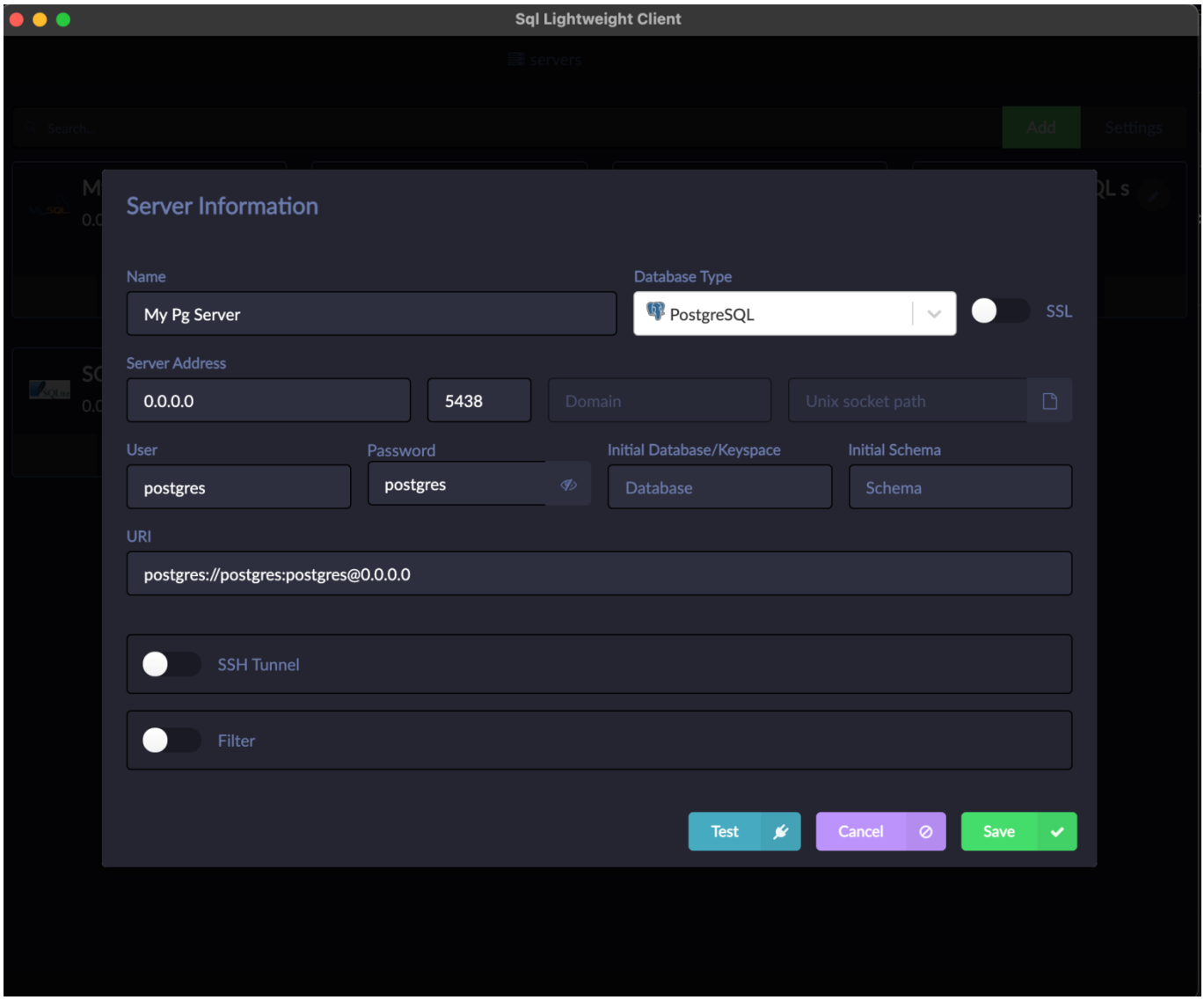


Рисунок 4.5 – Заповнена форма підключення

Натискання на кнопку виконує POST запит по REST API через попередньо встановлену бібліотеку electron-db-core на підключення до серверу і в залежності від коду відповіді у верхній частині екрану буде відображено статус виконаного підключення.

На Рисунку 4.6 відображений результат успішного тестового підключення до серверу.

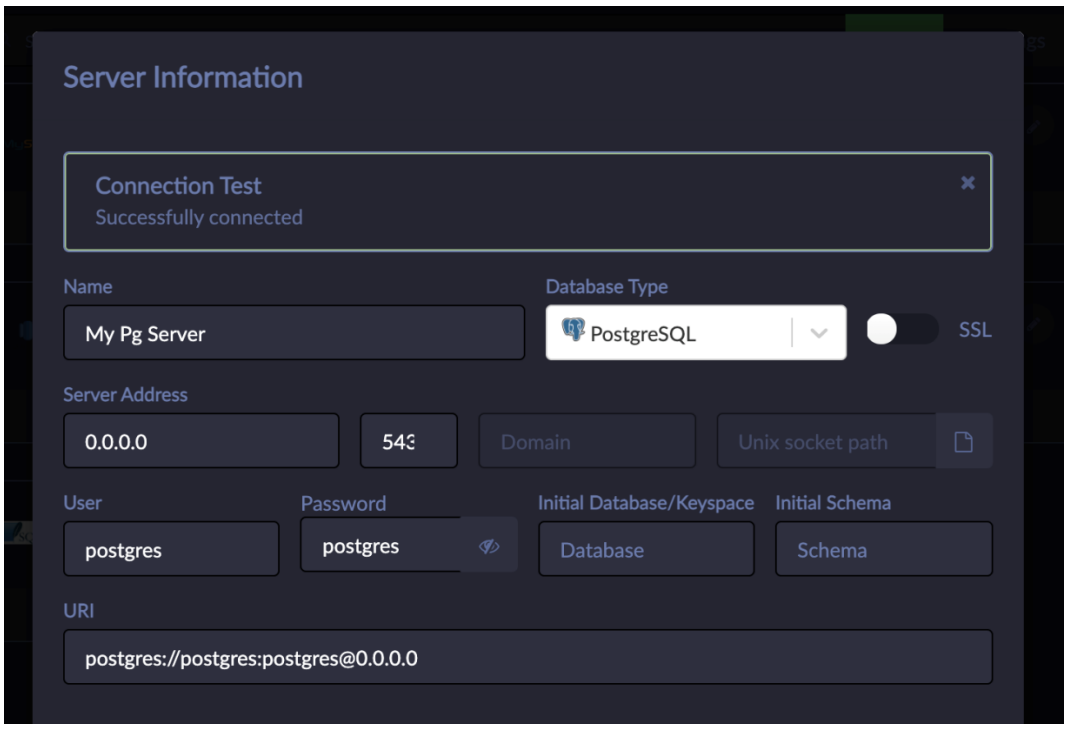


Рисунок 4.6 – Відображення успішного тестового підключення до серверу

У випадку, якщо якийсь з параметрів підключення вказаний невірно, то буде відображене повідомлення як на Рисунку 4.7.

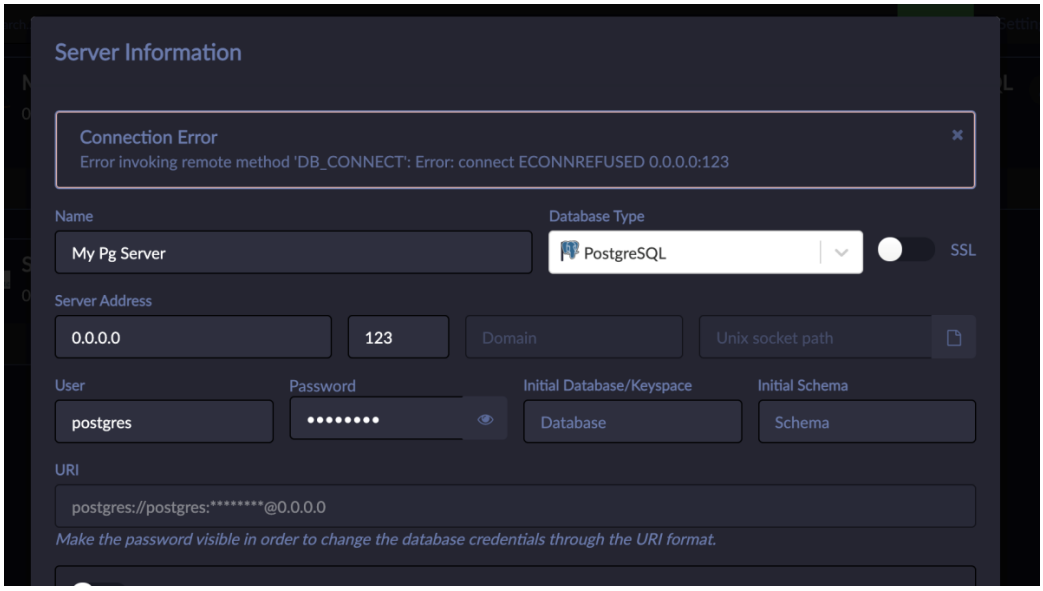


Рисунок 4.7 – Невдале підключення до серверу

На головній сторінці відображена сітка створених підключень, що відображена на Рисунку 4.8.

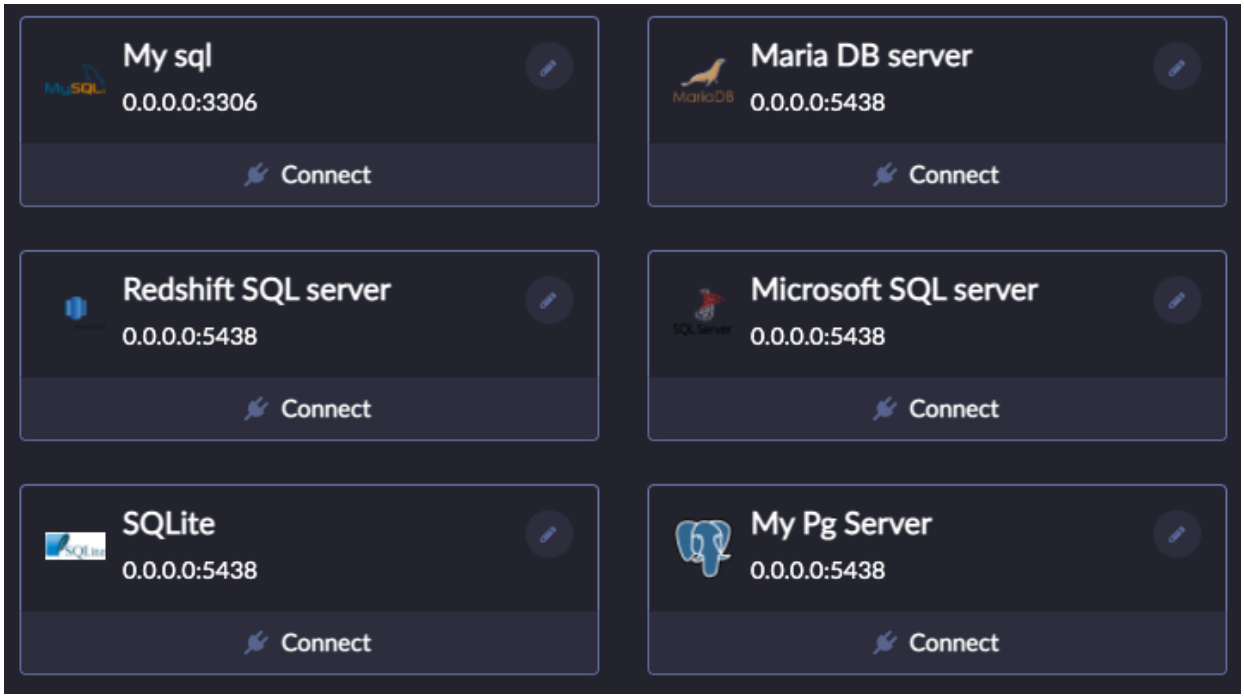


Рисунок 4.8 – Сітка доданих підключень

У конфігураційному файлі, що знаходиться за шляхом /Users/dmytroarkhypenko/Library/Preferences/Sqlectron/sqlectron.json зберігаються дані про попередньо додані сервери в нашому додатку. На Рисунку 4.9 відображені структури даних, що використовуються для підключення до серверів, кожен з яких має наступну схему з полями, серед яких наступні являються обов'язковими.

```
{
 "name": "My sql",
 "client": "mysql",
 "host": "0.0.0.0",
 "port": 3306",
 "user": "root",
 "password": {
    "ivText": "BIjnq8kSdLDe1zHJ62OVpw==",
    "encryptedText": "oUmuC0YxVAe7NqTlmvq/EQ=="
  },
}
```

Варто зазначити, що пароль ми шифруємо для того, щоб у результаті фішенгу зловмисники не могли просто так заволодіти параметрами доступу

Рисунок 4.9 – Файл JSON з збереженими в ньому підключеннями

4.1 Підключення до сервера

Після натискання на кнопку Connect, що зображена на Рисунку 4.10, виконується підключення до серверу після чого через бібліотеку electron-db-core через інкапсульований API ми виконаємо отримання основних компонентів БД.

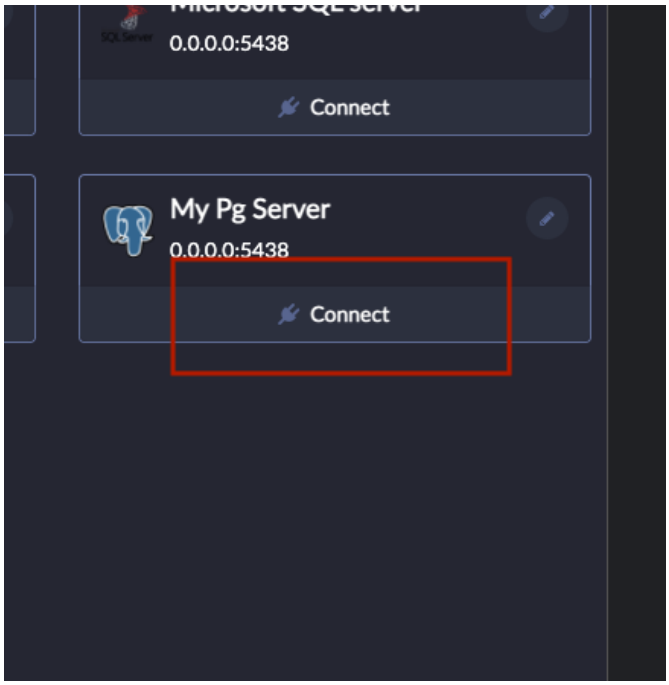


Рисунок 4.10 – Кнопка підключення до серверу

На Рисунку 4.11 показано body POST запиту на підключення

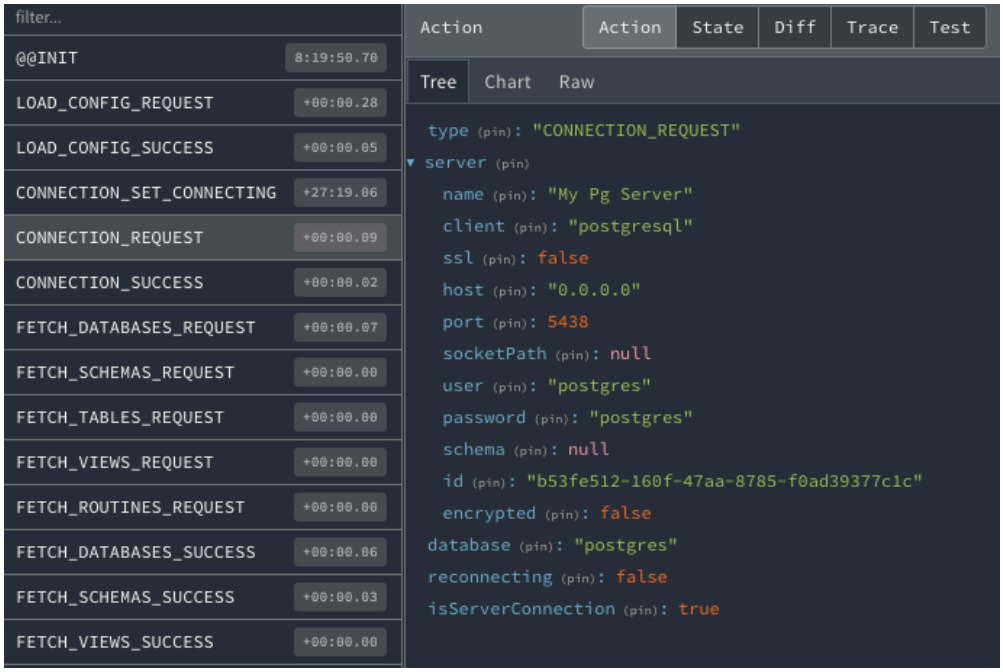


Рисунок 4.11 – Body API запиту на підключення

Успішне виконання даного запиту повертає користувачу клон стану у форматі JSON в теперішньому часі (див. Рисунок 4.12).

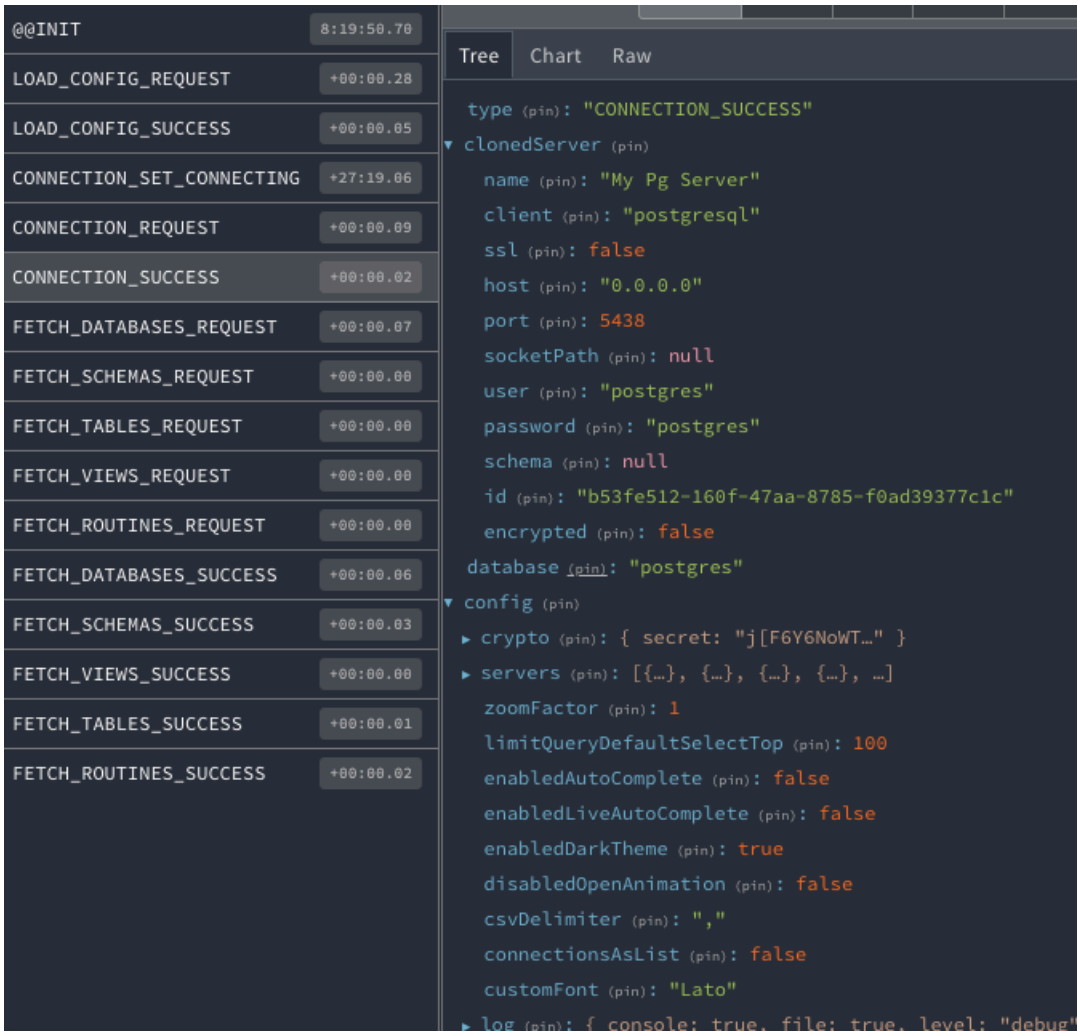


Рисунок 4.12 – Відповідь сервера на підключення з клоном стану сервера

Після виконання запиту йдуть наступні запити на отримання баз даних, схем, діаграм, таблиць функцій та процедур всередині серверу SQL (див. Рисунок 4.13).

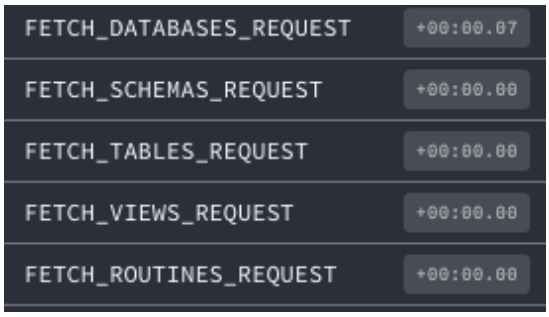


Рисунок 4.13. – Відправлення запитів на отримання наступних даних з серверу

Результат виконання запиту на отримання баз даних на сервері показаний на Рисунку 4.14.

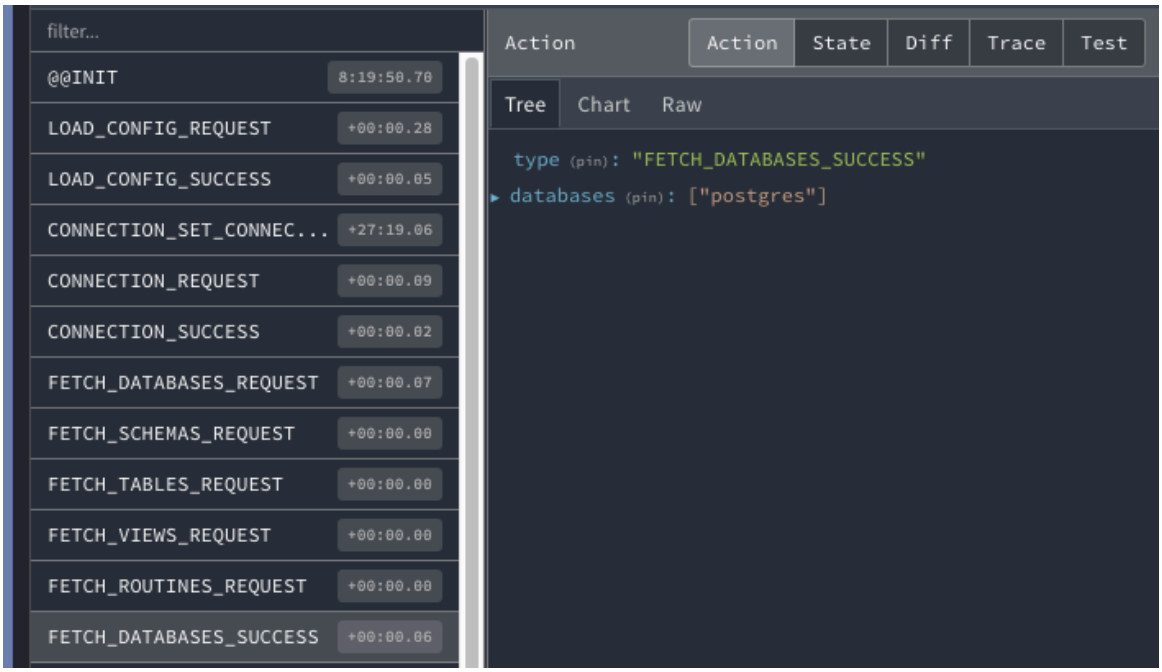


Рисунок 4.14 – Результат виконання запиту на отримання баз даних на сервері

На Рисунку 4.15 відображено результат виконання запиту на отримання схем на сервер.

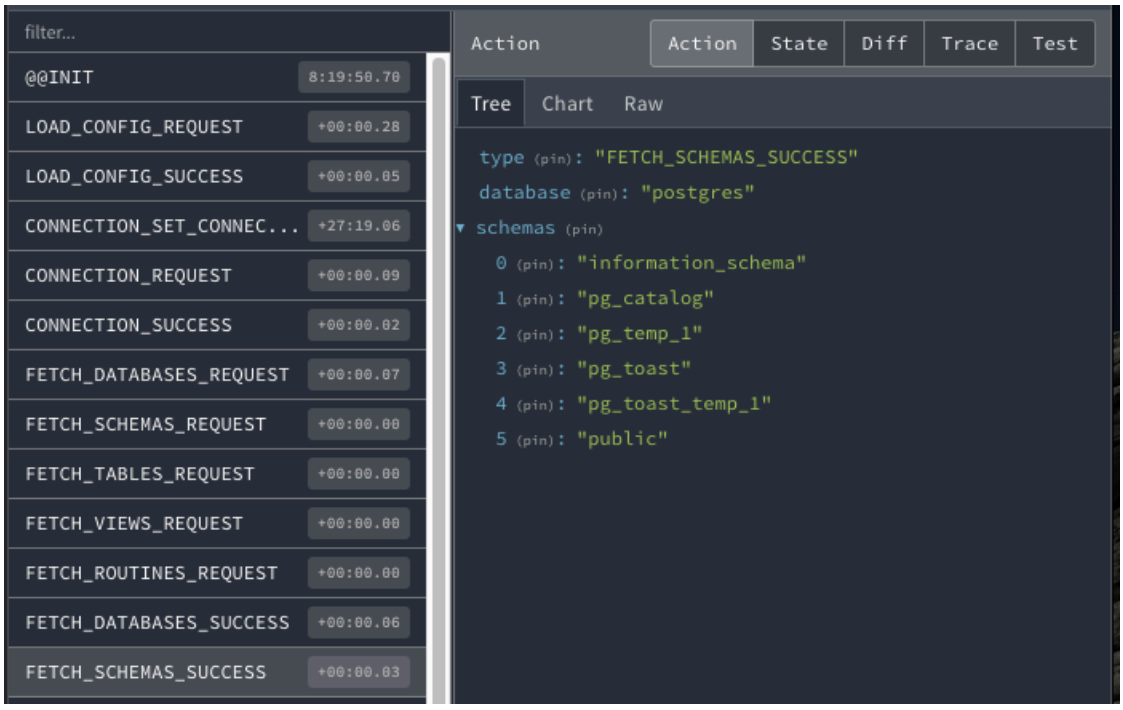


Рисунок 4.15 – Результат виконання запиту на отримання баз схем на сервері

Рисунок 4.16 Відображає представлення на сервері.

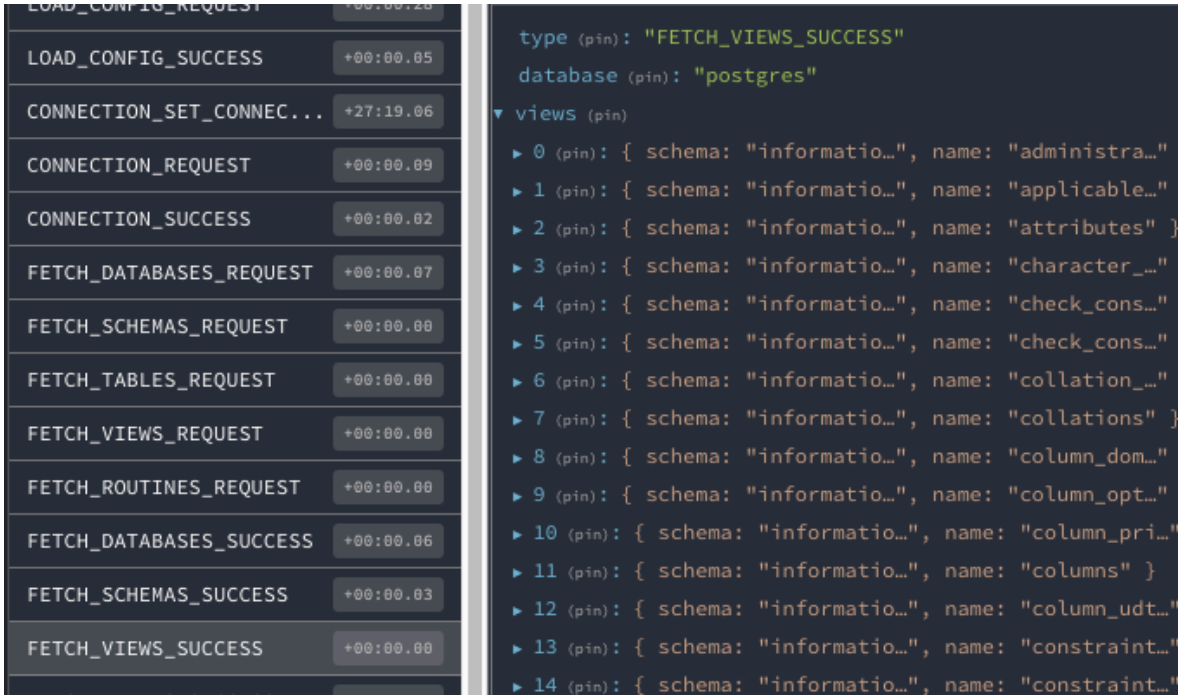


Рисунок 4.16 – представлення серверу SQL

Далі на Рисунку 4.17 відображено відповідь API на запит для отримання таблиць на сервері. Варто зазначити, що запит повертає всі таблиці включно з системними таблицями серверу, які там є за замовчуванням

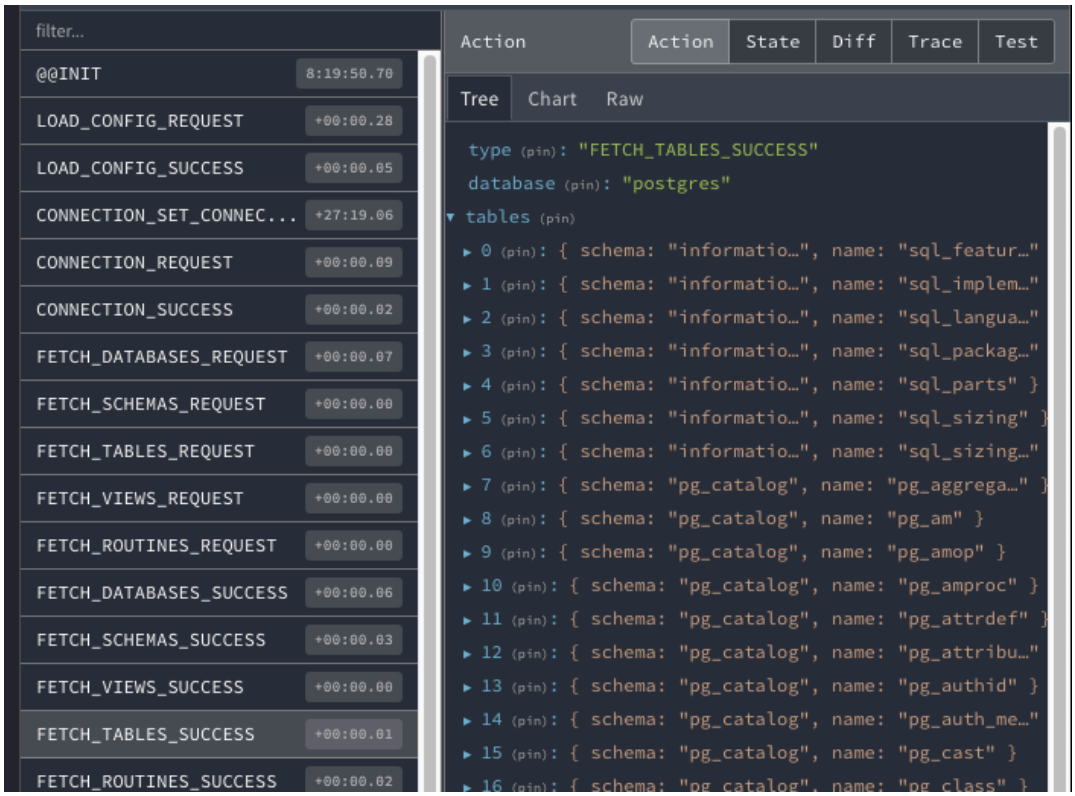


Рисунок 4.17 – Результат виконання запиту на отримання таблиць з серверу

Далі після виконання всіх вище описаних запитів, перед нами відкривається сторінка основного модулю, більш детально робота з модулем буде описана у наступному розділі.

4.2 Робота з скриптами усередині додатку

Після успішного підключення та виконання всіх запитів на отримання клону даних з серверу перед користувачем відкривається інтерфейс з інструментами для роботи з сервером (див. Рисунок 4.18).

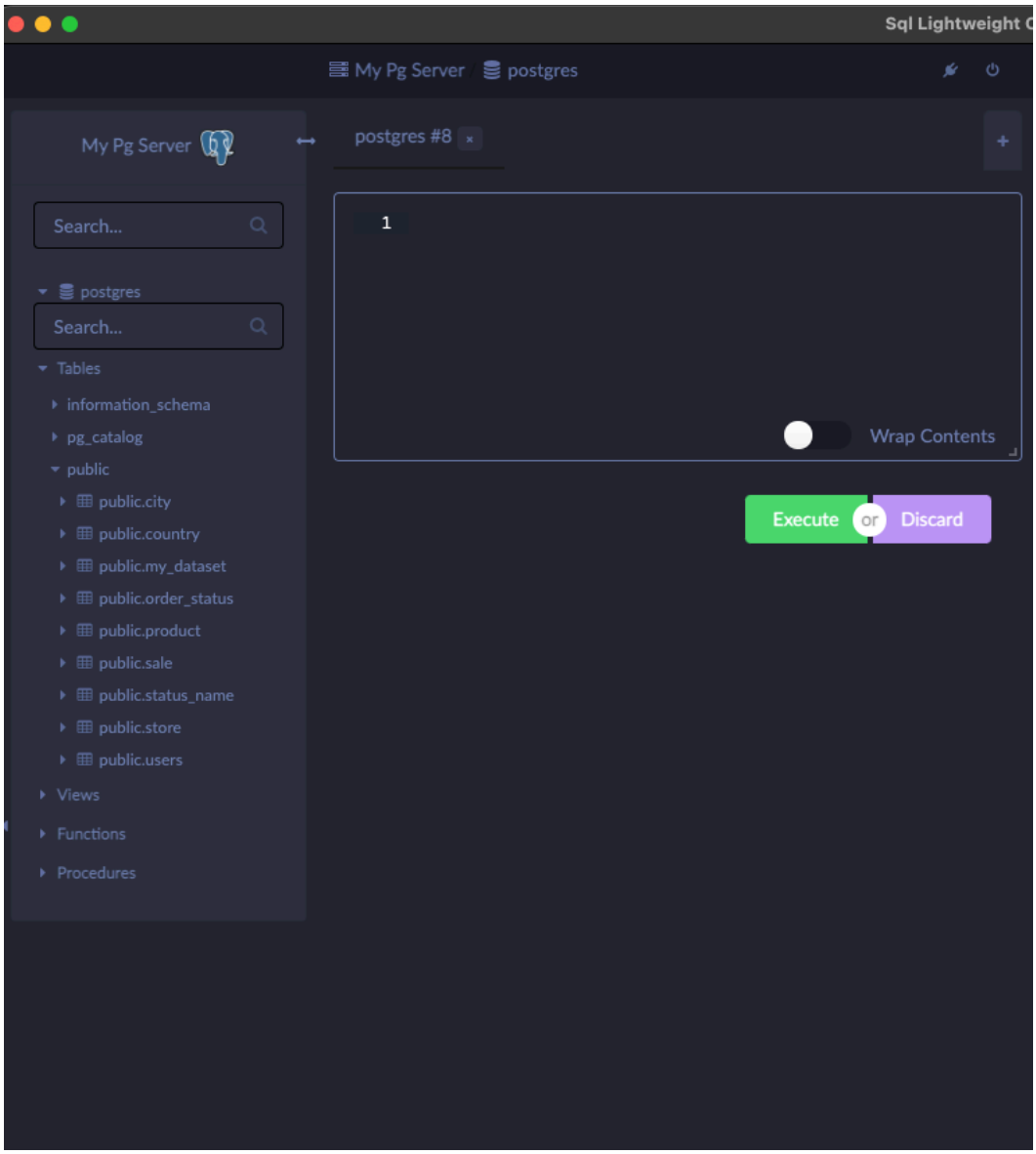
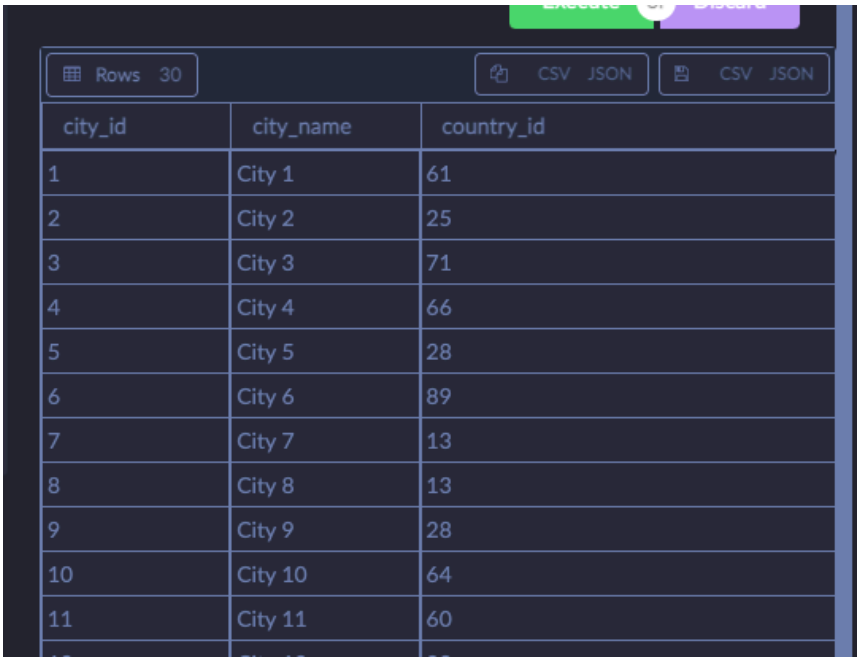


Рисунок 4.18 – Результат виконання запиту на отримання таблиць з серверу

У графічному вводу спробуємо написати скрипт для отримання записів з таблиці city (див. Рисунок 4.19).

Рисунок 4.19 – Отримання даних записів з таблиці city

Після натискання на кнопку Execute під модулем для вводу ми можемо побачити результат виконуваного скрипта (див. Рисунок 4.20).



city_id	city_name	country_id
1	City 1	61
2	City 2	25
3	City 3	71
4	City 4	66
5	City 5	28
6	City 6	89
7	City 7	13
8	City 8	13
9	City 9	28
10	City 10	64
11	City 11	60

Рисунок 4.20 – Результат виконання скрипта

Electron дозволяє описувати користувацькі таби для меню, де для скриптів передбачено додані наступні функції: для збереження, збереження на локальній системі та відкриття скриптів з файлу (див. Рисунок 4.21).

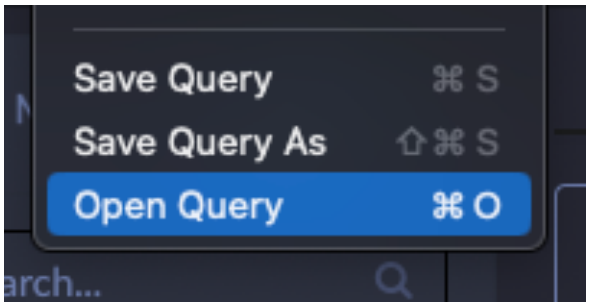


Рисунок 4.21 – додаткова панель інструментів для роботи з скриптами

На Рисунках 4.22-4.23 показаний процес збереження скрипта.

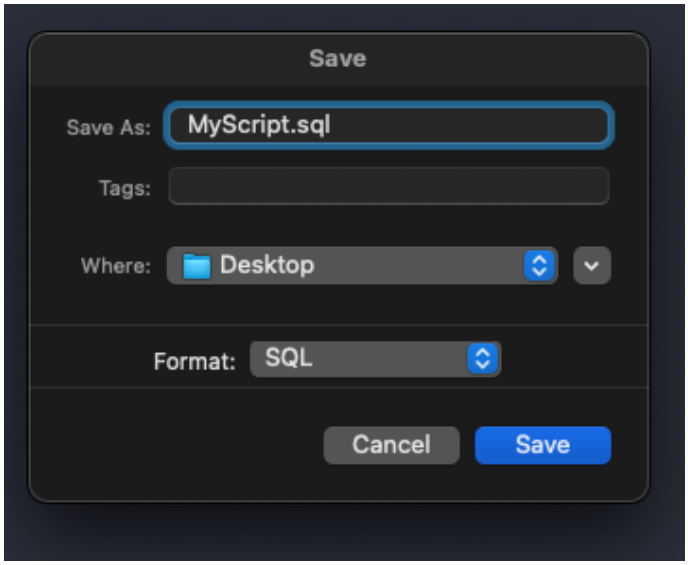


Рисунок 4.22 – Збереження файлу скрипт

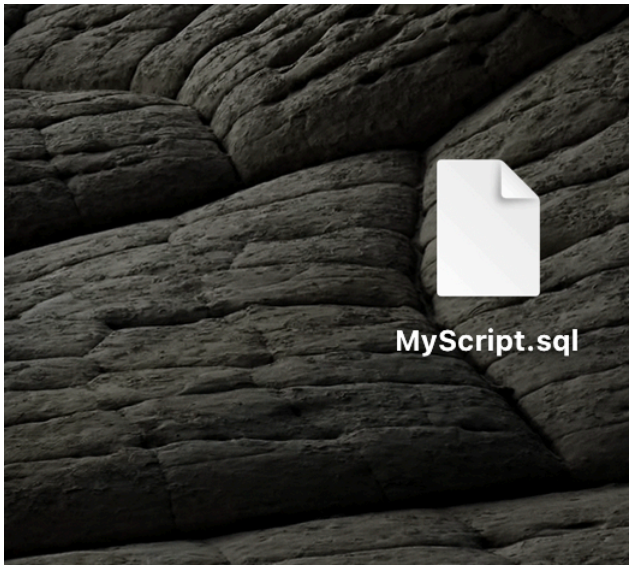


Рисунок 4.23 – Збережений файл скрипту

На Рисунках 4.24-4.25 відображено обернений процес відкриття скрипту з локальної машини.

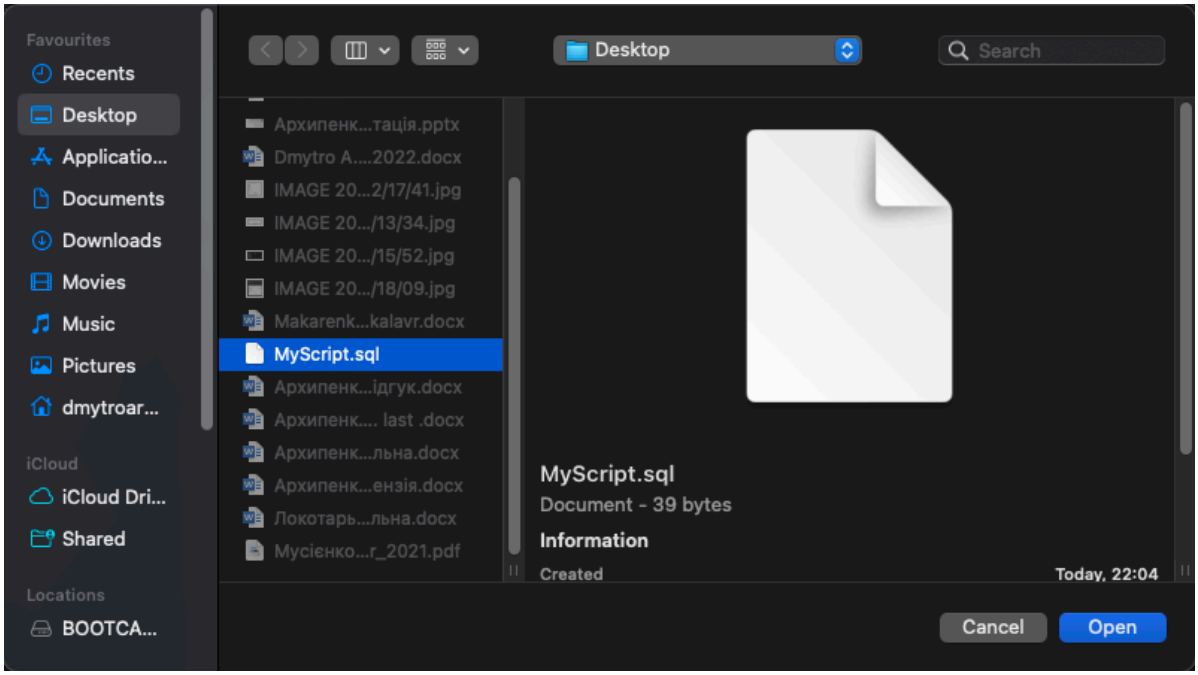


Рисунок 4.24 – Відкриття збереженого файлу скрипту

Рисунок 4.25 – Результат відкритого файлу скрипта

4.3 Копіювання даних та експорт

На панелі інструментів відображення результату виконання скрипта знаходяться дві кнопки для копіювання та експорту результату в двох форматах: CSV та JSON. На Рисунках 4.26-4.28 відображено процес копіювання, а далі на Рисунках 4.29-4.31 зображено процес експорту у відповідних форматах.

Рисунок 4.26 – Копіювання

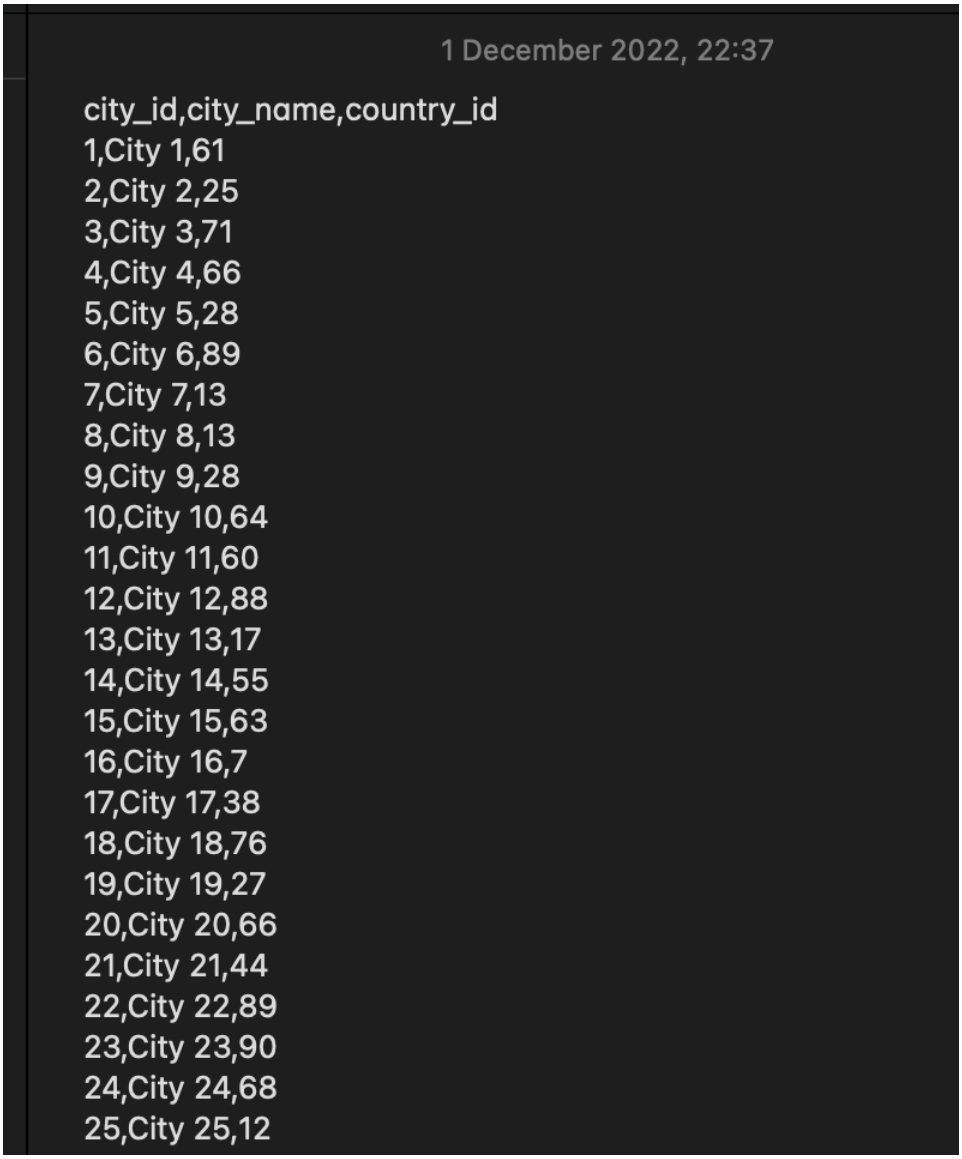


Рисунок 4.27 – Копіювання CSV

Рисунок 4.28 – Копіювання JSON

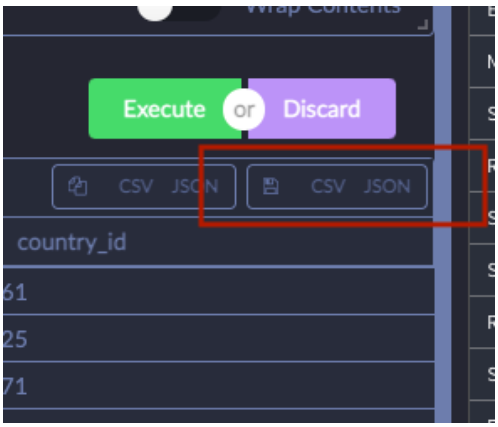
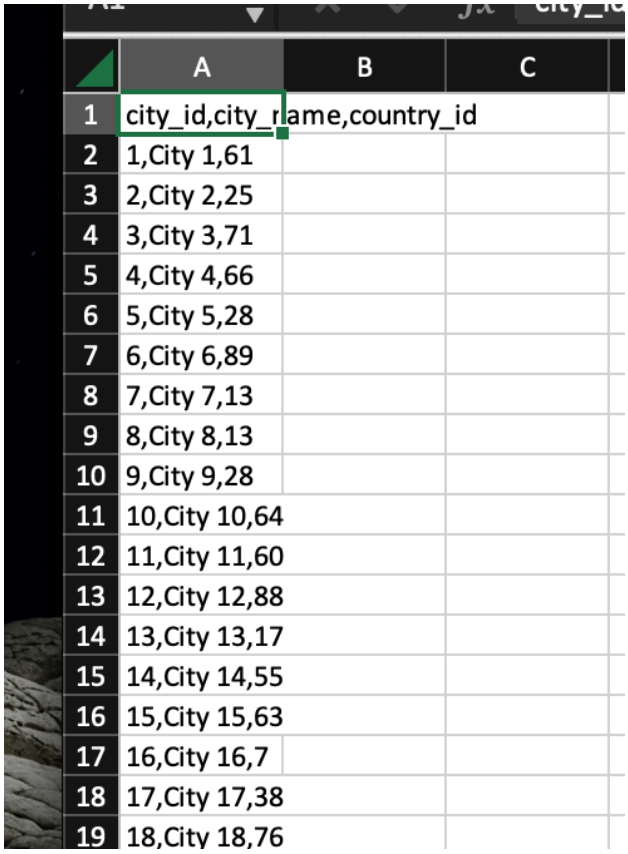


Рисунок 4.29 – Експорт даних



	A	B	C
1	city_id,city_name,country_id		
2	1,City 1,61		
3	2,City 2,25		
4	3,City 3,71		
5	4,City 4,66		
6	5,City 5,28		
7	6,City 6,89		
8	7,City 7,13		
9	8,City 8,13		
10	9,City 9,28		
11	10,City 10,64		
12	11,City 11,60		
13	12,City 12,88		
14	13,City 13,17		
15	14,City 14,55		
16	15,City 15,63		
17	16,City 16,7		
18	17,City 17,38		
19	18,City 18,76		

Рисунок 4.30 – Експорт даних CSV

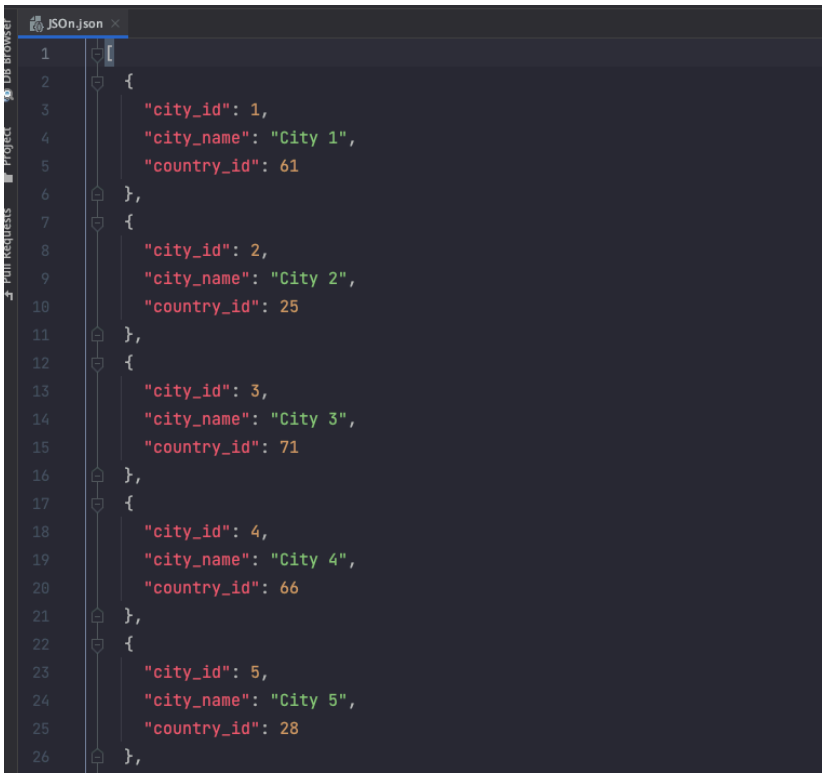


Рисунок 4.31 – Експорт даних JSON

4.4 Користувацькі налаштування додатку

Для покращення користувацького досвіду на головному екрані додатку додано модальне вікно з налаштуваннями, де користувач може змінити наступні параметри:

- Масштаб компонентів;
- Лімітування кількості записів;
- Відображення сітки списком підключень;
- Зміна кольорової схеми додатку на світлу/темну.

Модальне вікно з налаштуваннями зображене на Рисунку 4.32.

Рисунок 4.32 – Вікно налаштувань додатку

Нижче на Рисунку 4.33 зображена демонстрація світлої теми додатку.

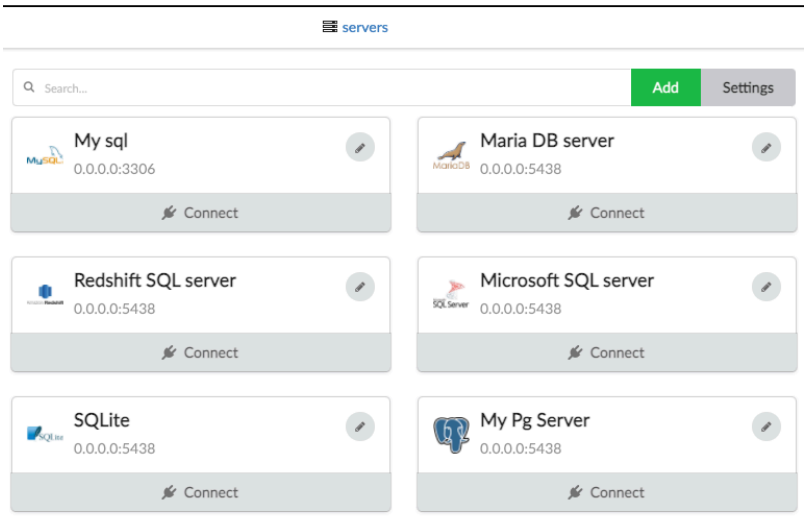


Рисунок 4.33 – Світла тема додатку

На Рисунку 4.34 продемонстровано зміну відображення списку створених підключень до серверів.

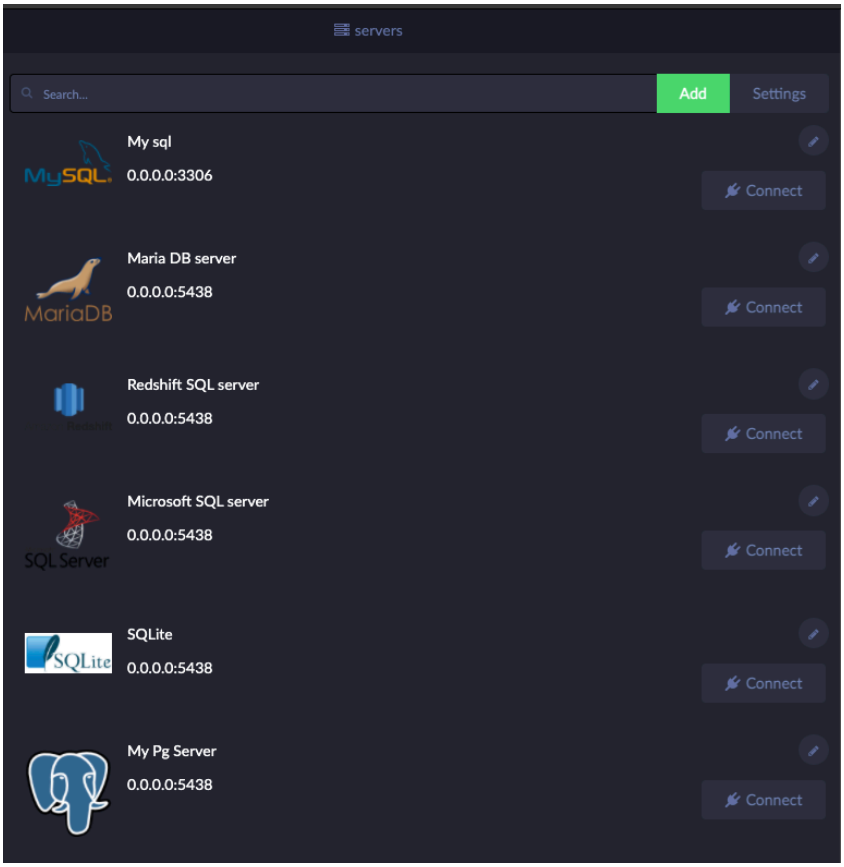


Рисунок 4.34 – Режим відображення підключень списком

На Рисунку 4.35 зображено збільшений масштаб елементів для зручного користування на екранах різного розширення.

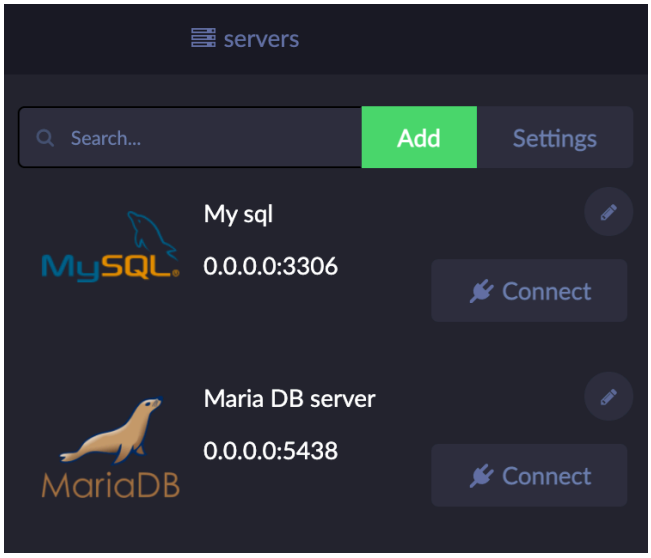


Рисунок 4.35 – Збільшений масштаб компонентів додатку

5 РОЗРОБЛЕННЯ СТАРТАП ПРОЕКТУ

Термін для позначення бізнесу та проектів, коли вони тільки починають розвиватися - це стартап. Важко сказати точно, скільки часу повинен функціонувати стартап, щоб його можна було вважати таким; це може варіюватися від декількох тижнів до місяців. Залежно від його життєздатності, іноді стартап отримує фінансування, що сприяє його подальшому успіху, однак, якщо перспективи виглядають несприятливими, він може бути взагалі закритий.

5.1 Опис ідеї стартапу

Даний проект полягає у створенні системи перегляду компонентів бази даних за допомогою параметрів ODBC. Вона спеціалізується на управлінні та оптимізації управління ресурсами.

За допомогою створеної моделі користувачі можуть підтримувати з'єднання з серверами, отримувати дані, що містяться в базах даних, а також легко ідентифікувати та здійснювати дії над підключеними об'єктами. Ця програма зручна у використанні і не потребує тривалого навчання для початку роботи з нею.

Важливо зазначити, що моделі в системі можуть швидко розширюватися, і вони можуть бути налаштовані за необхідності, тобто відповідно до потреб користувача.

У Таблиці 5.1 нижче представлені основні концепції, які були використані при розробці програмного продукту, потенційні сфери застосування та переваги для користувачів.

Таблиця 5.1

Опис ідеї закладеної у розробку стартапу

Ідея стартапу	Цільова аудиторія	Вигода для користувачів

Моніторинг стану серверів баз даних	Команди розробників різних за масштабом проектів	Уніфікований засіб для роботи з великою кількістю баз даних
Перегляд баз даних та взаємодія з їх наповненням		Можливість швидкого доступу до віддаленого серверу без спеціального програмного забезпечення
		Керування та моніторинг великої кількості даних

Таблиця 5.2 демонструє порівняльну характеристику з конкурентами, шляхом визначення сильних та слабких сторін.

Таблиця 5.2

Порівняльна характеристика

№ п/п	Техніко-економічні характеристики ідеї	(Потенційні) конкурентів Товари/концепції			W (слабка сторона)	N (нейтральна сторона)	S (сильна сторона)
		Даний проект	Конкурент 1	Конкурент 2			
1	Економія	Будь-яка команда продукту, на якому використовується SQL сервер у якості БД	Розробники продуктів, що застосовують їх дистрибутиву у якості серверу для збереження даних	Компанії різних розмірів у сфері IT	Вкладення на старті є дуже значними у порівнянні з доходом за рахунок	Вимагає аналізу на певному відрізку часу роботи	Можливість швидко масштабувати рішення

					конкуренції		
2	Гнучкість	Будь-яка команда продукту, на якому використовується SQL сервер у якості БД	Розробники продуктів, що застосовують їх дистрибутиву у якості серверу для збереження даних	Компанії різних розмірів у сфері IT	Складності у разі оновлення певного дистрибутиву SQL серверу	Вимагає аналізу на певному відрізку часу роботи	Покриття всіх можливих рішень використання SQL серверу
3	Капіталовкладення	Будь-яка команда продукту, на якому використовується SQL сервер у якості БД	Розробники продуктів, що застосовують їх дистрибутиву у якості серверу для збереження даних	Компанії різних розмірів у сфері IT	Капіталовкладення залежать від вимог цільової аудиторії	Вимагає аналізу на певному відрізку часу роботи	За рахунок команди швидке масштабування та адаптація рішень для інших дистриубуторів
4	Динаміка розвитку	Будь-яка команда продукту, на якому використовується SQL сервер у якості БД	Розробники продуктів, що застосовують їх дистрибутиву у якості серверу для збереження даних	Компанії різних розмірів у сфері IT	Моніторинг оновлення використовуваних дистрибутивів серверів ринку	Вимагає аналізу на певному відрізку часу роботи	Високий потенціал за рахунок уніфікованого рішення

Заздалегідь визначений відбір переваг, недоліків та об'єктивних ознак і складових концепції потенційного товару слугує основою його конкурентоспроможності.

Основною перевагою продукту – є те, що він присвячений спеціалізованій галузі – моніторинг та робота з серверами БД, і може бути використаний компаніями, які потребують великої кількості баз даних та уніфікованого рішення для них. У порівнянні з базовим програмним забезпеченням, які є більш прив’язаними до конкурентного дистрибутиву та складними у використанні, продукт є менш дорогим та більш простим в експлуатації.

Хоча, за потреби, майбутнім покупцям доведеться доопрацьовувати та модифікувати його. Однак вартість та час, що виділяються на внесення змін, є меншими, аніж у конкурентів.

5.2 Канва бізнес-моделі стартапу

Бізнес-модель – це метод роботи, за допомогою якого організація отримує дохід. Вона описує як заробляти гроші. Щоб лаконічно і просто пояснити і відобразити діяльність компанії на одній сторінці, використовується канва або схема. У цій схемі використано 9 блоків.

Блок 1 – "Ключові партнери" – описує можливих партнерів, які співпрацюють та допомагають у веденні бізнесу.

Другий блок описує ключові види діяльності, які необхідні для реалізації моделі компанії.

Блок 3 – "Ключові ресурси" – описує найбільш важливі активи, необхідні для забезпечення функціонування бізнес-моделі.

Блок 4 – "Ціннісні пропозиції" – описує продукти та послуги, які матимуть відчутні переваги для визначених кластерів споживачів.

Блок 5 фокусується на взаємовідносинах з клієнтами. Зазначаються типи зв'язків, які компанія має з різними демографічними групами споживачів.

Блок 6 присвячений каналам взаємодії – ілюструє, як компанія взаємодіє зі споживчими групами та поширює свої пропозиції серед них.

Блок 7 – "Споживчі сегменти" – розкриває демографічні характеристики споживачів, яких компанія прагне залучити та обслуговувати.

Блок 8 – "Структура витрат" - визначає, на які сфери бізнес витрачатиме найбільше коштів.

Блок 9 – "Потоки доходів" – визначає можливі доходи, припускаючи, звідки будуть надходити прибутки.

Таблиця 5.3 була створена з використанням дев'яти структурних блоків, згаданих в рамках оцінки проекту.

Таблиця 5.3

Канва стартапу як бізнес-проекта

<table>
<tr>
<td rowspan="2">(1) КЛЮЧОВІ ПАРТНЕРИ
1) Команди розробників
2) Контент менеджери
3) Системні адміністратори</td>
<td>(2) КЛЮЧОВІ ВИДИ ДІЯЛЬНОСТІ
1) Розробка уніфікованого клієнта СУБД для різних дистрибуторів SQL серверів</td>
<td rowspan="2">(4) ЦІННІСНІ ПРОПОЗИЦІЇ
1) Динамічний розвиток продукту
2) Дотримання дедлайнів та регулярне демо продукту при оновленні
3) Відкритість для вдосконалення за рахунок комунікації з цільовою аудиторією</td>
<td>(5) ВЗАЄМОВІДНОСИНИ ЗІ СПОЖИВАЧАМИ
1) Співпраця з цільовою аудиторією для покращення досвіду використання
2) Окремі менеджери, які відповідальні за великих клієнтів</td>
<td rowspan="2">(7) СПОЖИВЧІ СЕГМЕНТИ
Будь-які за масштабом команди розробників програмного забезпечення</td>
</tr>
<tr>
<td>(3) КЛЮЧОВІ РЕСУРСИ
1) Розумові
2) Трудові, а саме: менеджмент, розробники, маркетологи
3) Техніка для розробників</td>
<td>(6) КАНАЛИ КОМУНІКАЦІЇ
1) Соціальні мережі (Slack, Telegram)
2) Електронна пошта
3) Телекомунікація
4) Мітинги</td>
</tr>
<tr>
<td colspan="3">(8) СТРУКТУРА ВИТРАТ
1)Покупка техніки для працівників
2) Оренда офісу
3) Маркетинг
4) Заробітна плата</td>
<td colspan="3">(9) ПОТОКИ НАДХОДЖЕННЯ ДОХОДІВ
1) Партнерські організації
2) Потік клієнтів від маркетингу
3) Підписка на продукт</td>
</tr>
</table>

5.3 Бізнес-процеси проекту

Бізнес-процеси проекту описано в Таблиці 5.4.

Таблиця 5.4

Бізнес-процеси стартапу

Група процесів	*Бізнес-процес*	*Ступінь опрацювання бізнес-процесу*	
		є реалізовани м	*буде реалізовани м*
1	2	3	4
Розробка додатку	Проектування системи	+	
	Дизайн додатку	+	
	Дослідження ринку та визначення конкурентоспроможності продукту	+	
	Налагодження роботи додатку з популярними серверами	+	
Внутрішні процеси	Оренда офісу		+
	Маркетинг		+
	Пошук партнерів		+
	Вибір внутрішнього ПЗ для комунікації		+
	Створення workspace в Google		+
Робота з працівниками	Найм працівників		+
Обслуговування цільової аудиторії	Обслуговування та вдосконалення продукту		+
	Юридичні договори для співпраці		+

5.4 Характер формування споживчої цінності проекту

Характер формування споживчої цінності проекту направлений на:

- покращення рішень для існуючих потреб;

- зменшення ціни для задоволення існуючих потреб;
- реалізація нових можливостей для нових потреб.

5.5 Аналіз ринкових можливостей запуску продукту на ринку

Таблиця 5.5. відображає попит даної технології на ринку інформаційних технологій.

Таблиця 5.5

Попередня характеристика потенційного ринку стартап-проекту

№ п/п	Показники стану ринку (найменування)	Характеристика
1	Кількість головних гравців, од	~4-8
2	Загальних обсяг продаж, грн/ум.од	~5-10
3	Динаміка ринку (якісна оцінка)	Динамічний ріст
4	Наявність обмежень для входу	Необхідність вдосконалення для підвищення потенціалу конкурентоспроможності
5	Приблизна норма рентабельності в галузі, %	~10-20%

Таблиця 5.6 демонструє характеристику потенційних клієнтів.

Таблиця 5.6

Характеристика потенційних клієнтів стартап-проекту

№ п/п	Потреба, що формує ринок	Цільова аудиторія	Відмінності у поведінці різних потенціальних груп клієнтів	Вимоги споживачів до товару

(Продовження Таблиці 5.6)

1	Планування та керування великою кількість обчислювальних ресурсів в блокчейн-мережах	Компанії, що займаються майнингом або блокчейном	Ціна на продукт, якість програмного продукту, технічні регламенти	Введення проекту в експлуатацію завчасно та без додаткових грошових витратах

Таблиця 5.7 демонструє фактори можливих загроз.

Таблиця 5.7

Фактори загроз

№ п/п	Фактор	Зміст загрози	Можлива реакція
1	Велика кількість запитів на вдосконалення	Великий попит на вдосконалення може викликати потребу в розширенні команди	Збільшення кількості розробників
2	Необхідність підвищення рівня безпеки для роботи з великими клієнтами	Багато клієнтів з-за кордону потребують виконання сертифікацій безпеки	Планування виконання вимог великих клієнтів для продовження співпраці
3	Слабкий попит	Через низький рівень задоволення клієнтів слід постійно підтримувати з ними контакт для покращення продукту для їх задоволення	Інтенсивний аналіз ринку та комунікація з користувачами

Таблиця 5.8 показує конкурентні фактори на ринку. Дуже важливим при розробці стартапу є аналіз конкурентів, адже за рахунок цього аналізу ми можемо визначити їх проблемні місця та за рахунок ефективного рішення мати перевагу над ними при виборі продукту користувачем.

Таблиця 5.8

Ступеневий аналіз конкуренції на ринку

Особливості конкурентного середовища	В чому проявляється	Вплив на діяльність
Тип конкуренції – чиста	Інші компанії також можуть пропонувати свої рішення	Участь в конференціях, патентування
Рівень конкурентної боротьби: національний	Поширювати можна по всьому світу шляхом адаптації мовного пакету	Поширення програмного забезпечення, як на ринок України, так і на закордон
Галузева ознака: міжгалузева	Система такого типу може використовуватись не тільки розробниками, а й також іншими командами проектів	Адаптація користувацького інтерфейсу під потреби інших відділів компанії, що потребують виконання завдань з джерелами даних
За характером конкурентних переваг: нецінова	Під час вибору сервісу клієнти звертають увагу на функціональні можливості продукту, тому варто звернути увагу на розширення продукту новими унікальними можливостями у користувацькому досвіді	Подальше розширення можливостей продуту

5.6 Технологічний аудит ідеї проекту

Підрахунок технологічної можливості реалізації ідеї стартап – проекту передбачає аналіз таких складових:

- які технології будуть використовуватись у ході розробки проекту?
- чи є існуючі технології?

– чи доступні такі технології команді розробників?

У таблиці 5.9 наведено технологічний аналіз можливості реалізації кінцевого продукту.

Таблиця 5.9

Визначення технологічної здійсненності ідеї створення системи моделювання режимів роботи інтегрованих систем енергозабезпечення

№	Ідея проекту	Технології її реалізації	Наявність технологій	Доступність технологій
1	2	3	4	5
1.	Створення шаблону кросплатформенного додатку	Electron	Є у наявності	Безкоштовна, доступна
2.	Побудова інтерфейсу	React	Є у наявності	Безкоштовна, доступна
3.	Збереження отриманої інформації	Операційна система (кеш)	Є у наявності	Безкоштовна, доступна
Обрана технологія реалізації ідеї проекту: Всі вищезазначені				

5.7 SWOT-аналіз проекту

Фінальним етапом ринкового аналізу можливостей впровадження проекту – є складання SWOT-аналізу (матриці аналізу сильних та слабких сторін, загроз та можливостей) (Таблиця 5.10).

Таблиця 5.10

SWOT-аналіз проекту

1	2
Сильні сторони (S): – Створення власного бренду. – Розробка проекту без компанії посередника. – Рентабельність.	**Слабкі сторони (W):** – Індивідуальні запити цільової аудиторії. – Непротестований продукт на реальних користувачах. – Відсутність попереднього фідбеку цільової аудиторії. – Відсутність зв’язків у сфері.
Можливості (O): – Розширення у разі росту клієнтів. – Співробітництво з іншими компаніями. – Можливість популяризації через маркетинг. Відсутність подібного ПО на ринку може викликати стрімкий ріст.	**Загрози (T):** – Підвищення цін на техніку. – Економічний стан. – Вихід конкурентів на ринок. Поганий маркетинг.

5.8 Розроблення ринкової стратегії проекту

В Таблиці 5.11 показано базову стратегію розвитку проекту, 5.12 – базову стратегію конкурентної поведінки.

Таблиця 5.11

Визначення базової стратегії розвитку

№ п/п	Базова стратегія розвитку	Стратегія охоплення ринку	Ключові конкурентоспроможні позиції відповідно до обраної альтернативи

(Продовження Таблиці 5.11)

1	Уніфікація рішення	Концентрація на потребах цільової аудиторії	Покриття більшості випадків взаємодії з джерелами даних

Таблиця 5.12

Визначення базової стратегії конкурентної поведінки

№ п/п	Чи є проект «пер-шопрохідцем» на ринку?	Чи буде компанія шукати нових споживачів, або забирати існуючих у конкурентів?	Чи буде компанія копіювати основні характеристики товару конкурента і які?	Стратегія конкурентної поведінки
1	Так	Так	Ні	Стратегія заняття конкурентної ніші

5.9 Розроблення маркетингової програми стартап-проекту

У таблиці 5.13 наведено визначення ключових переваг концепції потенційного товару.

Таблиця 5.13

Визначення ключових переваг концепції потенційного товару

№ п/п	Потреба	Вигода, яку пропонує товар	Ключові переваги перед конкурентами

(Продовження Таблиці 5.13)

1	Потреба у масштабуванні	Пришвидшення виконання рутинних операції над різними серверами баз даних в рамках одного додатку	Вузька область конкурентів у реалізації уніфікованого рішення
2	Потреба в організації процесів та менеджменту проекту	Зменшення ресурсних затрат	Доступність та клієнтоорієнтованість

У таблиці 5.14 наведено концепцію маркетингових комунікацій.

Таблиця 5.14

Концепція маркетингових комунікацій

№ п/п	Специфікація поведінки цільових каналів	Канали комунікацій, якими користуються цільові клієнти	Ключові позиції, обрані для позиціонування	Завдання рекламного повідомлення
1	Міжнародні компанії, що займаються розробкою програмного забезпечення	Конференції, виставки, соціальні мережі	Експертність, клієнтоорієнтованість	Показати і зробити акцент на уніфікації роботи з джерелами даних

(Продовження Таблиці 5.14)

2	Компанії, що в рамках України працюють над розробкою ПЗ	Публікації на форумах, маркетинг	На ранніх етапах низька ціна на період становлення технології	Показати, що інтерфейс є інтуїтивно зрозумілим для користувача та сам додаток є легким у застосуванні

5.10 Розрахунок виробничих та загальних витрат

У Таблиці 5.15 наведені витрати, які необхідно враховувати для завершення поточної діяльності за проектом (виробництво товарів, надання послуг тощо).

Таблиця 5.15

Розрахунок виробничих витрат

№ з/п	*Стаття витрат*	*Сукупні витрати за період, тис. грн.*				
		2021	*2022*	*2023*	*2024*	*2025*
1.	Витрати на збут, просування та рекламу	5000	5000	5000	5000	5000
2.	Витрати на матеріальні ресурси	10000	10000	10000	10000	10000
3.	Витрати на фонд заробітної плати	5000	5000	5000	5000	5000
Разом:		20000	20000	20000	20000	20000

У Таблиці 5.16 відображено загальні витрати.

Таблиця 5.16

Розрахунок загальних витрат

Показник	*Значення по роках*					*Разом*
	2022	*2023*	*2024*	*2025*	*2026*	
Інвестиційні витрати	15000	0	0	0	0	15000

(Продовження Таблиці 5.16)

Виробничі витрати	12000	12000	12000	12000	12000	60000
Обсяг загальних витрат, в тому числі за рахунок	30000	15000	15000	15000	15000	90000
– власних коштів	10000	5000	5000	5000	5000	30000
– коштів інвестора	20000	0	0	0	0	20000

5.11 Грошовий потік та оцінка вартості проекту

Чиста теперішня вартість (NPV, Net Present Value) визначається шляхом віднімання загальної суми інвестицій, здійснених за період реалізації проекту, від загальної суми отриманих доходів. Інформація щодо цього розрахунку наведена в Таблиці 5.17.

Таблиця 5.17

Грошовий потік та оцінка вартості проекту

№	*Показник*	*Значення по роках*					*Разом*
		2022	*2023*	*2024*	*2025*	*2026*	
1.	Надходження від проекту (виручка від реалізації продукції, послуг) (D)	4000	4000	5000	6000	7000	28000
2.	Загальні витрати (I), в тому числі	10000	500	500	500	500	12000
3.	Грошовий потік (3 = 1 – 2) (CF)	-6000	2000	2400	4000	5000	17400
4.	Акумульований грошовий потік (ACF)	-8300	-4200	-1000	3200	4600	-

ВИСНОВКИ

У ході виконання даної роботи було проведено ретельний аналіз предметної області, дослідження існуючих систем управління базами даних, аналіз вимог, вибір технологій, спроектовано архітектуру системи, виконано моделювання прецедентів та внутрішніх процесів у програмному продукті. Крім того, було проведено моделювання зразка та власної системи для організації програмного коду. У результаті реалізації була створена автоматизована система підтримки та управління SQL-серверами з використанням параметрів ODBC для наочних цілей.

У роботі була проведена оцінка методів вирішення технологічних питань, які існують на сьогоднішній день. При цьому було враховано багато сучасних аналогів. Розглянуто інструменти, що використовувалися при створенні програми, та їх відповідні повноваження.

Описані засоби реалізації програмного комплексу, що включають в себе розробку діаграм прецедентів, розробку макету програмного продукту, побудову концептуальної бази даних та деталізацію користувацького інтерфейсу.

Удосконалено досвід користування СУБД за рахунок розробленого крос-платформенного додатку для візуалізації серверу SQL через дані ODBC та роботи з сервером.

Демонстрація функціонування програми одночасно з графічним інтерфейсом користувача та серверної системи усередині операційної системи MacOS.

ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Ви не знаєте JS (серія книг) – [Електронний ресурс] - Режим доступу до ресурсу: https://github.com/azat-io/you-dont-know-js-ru
2. Технічна документація Electron [Електронний ресурс] / – Режим доступу до ресурсу: https://www.electronjs.org/
3. Технічна документація Typescript [Електронний ресурс] / – Режим доступу до ресурсу: https://www.typescriptlang.org/
4. Алексей Васильев - Программирование на Java для начинающих: - П.: Эксмо, 2017 – 704с.
5. CSS офіційна документація по веб розробці з використання HTML CSS JAVASCRIPT [Електронний ресурс] — Режим доступу: https://www.w3schools.com/css/
6. Форум науково-технічних статей по IT технологіях [Електронний ресурс] — Режим доступу: https://medium.com/
7. Технічна документація Oracle DB [Електронний ресурс] / – Режим доступу до ресурсу: https://docs.oracle.com/en/database/oracle/oracle-database/
8. Технічна документація MySQL [Електронний ресурс] / – Режим доступу до ресурсу: https://dev.mysql.com/doc/
9. Загальні відомості про вебдизайн [Електронний ресурс] — Режим доступу: https://uk.wikipedia.org/wiki/%D0%92%D0%B5%D0%B1%D0%B4%D0%B8%D0%B7%D0%B0%D0%B9%D0%BD
10. Описание основных приемов нормализации базы данных [Електронний ресурс] / – Режим доступу до ресурсу: https://support.microsoft.com/ru-ru/help/283878/description-of-the-database-normalization-basics
11. NPM сховище великої кількості бібліотек для Javascript-а [Електронний ресурс] — Режим доступу: https://www.npmjs.com/
12. Технічна документація SQlite [Електронний ресурс] / – Режим доступу до ресурсу: https://www.sqlite.org/docs.html
13. Паттерн Data Access Object [Електронний ресурс] /– Режим доступу до ресурсу:

http://javatutor.net/articles/j2ee-pattern-data-access-object

14. SQL или NoSQL — вот в чём вопрос [Електронний ресурс] / – Режим доступу до ресурсу: https://habr.com/ru/company/ruvds/blog/324936/
15. ТОП-11 систем для управления бизнес-процессами в 2021 году [Електронний ресурс] /– Режим доступу до ресурсу: https://www.kickidler.com/ru/info/top-10-sistem-dlya-upravleniya-biznes-proczessami-v-2020-godu.html
16. Ларман К. Применение UML 2.0 и шаблонов проектирования: Пер. с англ. — М.:Вильямс, 2009. — 736 с.
17. Вирт Н. Алгоритмы и структуры данных: Пер. с англ. — М.: Невский диалект,2008. — 352 с.
18. Ярмола Ю. А. Компьютерные шрифты. — СПб.: BHV — Санкт-Петербург, 1994. —208 с.
19. Кэй М. XSLT. Справочник программиста. — СПб.: Символ-Плюс, 2002. — 1016 с.
20. REST Resourse Naming Guide [Електронний ресурс]. /– Режим доступу до ресурсу: https://restfulapi.net/resource-naming/
21. Data modeling in UML [Електронний ресурс] / – Режим доступу до ресурсу: http://www.agiledata.org/essays/umlDataModelingProfile.html
22. UML 2 Use Case Diagramming Guidelines [Електронний ресурс] / – Режим доступу до ресурсу: http://agilemodeling.com/style/useCaseDiagram.htm

ДОДАТОК А

Компоненти візуальних середовищ для баз даних через дані ODBC

Акт впровадження

УКР.НТУУ“КПІ ім. Ігоря Сікорського” номер зал.книжки _22М

Аркушів 1

2022

Затверджую
Генеральний директор ПрАТ
“Київська кондитерська фабрика
ROSHEN”
_____ Архипенко П.В.

Акт

впровадження результатів дослідження

Архипенка Д. П. на тему «Компоненти візуальних середовищ для БД через дані ODBC»

Я, що нижче поставила свій підпис, Генеральний директор Архипенко П.В. ТОВ ПрАТ “Київська кондитерська фабрика ROSHEN”, склав дійсний акт про те, що була спроектована та розроблена програмна система «Компоненти візуальних середовищ для БД через дані ODBC».

Прийнята до впровадження система дозволяє суттєво скоротити час на роботу з базами даних і дозволяє цей час використати для інших цілей.

Дана система є зручною, та була необхідною на підприємстві, її архітектура, яка заключається в модульності, була схвалена мною, з метою подальшого додавання до неї нових модулів, що надасть більше функціоналу і ще більше спростить роботу з базами даних.

Генеральний директор ПрАТ “Київська
кондитерська фабрика ROSHEN” _____ Архипенко Павло Васильович

ДОДАТОК Б

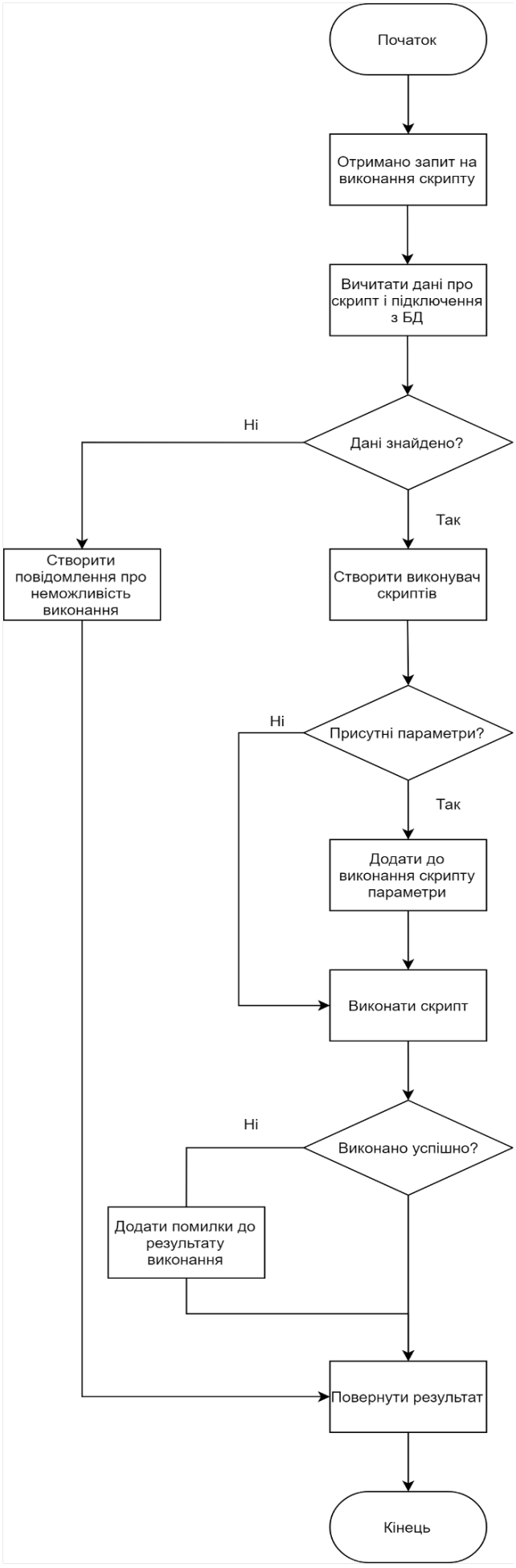
Компоненти візуальних середовищ для баз даних через дані ODBC

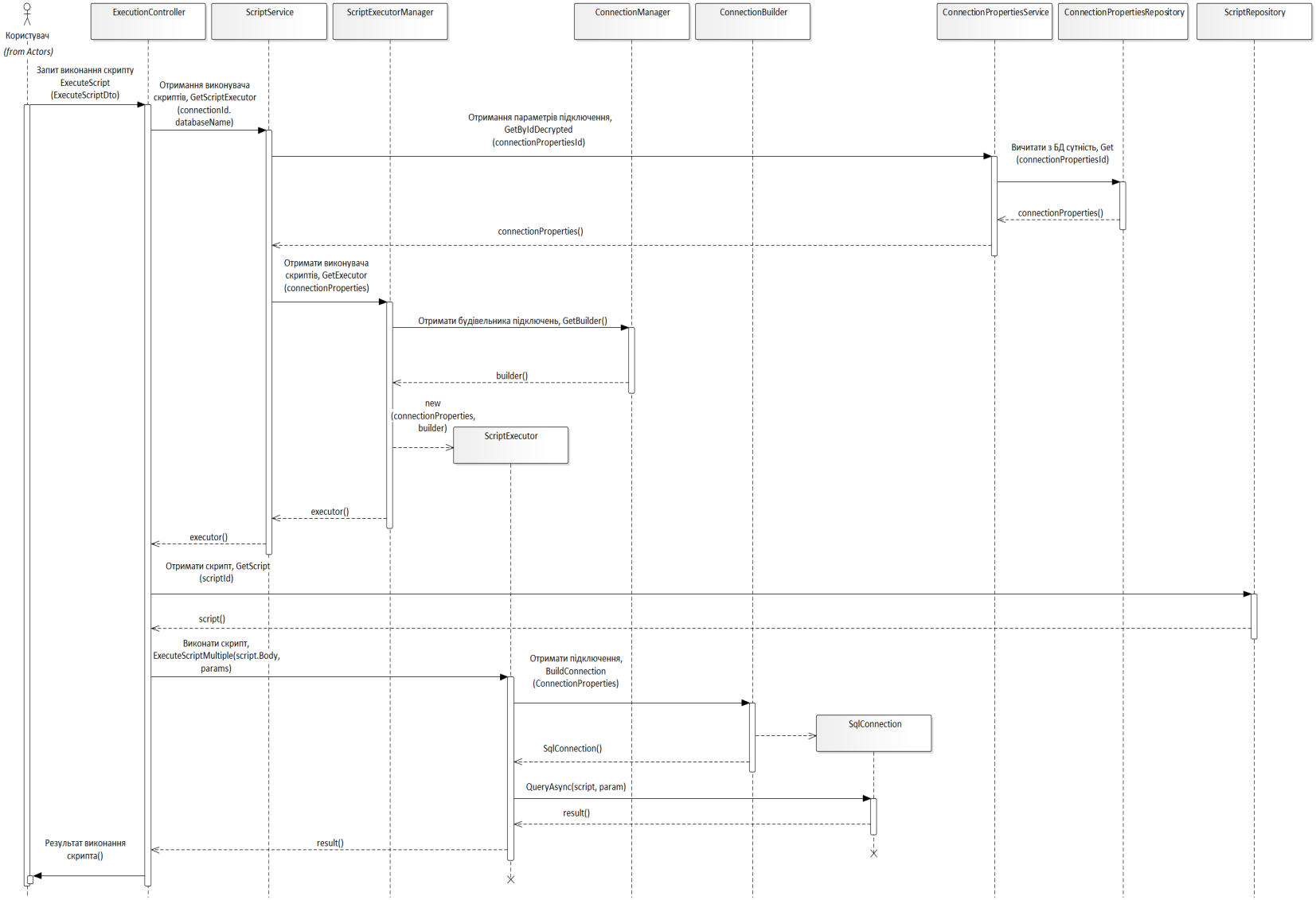
Блок-схема послідовності роботи скриптів всередині додатку

УКР.НТУУ“КПІ ім. Ігоря Сікорського” номер зал.книжки _22М

Аркушів 2

2022


Початок
Отримано запит на виконання скрипту
Вичитати дані про скрипт і підключення з БД
Дані знайдено?
Ні
Так
Створити повідомлення про неможливість виконання
Створити виконувач скриптів
Присутні параметри?
Ні
Так
Додати до виконання скрипту параметри
Виконати скрипт
Виконано успішно?
Ні
Додати помилки до результату виконання
Повернути результат
Кінець


Користувач
(from Actors)
ExecutionController
ScriptService
ScriptExecutorManager
ConnectionManager
ConnectionBuilder
ConnectionPropertiesService
ConnectionPropertiesRepository
ScriptRepository
Запит виконання скрипту ExecuteScript (ExecuteScriptDto)
Отримання виконувача скриптів, GetScriptExecutor (connectionId. databaseName)
Отримання параметрів підключення, GetByIdDecrypted (connectionPropertiesId)
Вичитати з БД сутність, Get (connectionPropertiesId)
connectionProperties()
connectionProperties()
Отримати виконувача скриптів, GetExecutor (connectionProperties)
Отримати будівельника підключень, GetBuilder()
builder()
new (connectionProperties, builder)
ScriptExecutor
executor()
executor()
Отримати скрипт, GetScript (scriptId)
script()
Виконати скрипт, ExecuteScriptMultiple(script.Body, params)
Отримати підключення, BuildConnection (ConnectionProperties)
SqlConnection
SqlConnection()
QueryAsync(script, param)
result()
result()
Результат виконання скрипта()