Київський національний університет імені Тараса Шевченка

Факультет інформаційних технологій

Кафедра програмних систем і технологій

Індивідуальне завдання № 1

Тема: «Створення SQL database Azure»

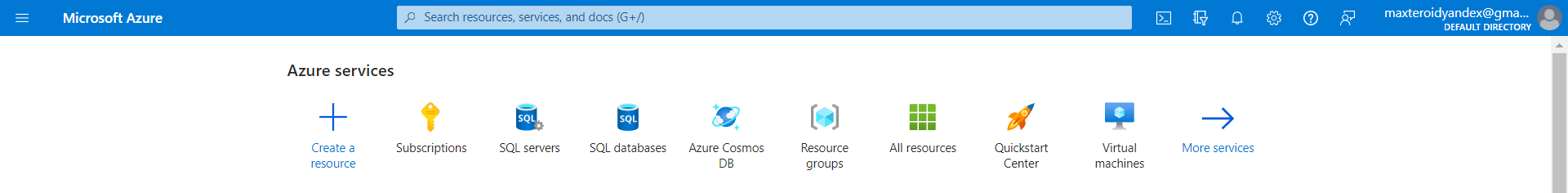
Дисципліна «Хмарні технології»

Підготував:

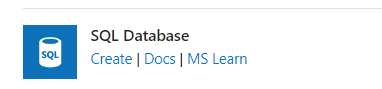
студент гр. ІПЗ-43(1)

Мішак Максим

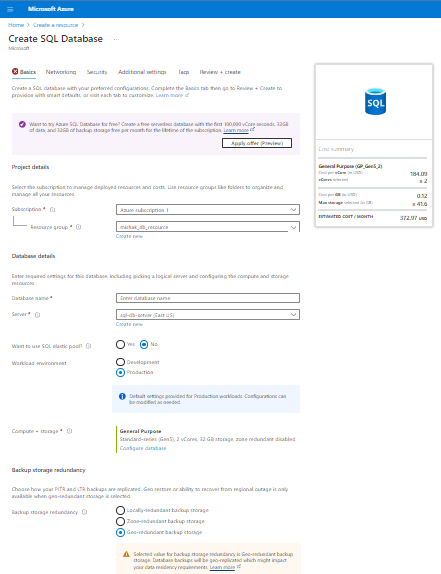
Для виконання роботи було обрано хмарний провайдер Azure, та базу данних SQL Azure. Для використання бази данних потрібно наслідувати декілька правил. Перш за все база данних повинна бути інкапсульована, що означає її незалежність та недоступність для зміни з будь якого місця. Також потрібно налаштувати правила авторизації, щоб тільки довірені джерела могли виконувати операції по її редагуванню. База данних та вся супутня інфраструктура буде розгортатись в хмарі. Для цього потрібно створити її екземпляр.



Щоб створити її екземпляр потрібно перейти в Create a resource, після чого обрати SQL databases

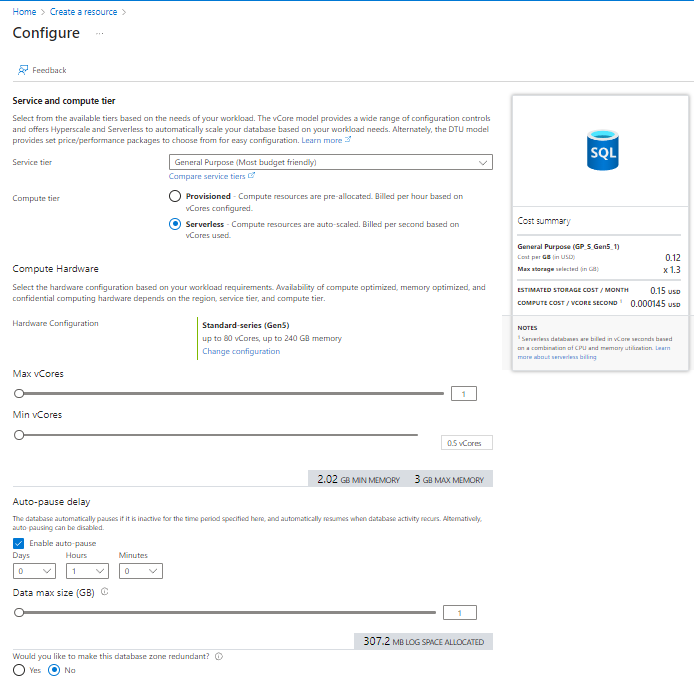


Після цього нам відкриється форма, яку потрібно заповнити. Форма включає в себе всі параметри об’єму, потужності, захищеності та правил підключення.

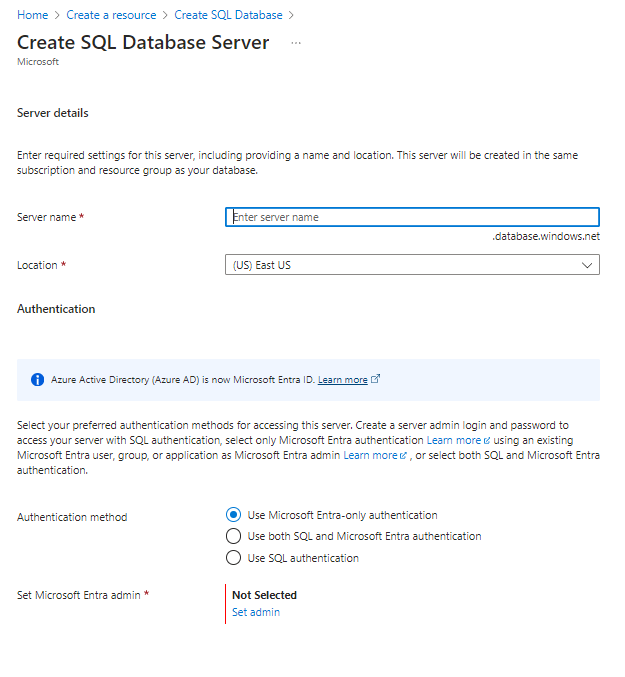


Важливими параметрами є:

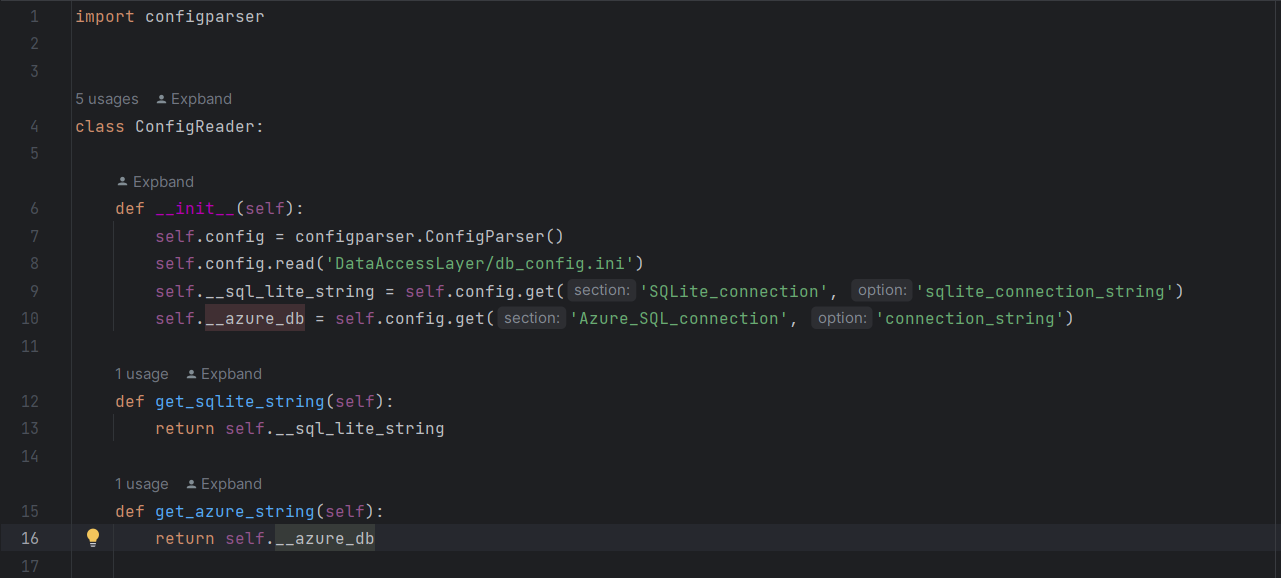
* Resource group
* Server
* Backup location

Також потрібно заповнити поля з назвою бази данних, налаштувати середовище роботи ,використання elastic pool. Для своєї бази данних я обрав такі параметри: назва бази данних sql\_db, середовище роботи: development, elastic pool - false. Параметри середовища:   


Тут я обрав без серверну архітектуру, віртуальне ядро: 0,5 - 1, автовимкнення: через годину, розмір бази данних: 1 гігабайт, решту налаштувань залишив стандартними. Після цього було налаштовано сервер.

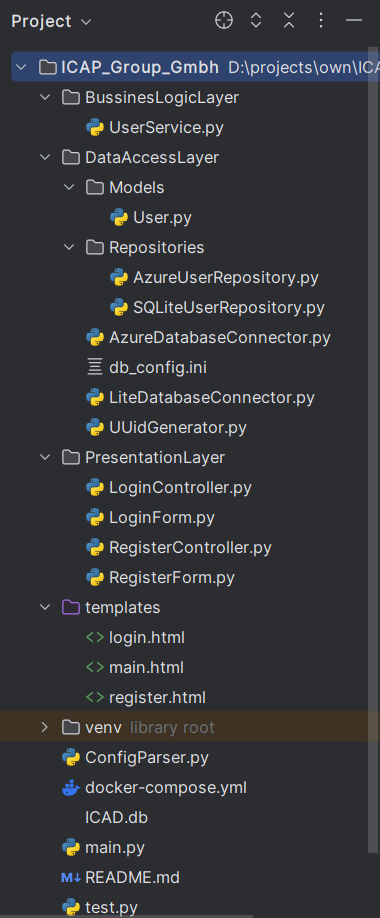


Назва сервера: sql-db-server, локація West EU, авторизація SQL and Entra authentication. Для ролі адміністратор обрав тільки власний акаунт. Після цього потрібно провести налаштування правил доступу. Для цього на вкладці Networking було обрано доступ з публічного ендпоінта, дозволено іншим сервісам інтегрувати з базою данних, та додано мій зовнішній IP адрес. Після цього було розгорнуто базу данних з усією супутньою архітектурою. Для її використання з додатку потрібно отримати connection string. В залежності від обраного методу авторизації нюанси використання connection string можуть бути різними. В моєму випадку я використовував авторизацію по SQL. Для отримання рядка переходимо на вкладку Connections. Я буду використовувати odbc драйвер для, python, тому копіюю відповідний рядок. На порталі надано стрічку без пароля. Для доступу він потрібен. Щоб не зберігати його у відкритому вигляді, я буду використовувати файл конфігурації, та клас, який буде зчитувати значення з конфігурації.

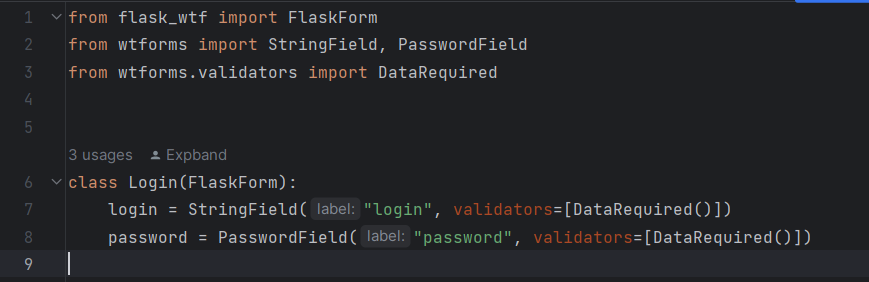


Цей клас зчитує значення з ini файлу, завдяки цьому приватна та чутлива інформація залишається поза кодом, через що стає складніше її отримати третім лицям.

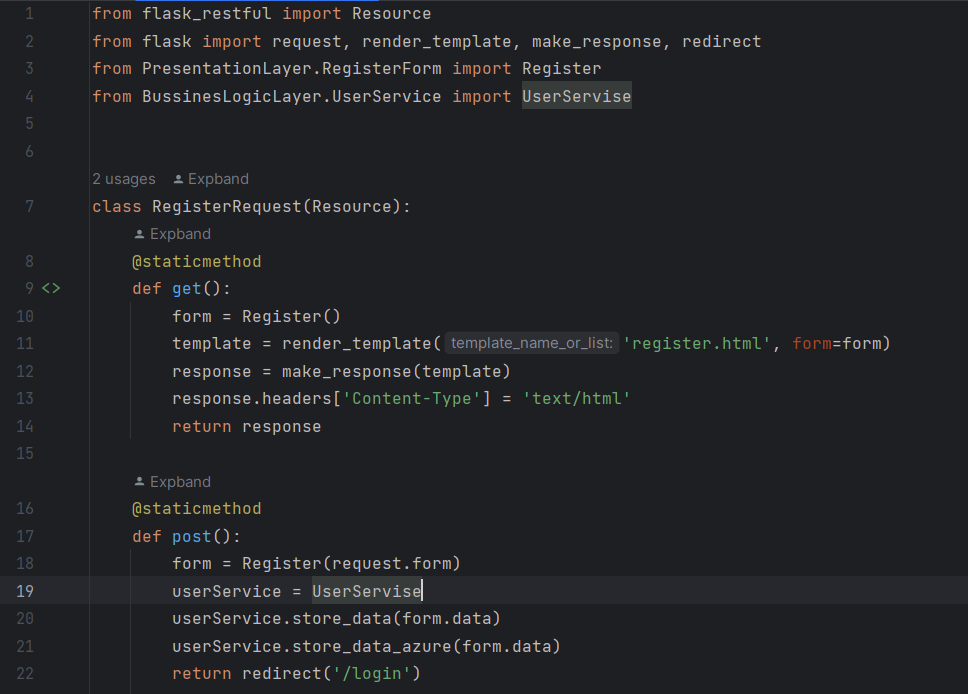
Архітектуру додатку я обрах трьохрівневу, тому, що вона гарно підходить для нарощування функціоналу, та використання різних баз данних з однією моделлю. Структура проекту виглядає має вигляд класичної трирівневої архітектури та поділена на три рівні



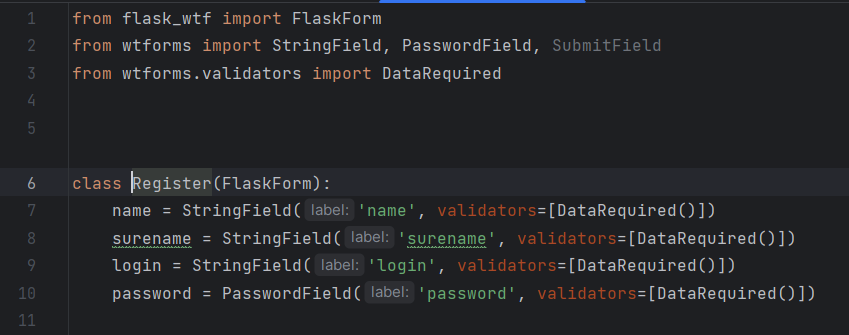
Кожен рівень відповідальний за конкретний аспект, та виконує тільки певний спектр дій.

Рівень представлення складається з контролера користувача, контролера реєстрації, та двох форм для них.

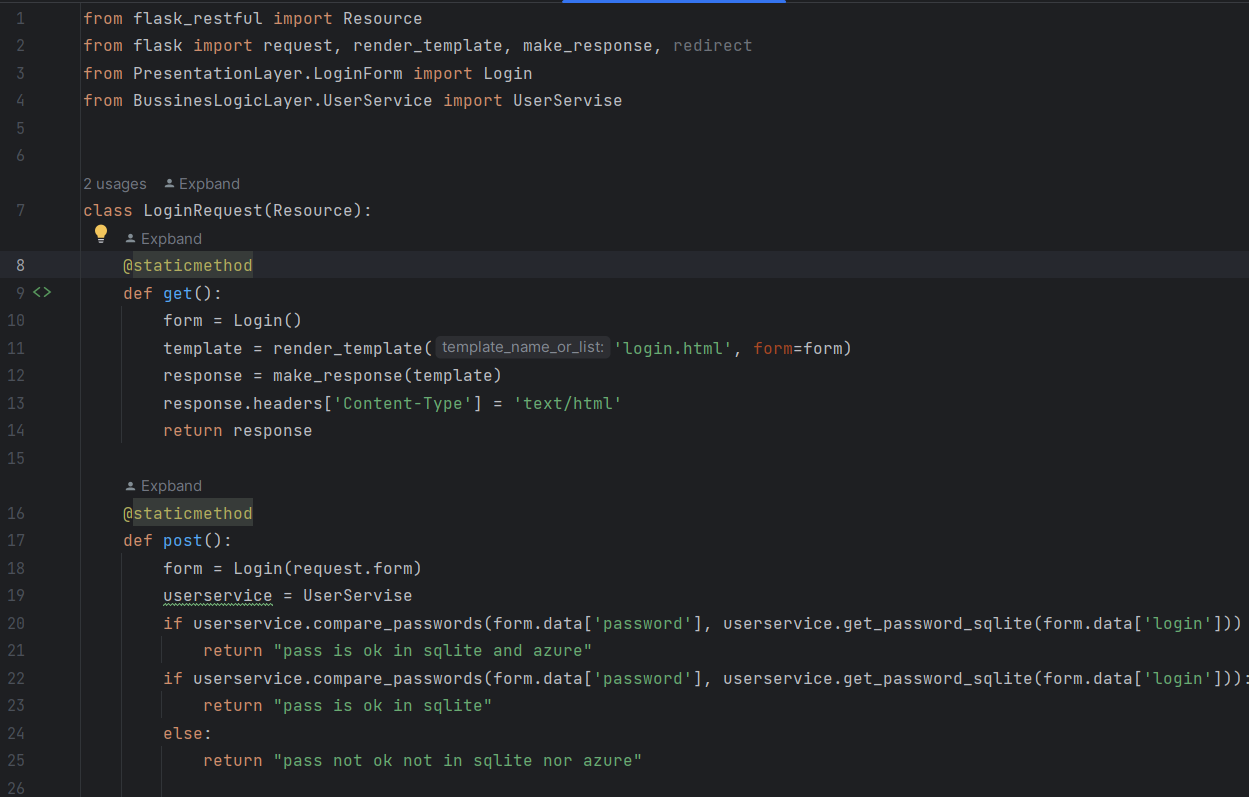
1. Форма логування



1. Контролер логування



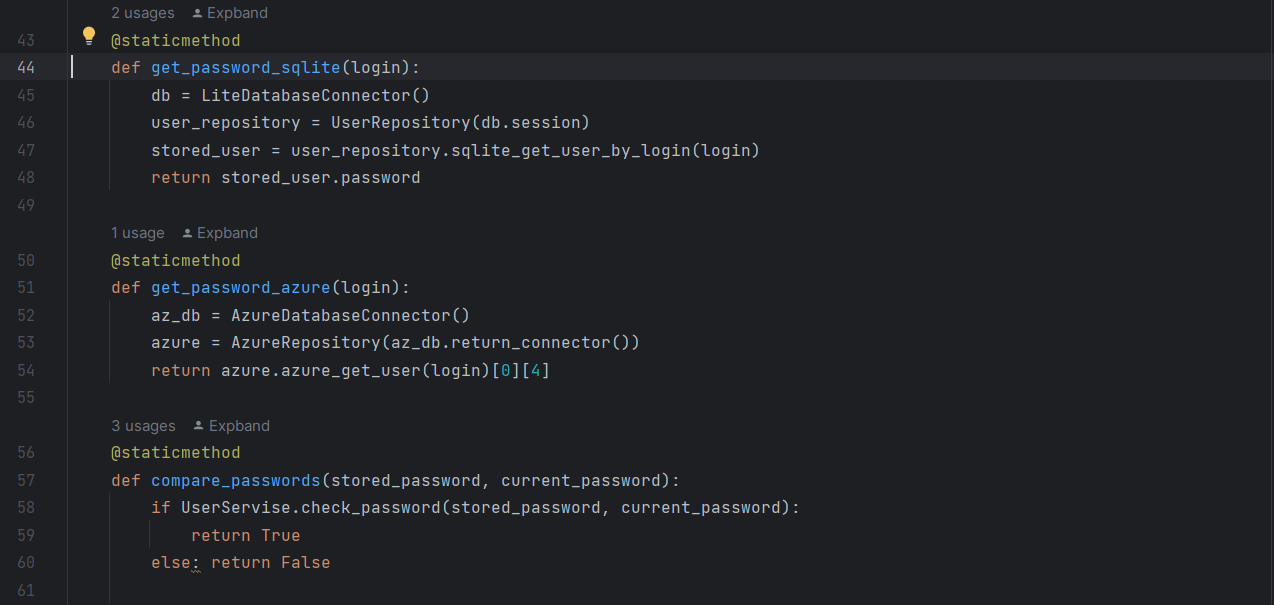
1. Форма реєстрації



1. Контролер реєстрації

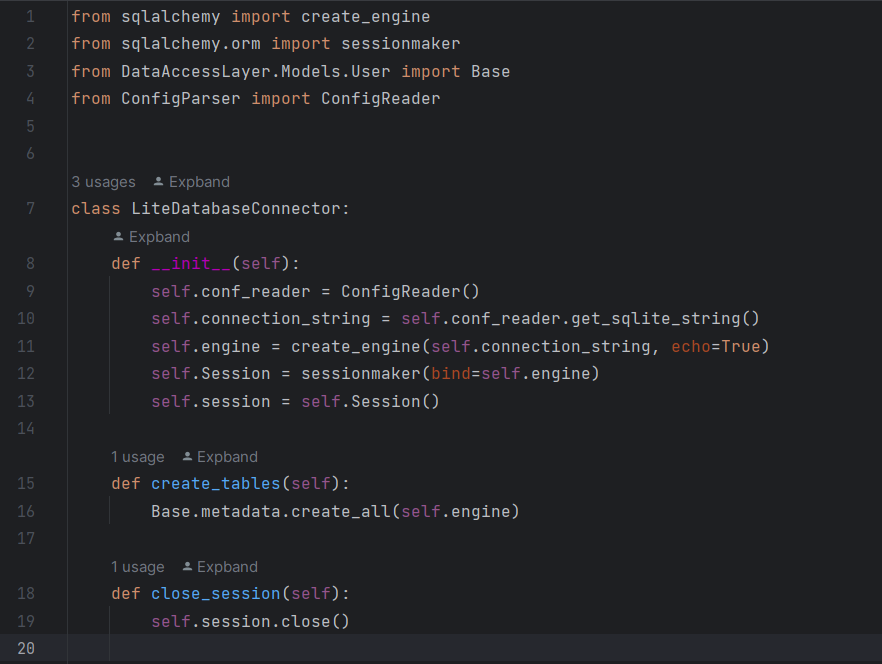
Ці компоненти відповідають первинний вигляд данних, та їх отримання з користувацького інтерфейсу. Також за їх передачу на рівень бізнес логіки. Через те, що проект реалізовує тільки реєстрацію та логування - рівень бізнес логіки досить простий.



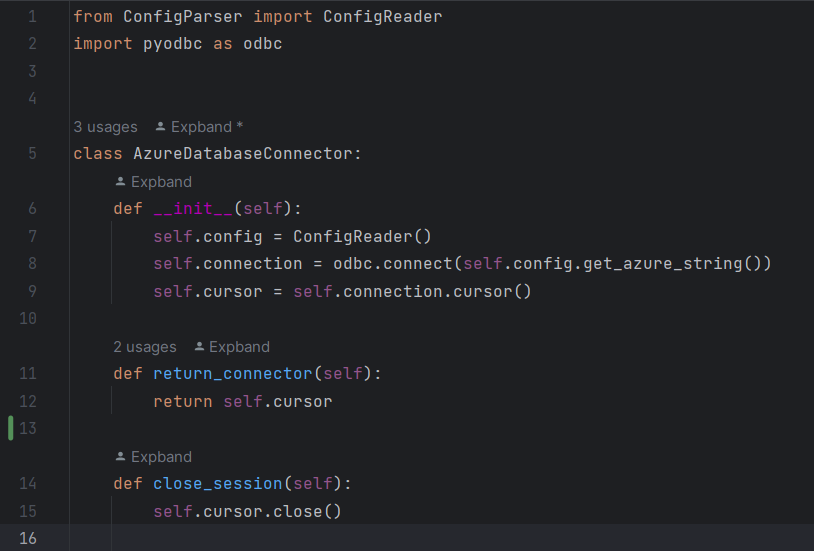


Сервіс отримує данні. Після чого в різних випадках веде себе по різному. При реєстрації на рівні сервісу генерується uuid. Я використовую uuid для того, щоб не було проблем в разі заміни якихось записів, їх видалень і так далі. Для того, щоб уникнути конфліктів, також гарантовано мати унікальний ID, я використовую 36-ти значний рядок в якості ідентифікатора. Також, якщо користувач реєструєця, потрібно зберігати його пароль. Зберігати пароль у відкритому вигляді, так само, як і будь яку чутливу інформацію, у відкритому вигляді не можна. Тому я використовую хешування паролю за допомогою bcrypt. При логуванні користувача, ми викликаємо метод, який отримує по логіну користувача так звану “сіль” паролю. Сіль пародю це пароль в хешованому вигляді. Після чого, пароль який надав користувач порівнюється з сіллю паролю, яка збережена в базі данних для цього користувача. В разі їх співпадіння - користувач отримує повідомлення pass is ok in sqlite and azure, якщо в обох базах данних пароль співпадає, або pass is ok in sqlite, якщо співпадає тільки в локальній базі данних.

Наступним в ієрархії є рівень представлення данних. Він відповідає за взаємодію з базами данних та застосування моделей.

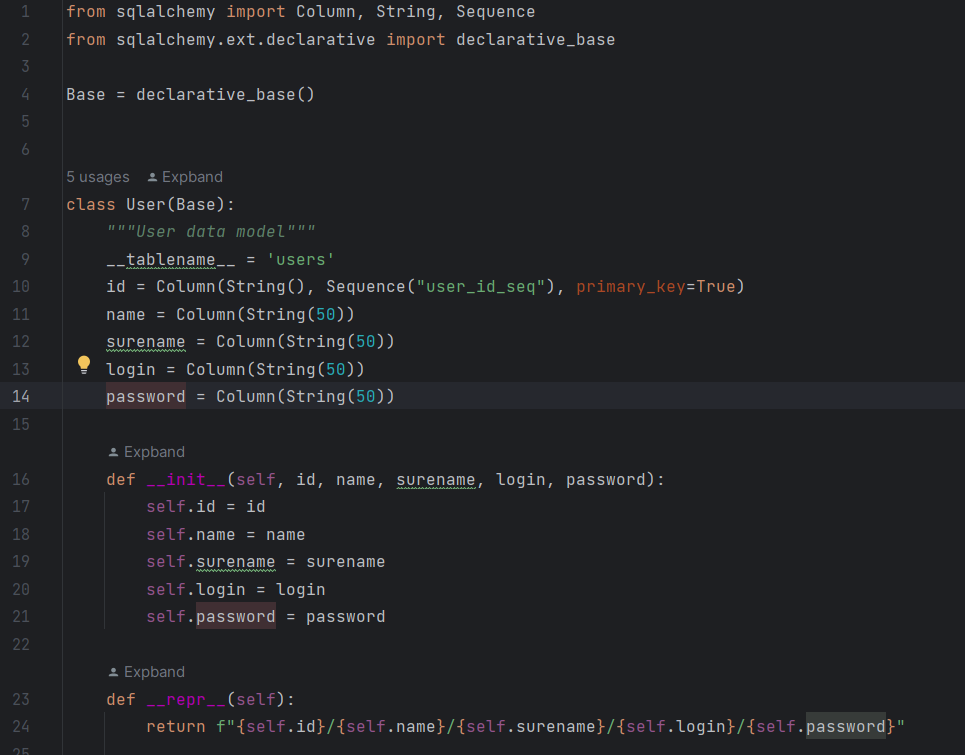
Для підключення до обох баз данних я використовую відповідні класи-конектори. 

1. Коннектор для SQLite



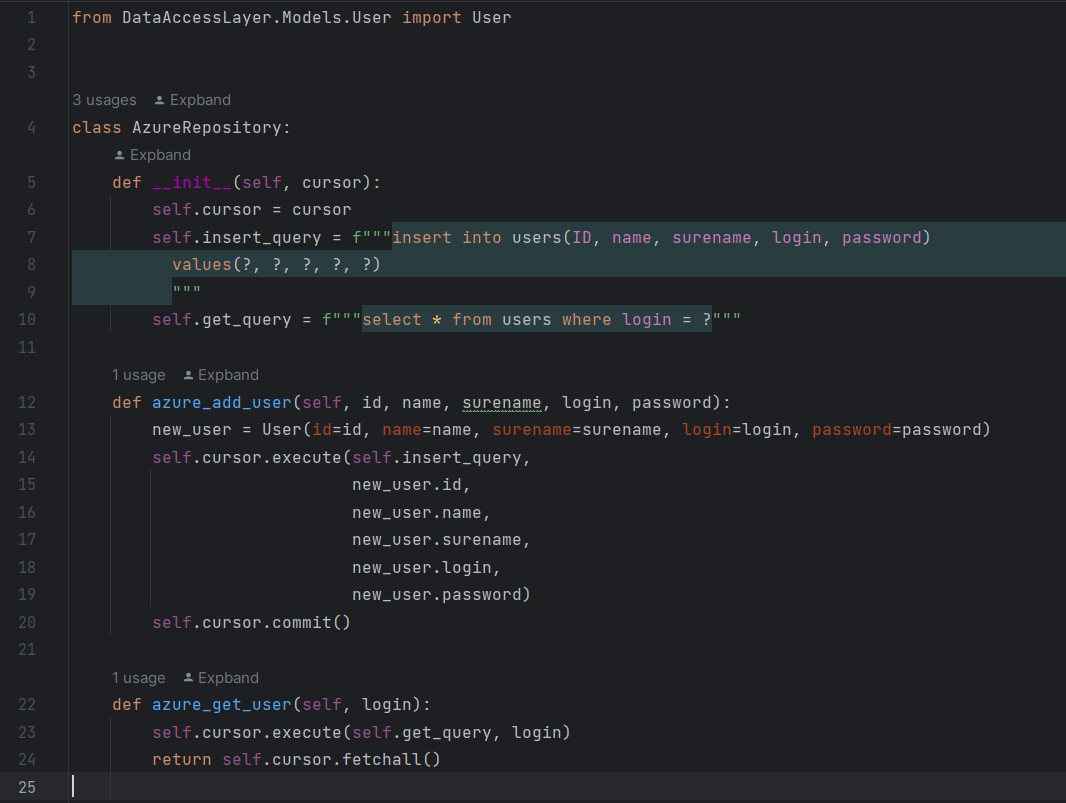
1. Коннектор для Azure

Ці класи відповідають за ініціацію підключення до бази данних. Для збереження данних я використовую модель користувача. Модель говорить, як повинен структуруватися об’єкт користувач.

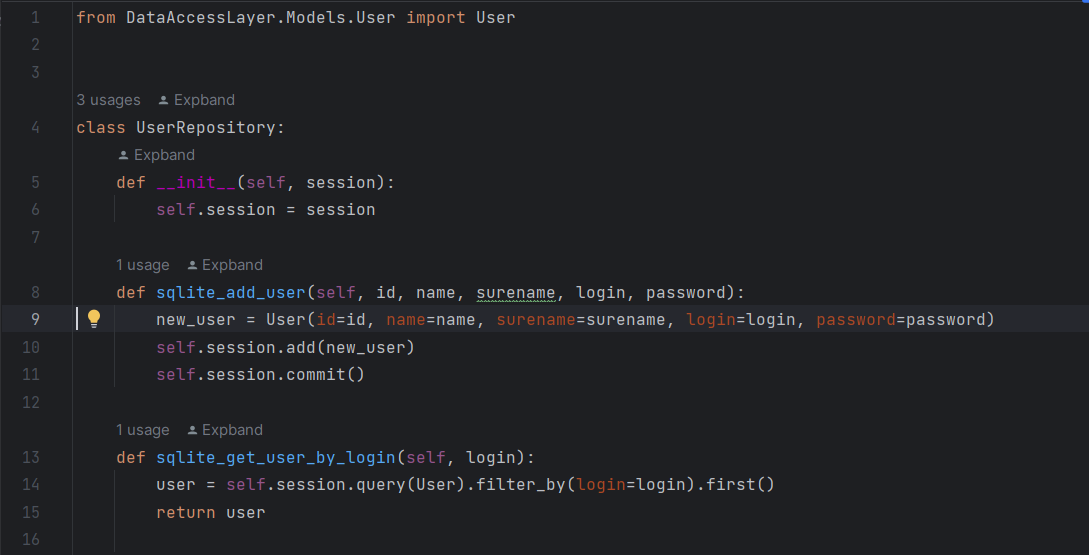


1. Модель користувача

Після цього виконуються саме операції з базами данних. Виконується дві операції запису та читання.



1. Репозиторій для Azure. Тут можна побачити, яким саме чином виконуються запити. В полях класу записані запити, а в функціях підставляються значення



1. З базою данних SQLite виконуються запити за допомогою SQLalchemy, тому не потрібно писати запити на SQL