МІНІСТЕРСТВО ОСВІТИ ТА НАУКИ УКРАЇНИ

ЛЬВІВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ імені ІВАНА ФРАНКА

**Звіт**

**з курсу «Об'єктно-орієнтовний аналіз і проектування ПЗ»**

Виконали:

Мочернюк Данило

Панковець Злата

Джура Михайло

(ПМі-53)

**Опис продукту**

Метою роботи є створення веб-застосунка за допомогою Python фреймворку Flask для веб-розробки.



У результаті виконання роботи розроблено сервіс, для побудови простої моделі машинного навчання, яка опрацьовує файл з набором даних і робить передбачення невідомої величини спираючись на дані з цього набору.

**Архітектура**

При розробці застосунка було використано модель клієнт-сервер.

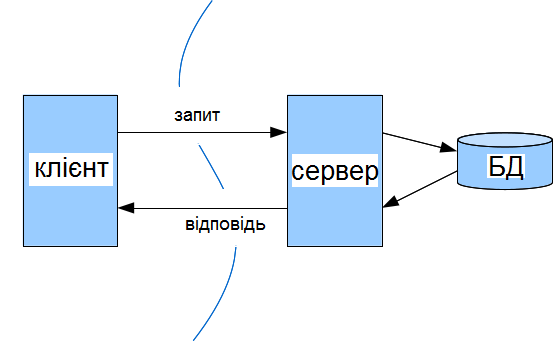
Програма клієнт / сервер складається з клієнтської програми, яка використовує послуги, що надаються серверною програмою. Клієнт надсилає запити на послуги у сервера, викликаючи функції в серверній програмі. У розподіленому обчислювальному середовищі, коли клієнтська програма та серверна програма виконуються на різних машинах і, можливо, навіть на різних платформах, клієнт і сервер взаємодіють через рівень зв'язку, який часто називають проміжним програмни забезпеченням.

Функції, які реалізуються на сервері:

* зберігання, доступ, захист і резервне копіювання даних;
* обробка клієнтського запиту;
* оправлення результату (відповіді) клієнту.

Функції, які реалізуються на стороні клієнта:

* надання користувацького інтерфейсу;
* формування запиту до сервера і його відправка;
* отримання результатів запиту і відправка додаткових команд (запитів на додавання, оновлення або видалення даних).

****

Було використано Werkzeug для забезпечення інтерфейсу WSGI.



Інтерфейс WSGI

WSGI (Web-Server Gateway Interface) є нащадком CGI(Common Gateway Interface). Коли веб почав розвиватися, CGI розростався через підтримку величезної кількості мов і через відсутність інших рішень. Однак, таке рішення було повільним і обмеженим. WSGI був розроблений як інтерфейс для маршрутизації запитів від веб-серверів (Apache, Nginx і т.д.) на веб-додатки. У найпростішому випадку WSGI складається з двох основних сутностей:

* веб-сервер (Nginx, Apache і т. д.);
* веб-додаток, написаний на мові Python.

scikit-learn - безкоштовна програмна бібліотека машинного навчання для мови програмування Python.

**Storage**

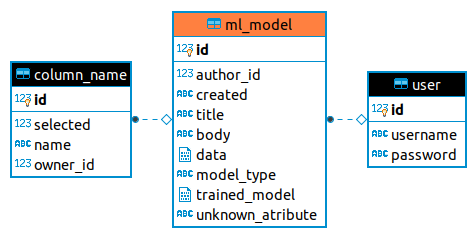
SQLAlchemy - ORM mapper. ORM (англ. *Object-relational mapping*, Об'єктно-реляційна проекція) — технологія програмування, яка зв'язує бази даних з концепціями об'єктно-орієнтованих мов програмування, створюючи «віртуальну об'єктну базу даних».



SQLite - полегшена реляційна система керування базами даних.

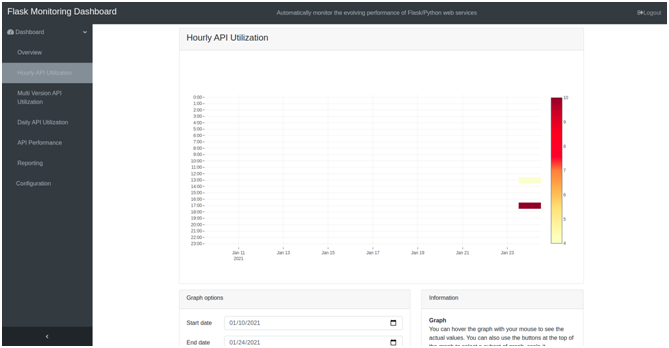


ER діаграма бази даних



**In-built Flask monitoring**

Flask забезпечує потужний інструмент для моніторингу. Цей інструмент можна легко підключити до будь-якої веб-аплікацій для того щоб отримувати зручну статистику про утилізацію API.



**Security model**

Було використано Flask-security, що дозволило швидко додавати механізми безпеки для цього застосунку, як от керування ролями, шифрування пароля, реєстрація користувача, аутентифікація на основі сесії.

Аутентифікація на основі сесії здійснюється за рахунок розширення Flask-Login.

Flask-Security реалізує дуже просте управління групами користувачів. З допомогою розширення Flask-Principal можна реалізувати контроль доступу на основі ролей.

За замовчуванням, паролі зберігаються у вигляді звичайного тексту, але це можна легко налаштувати через алгоритм шифрування паролів, котрий реалізовано через passlib.

Flask-Security надає функціонал для реєстрації користувачів, для реєстрації нового користувача потрібно ввести лише адресу електронної пошти та пароль.

Flask-Security відправляє повідомлення з посиланням на електронну пошту користувачеві, по якій вони можуть скинути свій пароль. Коли пароль буде скинутий, вони автоматично увійдуть в систему і з того моменту зможуть використовувати новий пароль.

**SOLID / GRASP Compliant**

Принципи SOLID

SOLID - це абревіатура складена з перших літер п'яти базових принципів об'єктно-орієнтованого програмування та дизайну запропонована Робертом Мартіном.

Принципи SOLID використовуються для дизайну та розробки таких програмних систем, які, з великою ймовірністю, зможуть тривалий час розвиватися, розширюватися і підтримуватися.

Дані принципи дуже тісно корелюють з принципами об'єктно-орієнтованого програмування. Зокрема принцип підстановки Барбари Лісков уособлює успадкування, він може трактуватись так: об'єкт, клас якого успадкований від іншого класу, отримує методи останнього. Принцип інверсії залежностей — це більш розширене трактування поліморфізму. А принцип єдиного обов'язку тісно пов'язаний з інкапсуляцією, що також передбачає існування інкапсульованого (відокремленого) набору методів.

|  |  |  |  |
| --- | --- | --- | --- |
| **Літера** | **Абревіатура** | **Назва (англ.)** | **Назва (укр.)** |
| **S** | SRP | Single responsibility principle | Принцип єдиного обов’язку |
| **O** | OCP | Open/Closed principle | Принцип відкритості / закритості |
| **L** | LSP | Liskov substitution principle | Принцип підстановки Лісков |
| **I** | ISP | Interface segregation principle | Принцип розділення інтерфейсу |
| **D** | DIP | Dependency inversion principle | Принцип інверсії залежностей |

Детальне пояснення кожного з принципів:

S - Кожен об'єкт має виконувати лише один обов'язок.

O - Програмні сутності повинні бути відкритими для розширення, але закритими для змін. Розширення певного класу/інтерфейсу може здійснюватись через його успадкування.

L - Об'єкти в програмі можуть бути зміненими їх нащадками без зміни коду програми.

I - Багато спеціалізованих інтерфейсів краще за один універсальний. Інтерфейс може бути поділений на спеціалізовані ще на стадії проектування, заради майбутньої гнучкості програмних компонентів.

D - Залежності всередині системи будуються на основі абстракцій, що не повинні залежати від деталей; навпаки, деталі мають залежати від абстракцій. Модулі вищих рівнів не мають залежати від модулів нижчих рівнів.

Принципи GRASP

GRASP (англ. General Responsibility Assignment Software Patterns) — набір патернів (шаблонів, принципів), що дозволяють вирішувати проблеми розподілу обов'язків між різними класами. За своєю суттю, цей набір патернів більш абстрактний, ніж загально відомий каталог шаблонів від «Банди чотирьох» (GOF-шаблони).

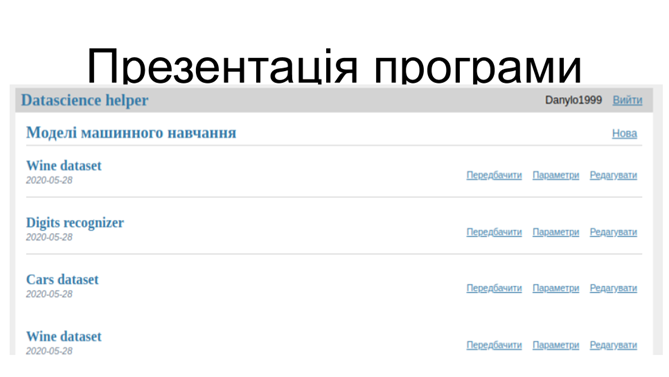
До складу шаблонів GRASP входить 9 шаблонів:

* Information Expert
* Creator
* Low Coupling
* High Cohesion
* Controller
* Polymorphism
* Pure Fabrication
* Indirection
* Protected Variations

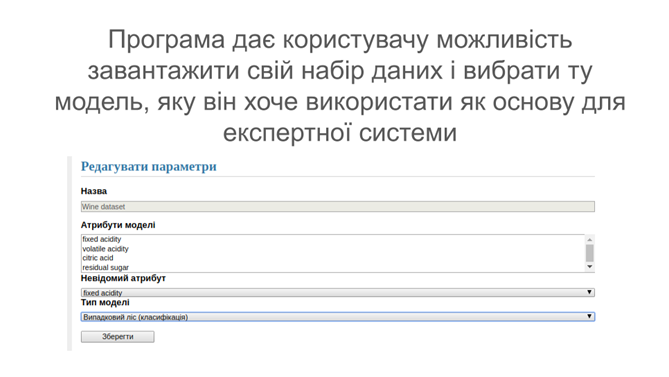
Відомо дев'ять GRAPS шаблонів, спочатку описаних у книзі Крейга Ларман «Застосування UML і шаблонів проектування». На відміну від звичних читачеві патернів з Банди Чотирьох, GRAPS патерни не мають вираженої структури, чіткої області застосування і конкретної розв'язуваної проблеми, а лише являють собою узагальнені підходи / рекомендації / принципи, використовувані при проектуванні дизайну системи. «Applying UML and Patterns — An Introduction to Object-Oriented Analysis and Design and Iterative Development»

|  |  |  |
| --- | --- | --- |
| **GRASP**  **(General Responsibility Assignment Software Patterns)** | | |
| **№** | **Назва (англ.)** | **Назва (укр.)** |
| **1** | Information Expert | Інформаційний експерт |
| **2** | Creator | Творець |
| **3** | Controller | Контролер |
| **4** | Low Coupling | Низька зв'язаність |
| **5** | High Cohesion | Високе зачеплення |
| **6** | Polymorphism | Поліморфізм |
| **7** | Pure Fabrication | Чиста вигадка |
| **8** | Indirection | Перенаправлення |
| **9** | Protected Variations | Стійкий до змін |

**Апробація**

****

Програма дає користувачу можливість завантажити свій набір даних і вибрати ту модель, яку він хоче використати як основу для експертної системи.



Після того, як модель було створено, користувач може використовувати її для здійснення передбачень невідомого атрибута, використовуючи свої дані.

