МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ “КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ

ІМЕНІ ІГОРЯ СІКОРСЬКОГО” КАФЕДРА ПРИКЛАДНОЇ МАТЕМАТИКИ

КУРСОВА РОБОТА

з навчальної дисципліни “Бази даних та інформаційні системи-2” на тему

СЕРВІС ДЛЯ ГЕНЕРАЦІЇ ТЕСТІВ ДЛЯ ПЕРЕВІРКИ КОДУ (ФУНКЦІЙ)

Виконала студентка групи КМ-62 Городнюк О.С.

Керівник ТЕРЕЩЕНКО І.О.

Оцінка: Кількість балів:

КИЇВ – 2019

ЗАВДАННЯ НА ВИКОНАННЯ КУРСОВОЇ РОБОТИ

Створити інформаційну систему, що дозволить протестувати усі вкладені для скрипту функції на основі належності їх атрибутів до класів даних, побудувати модель тестових випадків та можливих результатів виконання цих функцій. Для цього реалізувати вибірку невбудованих функцій, що знаходяться у файлі, провести аналіз аргументів та поставити у відповідність до випадкових значень даного типу даних. Реалізувати перевірку за допомогою цих даних та знайти результат виконання.

АНОТАЦІЯ

В пояснювальній записці до курсової роботи описано основні етапи розробки інформаційної системи. Перший розділ містить інформацію про виконання перед проектного дослідження. У другому розділі надано детальну постановку задачі. В третьому розділі виконано моделювання бізнес-процесів. У четвертому розділі визначено модель даних, описано сутності інформаційної системи, а в розділі 5 надано графічне представлення моделі.

РЕФЕРАТ

В ході виконання курсового проекту було створено інформаційну систему на базі веб-фреймворку Flask. База даних, на якій виконано завдання – PostgreSQL 11. ORM-модель, що використовується для CRUD операцій у Python, створюється на Flask-SQLAlchemy — бібліотека, що реалізована на базі SQLAlchemy.

Додавання даних валідується на WTForms з використання налаштувань від Flask-Bootstrap. З Python до HTML-сторінок передається інформація та відображається за допомогою шаблонізатора Jinja.

ЗМІСТ

[ВСТУП 6](#_bookmark0)

1. [АНАЛІЗ ПІДПРИЄМСТВА АВТОМАТИЗАЦІЇ 7](#_bookmark1)
2. [ПОСТАНОВКА ЗАДАЧІ 8](#_bookmark2)
   1. [Категорії користувачів 8](#_bookmark3)
   2. [Класи даних 9](#_bookmark4)
   3. [Бізнес-правила 12](#_bookmark5)
   4. [Матриця елементарних події(сценарії) 12](#_bookmark6)
3. [МОДЕЛЮВАННЯ БІЗНЕС-ПРОЦЕСІВ 16](#_bookmark7)
4. [ІНФОЛОГІЧНЕ ПРОЕКТУВАННЯ 17](#_bookmark8)
5. [ДАТАЛОГІЧНЕ ПРОЕКТУВАННЯ 18](#_bookmark9)

[ВИСНОВКИ 20](#_bookmark10)

[СПИСОК ВИКОРИСТАНИХ ЛІТЕРАТУРНИХ ДЖЕРЕЛ 22](#_bookmark11)

ВСТУП

Сьогодні вже існує безліч сервісів для генерації тестових випадків або аналізу функцій. Для прикладу генерації можна розглянути [http://www.generatedata.com](http://www.generatedata.com/) та [https://www.mockaroo.com](https://www.mockaroo.com/) , а для аналізу сервісів - <https://github.com/klen/mixer> (тести для Django, Flask додатків), <https://ddt.readthedocs.io/en/latest> (модуль Python для генерації заданих функцій за допомогою Unittest). Але метою курсової роботи є поєднання ідей генерації та тестування, яке дозволить швидко та якісно оцінювати можливості функцій та вказувати на їх сильні та слабкі місця (якщо такі наявні).

Додатковою цінністю додатку буде те, що функціонал не отримує допомогу від сторонніх сервісів тестування (pytest, unittest, nose, тощо), які надають інформацію про правильність виконання, а вказує де та чому виникли помилки та пропозиції їх вирішення.

1 АНАЛІЗ ПІДПРИЄМСТВА АВТОМАТИЗАЦІЇ

Відповідно до постановки задачі, маємо процеси завантаження файлу, аналізу його змісту, генерування тестових випадків та, власне, тестування. За бізнес логікою (ст. 12) лише зареєстрований користувач може повноцінно користуватись сервісом, тому маємо наступну ієрархію процесів (рис. 1.1)

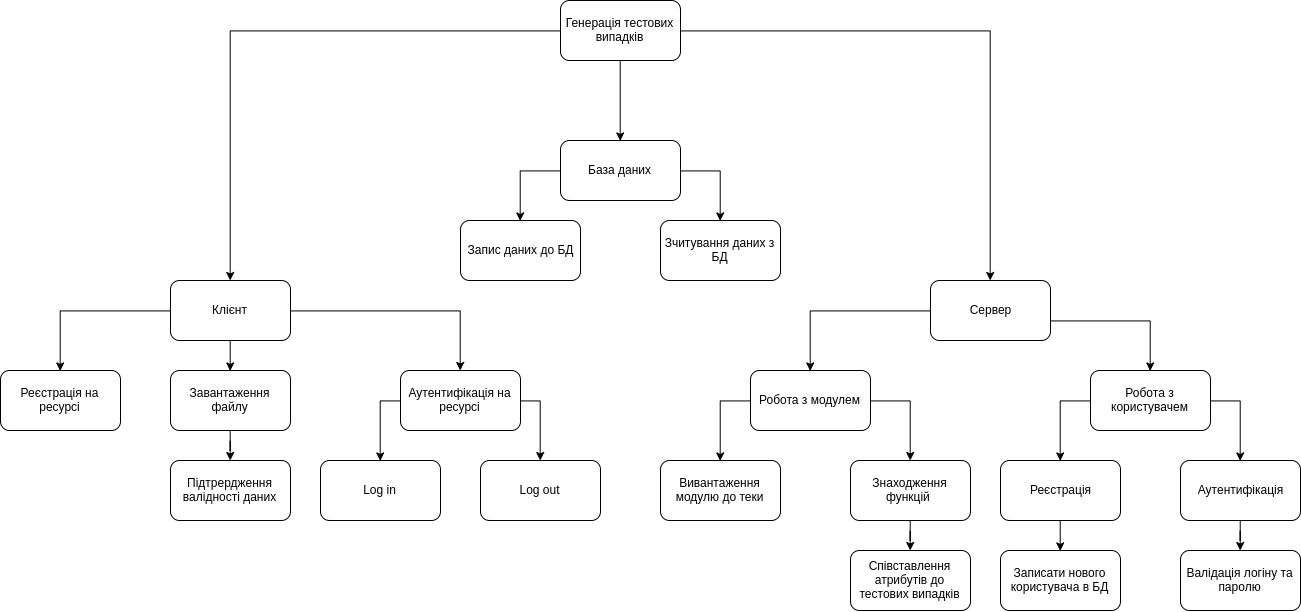


Рисунок 1.1 – Ієрархія процесів

2 ПОСТАНОВКА ЗАДАЧІ

Основна відмінність простої автоматизації робочого процесу від пропонованого мікросервісу в тому, що кожного разу сервіс підлаштовується до окремих типів функцій, не знаючи про її функціонал, ідею, аргументи та даних, які функція повинна повернути. Залежно від цих критеріїв — сервіс буде надавати унікальні тестові випадки та реалізовувати їх незалежно від очікуваного результату. В окремих випадках буде надано підказки щодо написання функції.

Основна функція мікросервісу: генерація тестових випадків.

Вхідні дані мікросервісу: файл з розширенням .py.

Результат роботи мікросервісу: тестові випадки та результат виконання функції з ними.

Далі розглянемо основні категорії користувачів сервісу, класи даних та елементарні події, що відбуваються у ході користування системою.

* 1. Категорії користувачів

У сервісі, що реалізовано в курсовому проекті наявна реєстрація та аутентифікація користувачів. Лише користувачі, що увійшли до системи, можуть користуватись нею.

* 1. Класи даних

Опишемо наявні у системі класи даних у таблицях 2.1 –2.6.

Таблиця 2.1 – Клас даних «Юзер»

|  |  |  |
| --- | --- | --- |
| **Сутність** | Юзер | |
| **Опис сутності** | Користувач системи | |
| **Атрибути сутності** | **Опис атрибуту** | **Пов’язана сутність із**  **атрибутом** |
| Id | Унікальний  ідентифікатор | File |
| Email | Email користувача | - |
| Name | Ім’я користувача | - |
| Password | Пароль для входу до системи | - |

Таблиця 2.2 – Клас даних «TestCase»

|  |  |  |
| --- | --- | --- |
| **Сутність** | TestCase | |
| **Опис сутності** | Унікальні тестові випадки | |
| **Атрибути сутності** | **Опис атрибуту** | **Пов’язана сутність із**  **атрибутом** |
| ID | Універсальний  ідентифікатор | - |
| file\_id | Файл, у якому виявлено дефектний тестовий випадок. | File |
| function\_name | Назва функції | - |
| Case | Тестовий випадок | - |
| result | Результат виконання. Exception. | - |

Таблиця 2.3 – Клас даних «File»

|  |  |  |
| --- | --- | --- |
| **Сутність** | File | |
| **Опис сутності** | Файли, що були завантажені. | |
| **Атрибути сутності** | **Опис атрибуту** | **Пов’язана сутність із**  **атрибутом** |
| ID | Універсальний  ідентифікатор | TestCase |
| Upload\_time | Час завантаження | - |
| User | Користувач, який завантажив файл | - |
| Name | Назва файлу | - |
| language | Мова програмування | Language |

Таблиця 2.4 – Клас даних «Language»

|  |  |  |
| --- | --- | --- |
| **Сутність** | Language | |
| **Опис сутності** | Мови програмування | |
| **Атрибути сутності (1)** | **Опис атрибуту (2)** | **Пов’язана сутність із**  **атрибутом (3)** |

|  |  |  |
| --- | --- | --- |
| 1 | 2 | 3 |
| ID | Універсальний  ідентифікатор | File |
| Назва | Назва мови | - |
| Версія | Версія мови програмування | - |
| Дата релізу | Дата релізу мови | – |

* 1. Бізнес-правила

1. Лише зареєстрований юзер може повноцінно користуватись сервісом.

2. Незареєстрованому юзеру буде надано лише частину документації.

3. При реєстрації користувачеві буде надіслано лист на пошту.

4. Завантажувати можливо лише файли з розширенням .py. В іншому випадку отримаємо помилку.

5. Після завантаження користувачеві потрібно підтвердити що його скрипт було завантажено правильно, переглянувши його зміст та перелік функцій у ньому.

6. Лише функції можуть бути переглянуті програмою. Класи та інші структури будуть ігноруватись.

7. У випадку помилки у завантаженому коді - буде показано traceback цієї помилки.

8. Для ліпшого тестування атрибути функції повинні бути вказані через сигнатуру класу (def foo(a: int): ...)

9. Колаборація з проектом генерації документації пропонує скласти опис даного скрипту на іншому сервісі.

10. Можливо переглянути тестові випадки для файлів, що були завантажені раніше.

* 1. Матриця елементарних події (сценарії)

Перший рівень містить багато однотипних процесів, тому в їх рамках будемо описувати лише один, а інші – аналогічні.

Таблиця 2.7 – Процес створення нового запису в сутності

|  |  |
| --- | --- |
| Назва процесу | Додавання запису до сутності |
| Сутності | Будь-яка |
| Вхідні атрибути сутності | В процесі додавання вхідні атрибути  пусті, бо запис ще не вставлено. |
| Опис функціоналу | Дані з форми на веб-сторінці записуються у поля класу та зберігаються у базі даних за допомогою моделі серверу. |
| Змінені атрибути сутності | Рядок вставлених даних |

Таблиця 2.8 – Процес редагування запису в сутності

|  |  |
| --- | --- |
| Назва процесу | Редагування запису у сутності |
| Сутності | User |
| Вхідні атрибути сутності | Рядок, отриманий за ID поля, яке  збираємося відредагувати |
| Опис функціоналу | Дані приходять із серверу та пишуться у поля форми. Після цього у формі користувач змінює значення атрибутів. Дані зчитуються та відправляються на  сервер. |
| Змінені атрибути сутності | Рядок відредагованих даних |

Таблиця 2.10 – Процес імпортування модулів

|  |  |
| --- | --- |
| Назва процесу | Імпортування модулів |
| Сутності | File |
| Вхідні атрибути сутності | Назва модулю |
| Опис функціоналу | Імпортування модулів для обробки тестових випадків для функцій у ньому. |
| Змінені атрибути сутності | Процес не змінює атрибути |

Таблиця 2.11 – Процес підбору тестових випадків

|  |  |
| --- | --- |
| Назва процесу | Підбір тестових випадків |
| Сутності | TestCase |
| Вхідні атрибути сутності | Класи атрибутів функції |
| Опис функціоналу | Виконується вставка випадкових тестових випадків задля знаходження помилок. |
| Змінені атрибути сутності | Запис до таблиці TestCase у разі знаходження |

3 МОДЕЛЮВАННЯ БІЗНЕС-ПРОЦЕСІВ

Змодельований Use-Case для процесів, що можна виконувати наразі у веб- застосунку, зображено на рисунку 3.1.

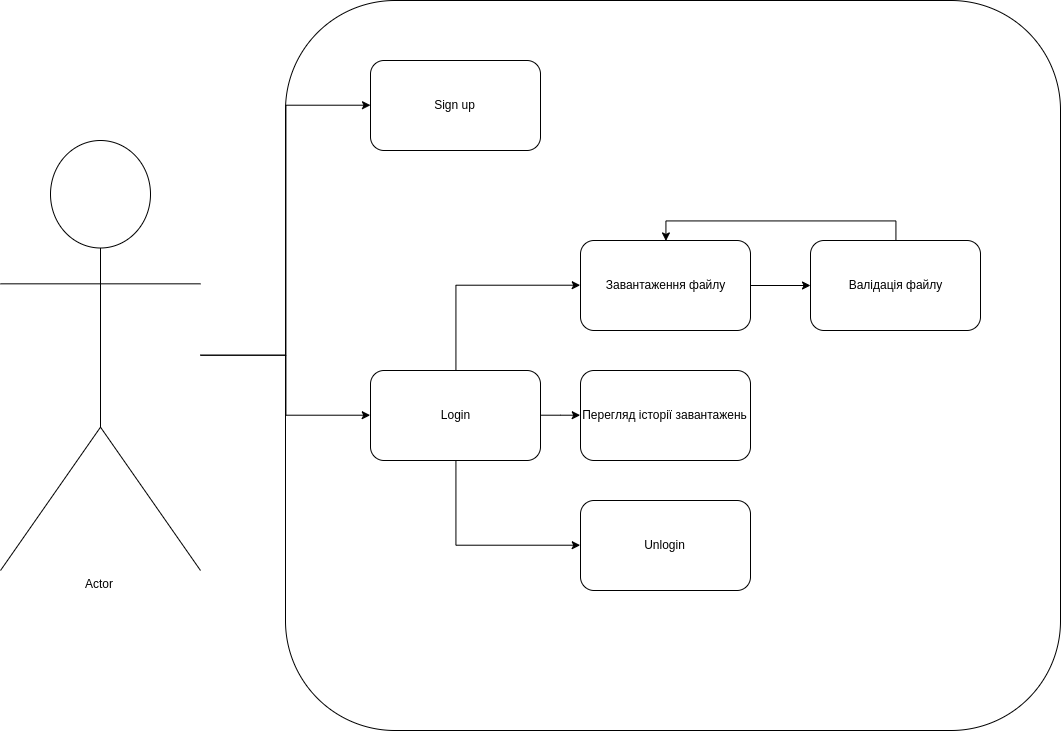


Рисунок 3.1 – Use-case diagram для веб-застосунку

1. ІНФОЛОГІЧНЕПРОЕКТУВАННЯ

Змоделюємо сутності та зв’язки звичною мовою.

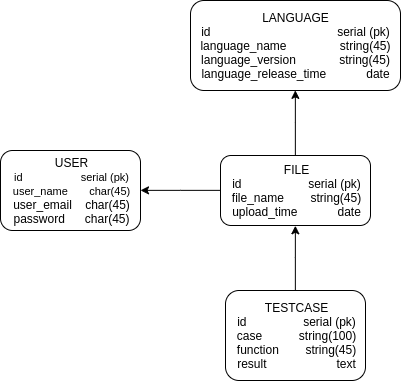
а) Користувач реєструється та завантажує файл.

б) Файл — це скрипт, який написаний на певній мові програмування.

в) Тестові випадки надаються для конкретного файлу (функції).

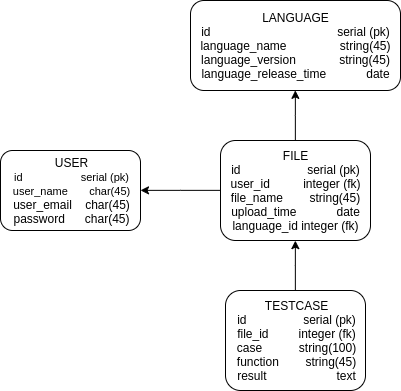
Усі ключі будуть реалізовані за допомогою типу serial для найбільш можливого рівня ізоляції.

Концептуальна діаграма зображена на рисунку 4.1.

Рисунок 4.1 – Концептуальна діаграма бази даних

1. ДАТАЛОГІЧНЕ ПРОЕКТУВАННЯ

Наступний крок – перетворити концептуальну модель на логічну та оптимізувати її. При такому перетворенні вводять штучні ключі, а на атрибути, що були ключовими у концептуальній моделі накладається обмеження унікальності. Оптимізована логічна модель зображена на рисунку 5.1.

Рисунок 5.1 – Оптимізована логічна модель даних

Далі логічна модель перетворюється на фізичну.

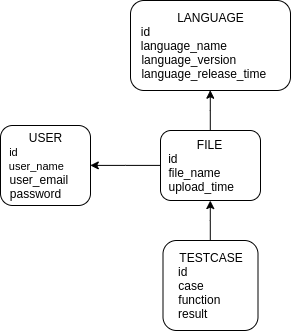


Рисунок 5.2 – Фізична модель даних

ВИСНОВКИ

В ході виконання курсової роботи було визначено наступні етапи впровадження проекту: побудова ієрархії процесів, обґрунтування постановки задачі, визначення сценаріїв перебігу елементарних процесів, визначення класів даних, сутностей та атрибутів, побудова моделей даних, інтеграція відомих рішень та їх покращення новим функціоналом.

Як результат виконання проекту досягнуто: інформаційна система, що дозволяє переглядати, додавати, змінювати зміст сутностей. База даних, що взята за основу – PostgreSQL, а ORM модель на Python побудована за допомогою SQLAlchemy. Веб-застосунок збудований на базі фреймворків Flask та Bootstrap-4. Ресурс розгорнутий на хостингу Heroku.

Окрема вкладка, зроблена на сайті, дозволяє переглядати історію завантажень користувача та тестові випадки, які не підійшли під функції, завантажені на сервер.

СПИСОК ВИКОРИСТАНИХ ЛІТЕРАТУРНИХ ДЖЕРЕЛ

1. Create An API To Deploy Machine Learning Models Using Flask and Heroku [Електронний ресурс] – Режим доступу до ресурсу:<https://towardsdatascience.com/create-an-api-to-deploy-machine-learning-models-using-flask-and-heroku-67a011800c50>.
2. PostgreSQL 10.11 Documentation [Електронний ресурс] – Режим доступу до ресурсу:[https://www.postgresql.org/docs/10/index.html.](https://www.postgresql.org/docs/10/index.html)
3. SQLAlchemy 1.3 Documentation [Електронний ресурс] – Режим доступу до ресурсу:<https://docs.sqlalchemy.org/en/13/>.
4. WTForms Documentation [Електронний ресурс] – Режим доступу до ресурсу:<https://wtforms.readthedocs.io/en/stable/>.