МІНІСТЕРСТВО ОСВІТИ НАУКИ УКРАЇНИ
СТРУКТУРНИЙ ПІДРОЗДІЛ «ФАХОВИЙ КОЛЕДЖ ЕКОНОМІКИ ТА ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ ПРАТ «ПВНЗ «ЗІЕІТ»

Циклова комісія з інформаційних технологій

ДО ЗАХИСТУ ДОПУЩЕНИЙ
Голова _____
С.І. Левицький

ВИПУСКНА РОБОТА МОЛОШОГО СПЕЦІАЛІСТА
РОЗРОБКА ВЕБСЕРВІСУ «КАРТКИ ДЛЯ ЗАПАМ’ЯТОВУВАННЯ» З ВИКОРИСТАННЯМ PYTHON ТА MYSQL

Виконав
ст. гр. ІПЗ – 118к9 _____ О.С. Бірюков

Науковий керівник
ст. викл. _____ К.В. Дереза

Запоріжжя
2022

СТРУКТУРНИЙ ПІДРОЗДІЛ «ФАХОВИЙ КОЛЕДЖ ЕКОНОМІКИ ТА ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ ПРАТ «ПВНЗ «ЗІЕІТ»

Циклова комісія з інформаційних технологій

ЗАТВЕРДЖУЮ

Голова

Левицький С.І.

_____

__.__.__ р.

З А В Д А Н Н Я

НА ВИПУСКНУ РОБОТУ МОЛОДШОГО СПЕЦІАЛІСТА

Студенту гр. ІПЗ – 118к9

Спеціальності: «Інженерія програмного забезпечення»

Бірюкова Олексія Сергійовича

(прізвище, ім’я, по батькові)

1. Тема: Розробка вебсервісу «Картки для запам’ятовування» з використанням Python та MySQL

затверджена наказом по коледжу № 09.2-20 від 04.03.22 р.

2. Термін здачі студентом закінченої роботи: __.__.__ р.

3. Перелік питань, що підлягають розробці:

1. Розлянути предметну область
2. Розглянути існуючі аналоги та сформувати задачі проекту
3. Визначити та оглянути стек технологій
4. Визначити структуру проекту
5. Розробити програму
6. Провести тестування
7. Оформити інструкції до розгортки
8. Оформити результати роботи у вигляді звіту

4. Календарний графік підготовки випускної роботи молодшого спеціаліста

№ етапу	Зміст	Терміни виконання	Готовність по графіку (%), підпис керівника	Підпис керівника про повну готовність етапу, дата
1	Формулювання (корегування) теми випускної роботи молодшого спеціаліста та збір практичного матеріалу за темою випускної роботи	17.01.22-26.02.22		
2	І атестація І розділ випускної роботи молодшого спеціаліста	28.03.22-02.04.22		
3	ІІ атестація ІІ розділ випускної роботи молодшого спеціаліста	10.05.22-14.05.22		
4	ІІІ атестація ІІІ розділ випускної роботи молодшого спеціаліста, висновки та рекомендації, додатки, реферат	30.05.22-04.06.22		
5	Перевірка випускноїм роботи програмою «Антиплагіат»	30.05.22-18.06.22		
6	Доопрацювання випускної роботи молодшого спеціаліста, підготовка презентації, отримання відгуку керівника і рецензії	06.06.22-11.06.22		
7	Попередній захист випускної роботи молодшого	14.06.22-18.06.22		
8	Подача випускної роботи молодшого спеціаліста на кафедру	за 3 дні до захисту		
9	Захист випускної роботи молодшого спеціаліста	20.06.22-25.06.22		

Керівник _____ К.В. Дереза

(підпис) (прізвище та ініціали)

«____» _____ 2022 р.

Завдання отримав до виконання _____ О.С. Бірюков

(підпис студента) (прізвище та ініціали)

«____» _____ 2022 р.

РЕФЕРАТ

Випускна робота молодшого спеціаліста містить * сторінок, 68 рисунків, 13 лістингів, 47 бібліографічних посилань, один додаток.

Метою роботи є розробка вебсервісу для роботи з картками для запам’ятовування за допомогою сучасних технологій розробки вебсервісів.

Об’єктом дослідження є застосування сучасних веб та клієнт-серверних технологій для створення вебсервісів.

Предметом дослідження є вебсервіс для створення, редагування, переглядання та вивчення карток для запам’ятовування.

Здійснено детальний огляд предметної області та популярних аналогів. Виявлено, що тема карток для запам’ятовування є актуальною, а розробка вебсервісу для роботи з картками для запам’ятовування є доцільною. Проект реалізовано за допомотою таких засобів, як Python, MySQL, HTML, CSS, JavaScript. Здійснено проектування моделі предметної області, програмування сутностей та алгоритмів за принципами об’єктно-орієнтованого програмування та побудови клієнт-серверної архітектури за сучасними стандартами.

Розроблений програмний продукт є легким у використанні, має привабливий зовнішній дизайн та інтуїтивно зрозумілий дизайн інтерфейсу. Вебсервіс дозволяє легко та швидко створювати картки з тектом та зображеннями, редагувати їх, переглядати та вивчати методом інтервального повторення.

ВЕБСЕРВІС, МНЕМОНІКА, ФЛЕШКАРТКИ, КЛІЄНТ, СЕРВЕР, PYTHON, MYSQL, HTML, CSS, JAVASCRIPT

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

Скорочення	Повна назва	Пояснення/переклад
БД	База Даних	Сукупність організованих даних
ООП	Об’єктно-Орієнтоване Програмування	Парадигма програмування, основана на взаємодії об’єктів
ОС	Операційна Система	Комплекс програм управління апаратним забезпеченням
ПЗ	Програмне Забезпечення	Комплекс програм та компонентів для їх роботи
СУБД	Система Управління Базами Даних	Комплекс взаємопов’язаних даних та програм для доступу та управління ними
API	Application Programming Interface	Програмний інтерфейс засосунку
apt	Advanced Package Tool	Програма для для роботи з програмними пакетами Debian
CSS	Cascading Style Sheets	Каскадна таблиця стилів
DB	Data Base	База даних
env	Environment	Середовище
HTML	HyperText Markup Language	Мова розмітки гіпертексту
IOS	IPhone Operating System	Операційна система IPhone
IP	Internet Protocol	Протокол інтернету
JS	JavaScript	Джаваскрипт, мова

		програмування
JSON	JavaScript Object Notation	Нотація об'єктів JavaScript
OS	Operating System	Операційна система
pic	Picture	Зображення
pip	Package Installer for Python	Інсталятор пакетів Python
py	Python	Пайтон, мова прогрмування
REST	Representational State Transfer	Передача самоописуваного стану, принцип побудови API
SQL	Structured Query Language	Мова структурованих запитів
sudo	Super User Do	Дія від імені адміністратора Unix-подібної системи
UML	Unified Modeling Language	Уніфікована мова моделювання
URL	Uniform Resource Locator	Уніфікований вказіваник ресурсу
UUID	Universal Unique Identifier	Універсальний унікальний ідентифікатор

ВСТУП

У повсякденному житті постійно необхідно щось запам’ятовувати. А в сучасному світі для цього залишається все менше часу при зростаючому з кожним днем обсязі інформації. Тому для швидшого, ефективнішого та цікавішого запам’ятовування інформації були розроблені спеціальні техніки. Однією з них є техніка «картки для запам’ятовування», або «флешкартки». Вивчення здійснюється шляхом асоціативного запам’ятовування зв’язаної між собою інфомації на сторонах картки [1].

За допомогою цієї техніки можна ефективно запам’ятовувати теоретичну інформацію для будь-якої сфери навчання і діяльності. Вивчення більшості дисциплін включає в себе практичну частину навчання, на яку варто приділяти більше уваги та направляти більше сил. Саме тому процес запам’ятовування теоретичної інформації має бути якомога легшим та швидким.

Для цього, картки для запам’ятовування часто використовують разом з технікою «інтервального повторення» [2], яку можна реалізувати програмним алгоритмом. У такий спосіб програма бере на себе більшу частину трудомісткості процесу запам’ятовування теоретичної інформації, дозволяючи користувачу економити час, увагу та сили на практичну частину вивчення дисципліни.

Із вищесказаного випливає, що програма для роботи з картками для запам’ятовування може значно полегшити процес навчання, спрощуючи процес запам’ятовування теоретчної інформації, завдяки використанню спеціальних алгоритмів для ефективного вивчення карток.

Метою роботи буде створення програми для роботи з картками для запам’ятовування та їх ефективного вивчення.

РОЗДІЛ 1
ОГЛАД ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Актуальність теми

На сьогоднішній день навчання є одним із основних аспектів життя людини. Велика частина процесу навчання полягає в опрацюванні та запам’ятовуванні інформації, якої з кожним днем з’являється тільки більше. Відповідно з кожним днем все більше часу необхідно виділяти для її запам’ятовування.

Саме для покращення досвіду запам’ятовування інформації були створені спеціальні техніки, а потім і комп’ютерні програми для автоматизації цих технік. Однією з таких технік є «картки для запам’ятовування», або «флешкартки» [1]. Такі картки почали використовуватися приблизно у 19 ст. Один із перших наборів, створений брианською вчителькою Фавелл Лі Мортімер, був створений для вивчення англійської мови [3]. З того часу флешкартки активно використовуються, особливо з розповсюдженням швидкісного інтернету по всьому світу. На момент початку 2022 року тема є актуальною. Дані Google Trends щодо популярності теми «Навчальні картки» з 2004 року по 2022 рік [4] наведено у рисунку 1.1.

Рисунок 1.1 – Динаміка попутярністі теми «Навчальні картки» за даними Google Trends з 2004 року по 2022 рік

Хоча картки для запам’ятовування починали використовувати для вивчення мов, з їх допомогою можна запам’ятовувати теоретичну інформацію з будь-якої сфери навчання і діяльності. Наприкад, окрім мов, за допомогою флешкарток вивчають медицину, фармакологію, хімію, біологію, літературу, математику, історію, фізику, географію та інші дисципліни [5-6].

1.2 Мнемоніка

1.2.1 Визначення

Мнемоніка – комплекс технік, направлених на полегшення процесу запам’ятовування та розширення обсягу пам’яті через створення штучних асоціацій та надання візуального, аудіального або кінестетичного уявлення абстрактним об’єктам, поняттям та фактам, а також зв’язування таких асоціацій із вже відомими фактами та між собою [7].

За сучасним трактуванням до поняття «мнемоніка» входять усі техніки та методи запам’ятовування інформації.

1.2.2 Прийоми мнемоніки

До основних прийомів мнемоніки [7] входять:

- асоціація зі звичними та вже відомими образами;
- формування осмислених фраз з початкових букв;
- знаходження у відомій мові слів, співзвучних до іноземних, на мові, що вивчається;
- ритмування та римування;

- просторова уява методом Цицерона;
- розвиток зорової пам'яті методом Айвазовського;
- запам'ятовування чисел методами фонетичної системи, Person-Action-Object, закономірностей та знайомих чисел;
- парне зв’язування образів;
- символізація;
- спеціальні фрази та приказки;
- прийом «флешкартки», або «картки для запам’ятовування».

1.2.3 Практичний прийом «флешкартки»

Одним з найбільш ефективних методів запам'ятовування є прийом «флешкарток», або «карток для запам’ятовування».

Флешкартки представляють собою фізичні або віртуальні двосторонні (зазвичай) картки, зі зв’язаним між собою вмістом на різних сторонах, часто у форматі «питання-відповідь» [7].

Обмежуючись лише форматом (фізична картка або віртуальна), вміст на сторонах картки може бути будь-яким: текст, зображення, аудіо, відео або будь-яка їх комбінація.

Користувач картки має, дивлячись або слухаючи вміст передньої сторони, пригадати вміст зворотної, а потім перегорнути картку і перевірити результат.

Группа флешкарток однієї категорії називається «колодою».

Основні принципи прийому: ефект тестування (повторення вже відомого матеріалу ефективніше за пасивне вивчення) [8] та техніка інтервального повторення [2].

1.2.4 Техніка інтервального повторення

Техніка інтервального повторення використовує психологічний ефект інтервалів, за яким навчання є більш ефективним, якщо сесії навчання відбуваються з інтервалами часу [9].

Техніка частіше всього використовується у зв’язці з прийомом флешкарток.

У зв’язці з флешкартками, принцип роботи такий: нові та більш складні картки з однієї колоди слід повторювати частіше, а більш старі та менш складні – рідше [2].

Після повторення флешкартки користувачу слід оцінити, як добре він пам’ятає її (наприклад, за п’ятибальною системою), і на основі оцінки перемістити картку «вглиб» колоди. Відповідно, чим краще користувач пам'ятає флешкартку, тим глибше у колоду її слід перемістити.

На цьому принципі основані програмні алгоритми інтервального повторення, що і використовуються у прикладному ПЗ.

1.3 Розгляд аналогів

У цьому підрозділі розглядаються аналоги до проекту, що розробляється [10].

1.3.1 Anki Flashcards

Найпопулярніший застосунок для флешкарток з інтервальним повторенням. Простий, але має широку базу користувацьких плагінів під вивчення найрізноманітніших дисциплін. Приклад інтерфейсу Anki Flashcards наведено на рисунку 1.2 [11].

Рисунок 1.2 – Приклад інтерфейсу Anki Flashcards

Переваги:

- мінімалістичний дизайн;
- широке онлайн-співтовариство;
- система інтервального повторення;
- безкоштовність (крім версії для IOS);
- наявність версій для великої кількості ОС;
- наявність браузерної веб-версії;
- безліч сторін карток;
- підтримка зображень та аудіо;
- підтримка додаткових форматів;
- синхронізація між пристроями;
- підтримка користувацьких плагінів;
- робота без підключення до мережі.

Недліки:

- висока ціна версії для IOS;
- система оцінки потребує від користувача періоду звикання.

1.3.2 Brainscape

Застосунок для флешкарток з інтервальним повторенням, має просунуті засоби вістежування прогресу навчання. Приклад інтерфейсу Brainscape наведено на рисунку 1.3 [11].

Рисунок 1.3 – Приклад інтерфейсу Brainscape

Переваги:

- відстежування прогресу;
- привабливий дизайн;
- торговий майданчик;
- система інтервального повторення;
- наявність версій для великої кількості ОС;
- наявність браузерної веб-версії;
- підтримка зображень та аудіо;
- підтримка додаткових форматів;
- синхронізація між пристроями;
- підтримка користувацьких плагінів;
- робота без підключення до мережі.

Недоліки:

- використання зображень, аудіо та карток інших користувачів потребує преміум-підписки;
- система оцінки потребує від користувача періоду звикання.

1.3.3 Mnemosyne Project

Застосунок для флешкарток з інтервальним повторенням, направлений на компактність і простоту, при цьому має широкі можливості до кастомізації. Приклад інтерфейсу Mnemosyne Project наведено на рисунку 1.4 [15].

Рисунок 1.4 – Приклад інтерфейсу Mnemosyne Project

Переваги:

- компактний дизайн;
- графічна статистика;
- фільтр карток та груп карток до тренування;
- до 3 сторін карток;
- підтримка відображення формул Latex;
- підтримка скриптів;
- система інтервального повторення;
- безкоштовність;
- наявність версій для декількох ОС;
- наявність браузерної веб-версії;
- підтримка зображень та аудіо;
- підтримка додаткових форматів;
- синхронізація між пристроями;
- підтримка користувацьких плагінів;
- робота без підключення до мережі.

Недоліки:

- вузьке онлайн-співтовариство;
- система оцінки потребує від користувача періоду звикання.

1.3.4 Quizlet

Фукціонально спрощений застосунок, що симулює роботу з фізичними картками. Приклад інтерфейсу Quizlet наведено на рисунку 1.5 [11].

Рисунок 1.5 – Приклад інтерфейсу Quizlet

Переваги:

- привабливий дизайн;
- відстежування прогресу;
- синтезатор мови на 18 мовах;
- широкі можливості для використання у навчальних закладах;
- широкі можливості для форматування тексту;
- вбудовані навчальні матеріали;
- наявність версій для Android та IOS;
- наявність браузерної веб-версії;
- підтримка зображень та аудіо;
- синхронізація між пристроями;

Недоліки:

- немає версій для настільних ОС;
- немає підтримки користувацьких плаганів;
- немає системи інтервального повторення;
- спрощений функціонал;
- робота без підключення до мережі потребує преміум-підписки;

1.4 Постановка задачі

Проаналізувавши переваги та недоліки аналогів, був сформований список задач проекту:

- розробити систему користувачів, їх реєстрації та аутентифікації;
- розробити хмарне сховище для особистих даних користувача, його карток та колод;

- розробити алгоритм інтервального повторення для вивчення та повторення флешкарток;
- розробити редактор для створення та редагування флешкарток;
- флешкартки повинні мати 2 сторони, наповнення може включати зображення, текст або їх комбінацію;
- розробити простий, інтуїтивний та привабливий дизайн інтерфейсу;
- розробити програму у вигляді вебсервісу для охоплення максимальної кількості ОС та пристроїв.

1.5 Висновки за розділом

У даному розділі було проаналізовано актуальність теми: визначено, що тема є актуальною, а розробка програми за цією темою є доцільною.

Було оглянуто предметну область: основні визначення, поняття, та техніки мнемоніки, зокрема зв’язку технік «флешкарток» та «інтервального повторення».

Були розглянуті існуючі аналоги, виділено їхні переваги та недоліки.

На основі аналізу аналогів сформовано список задач проекту.

РОЗДІЛ 2
ВИБІР ТА ОБГРУНТУВАННЯ ПРОГРАМНО-АПАРАТНИХ РІШЕНЬ

2.1 Серверна частина

2.1.1 Операційна система Ubuntu

Ubuntu – операційна система на основі ядра Linux-Debian, розроблена компанією Canonical [18].

За даними ресурсу Statista (рис.2.1) в період з 2018 по 2021 роки двома найпопулярнішими ОС для розробки ПЗ є Microsoft Windows та та Unix/Linux [19].

Рисунок 2.1 – статистика популярності ОС для розробки ПЗ

Так як Microsoft Windows є пропрієтарним ПЗ [20], а друга по популярності група ОС Unix/Linux має відкритий вихідний код [18], ОС з цієї групи доцільно вибрати для даного проекту завдяки безкоштовності використання.

За даними ресурсу Truelist (рис.2.2) ОС Ubuntu має майбільший відсоток ринку серед ОС на базі ядра Linux [21], тому її доцільно вибрати для даного проекту.

Рисунок 2.2 – статистика популярності ОС на базі ядра Linux

Для використання доступно дві редакції Ubuntu: Ubuntu Desktop та Ubuntu Server [18]. Ubuntu Desktop має сучасний графічний інтерфейс настільної ОС (рис.2.3), на відміну від повністю командного інтерфейсу Ubuntu Server, це дозволяє комфортно та ефективно розробляти проект, використовуючи графічні інструменти розробки та тестування.

Рисунок 2.3 – Інтерфейс Ubuntu Desktop

Окрім зручності процесу розробки, стандартний набір ПЗ, що встановлюється разом з Ubuntu Desktop, є мінімальним, тому ця редакція Ubuntu також може використовуватися у якості серверної ОС.

Цей вибір дає ширші, звучніші та наочніші можливості до моніторингу ресурсів, тестування та налагодження серверних застосунків. Для прикладу, стандартна утіліта Ubuntu Desktop, «System Monitor», дає можливість графічно та у реальному часі відстежувати використання ресурсів машини (рис.2.4).

Рисунок 2.4 – Інтерфейс утіліти System Monitor

З інших особливостей Ubuntu можна виділии [18]:

- ядро Linux-Debian, яке підтримується великою частиною виробників ПЗ, у тому числі ПЗ для розробки;
- підтримка роботи у режимі віртуальної машини;
- потужний командний рядок Linux Bash Shell;
- гнучке налаштування користувачів та прав доступу для адміністрування.

Отже, для розробки та у якості серверної ОС для проекту доцільно використовувати Ubuntu Desktop.

2.1.2 Мова програмування Python

Python – високорівнева, інтерпретована, інтерактивна, об'єктно-орієнтована скриптова мова програмування [23].

За даними ресурсу Tutorialspoint (рис.2.5) за квітень 2022 найпопулярнішою мовою програмування є Python [24].

Рисунок 2.5 – статистика популярності мов програмування

Особливості Python [23]:

- динамічна типізація і керування пам’яттю;
- автоматичний збірщик сміття;
- простий та читабельний синтаксис;
- інтерпретатор;
- підтримка ООП;
- пітримка багатьох платформ;
- підтримка багатох СУБД;
- безліч сторонніх модулів, у тому числі для розробки вебсервісів.

Отже, для розробки серверної частини сервісу доцільно використовувати мову програмування Python.

Приклад синаксису найпростішої програми на Python наведений у лістингу 2.1.

Лістинг 2.1 – найпростіша програма на Python

```
while True:
    age = int(input(“Your age: ”))
    if age < 18:
        print(“Not allowed”)
    elif age >= 18:
        print(“Welcome”)
        break
```

У ОС Ubuntu інтерпретатор Python входить до стандартного набору програм. Працювати з ним можна у Bash-терміналі Ubuntu у інтерактивному режимі, рядок за рядком, або передавати файл з розширенням «.py» у якості агрумента. Використання обох методів показано на рисунку 2.6.

Рисунок 2.6 – різні способи роботи з інтерпретатором Python у Ubuntu

2.1.3 СУБД та база даних MySQL

MySQL – система управління базами даних, розроблена корпорацією Oracle [26].

За даними ресурсу Statista на момент січня 2022 року (рис. 2.7) двома найпопулярнішими СУБД є Oracle та MySQL [27].

Рисунок 2.7 – статистика популярності СУБД

Так як СУБД Oracle є пропрієтарним ПЗ [28], а друга по популярності MySQL має відкритий вихідний код [26], її доцільніше вибрати для даного проекту завдяки безкоштовності використання.

Основними особливостями MySQL є [26]:

- відкритий вихідний код;
- бази даних є реляційними;
- швидкість, надійність, масштабованість, безпечність та легкість у викорстанні;
- система призначена для архітектури «клієнт-сервер»;
- широка підтримка серед засобів розробки та стороннього ПЗ.

Робота з MySQL підтримується у Python завдяки модулю mysql-connector [29].

Отже, у якості СУБД та бази даних даного проекту доцільно вибрати MySQL.

Робота з MySQL відбувається через запити стандарту мови SQL. За замовчуванням запити можна відправляти вручну з командного рядка (Рис. 2.8).

Рисунок 2.8 – Робота з MySQL у режимі командного рядка

2.1.4 Фреймворк FastAPI

Клієнтська частина має отримувати дані з серверу. Для цього існує технологія REST API – комплекс визначень та протоколів для створення програмних інтерфейсів вебсервісів за архітектурним стилем «REST». Працює технологія по принципу «запит-відповідь». Наприклад, щоб отримати дані про погоду, треба відправити запит з даними ідентифікації

регіону, і отримати дані прогнозу, які можна обробити і вивести для користувача [30].

Одним з найпопулярніших фрейворків Python для створення REST API є FastAPI [31].

З особливостей [32]:

- швидкість роботи;
- створений для високої швидкості написання коду;
- створений для мінімізації помилок програміста;
- мінімум коду, у тому числі дублюючого;
- легкий у використанні та вивченні;
- код одразу готовий до роботи;
- направлений на дотримання стандартів.

Отже, фреймворк FastAPI для Python дозволяє швидко та з мінімальним об’ємом коду створити REST API для серверної частини вебсервісу, тому його доцільно використати для проекту, що розробляється.

2.2 Клієнтська частина

2.2.1 Мова розмітки HTML та каскадна таблиця стилів CSS

HTML (HyperText Markup Language) – стендартна мова розмітки гіпертексту для документів, створених для відображення у веб браузері [33].

Структура HTML-документа складається з HTML-елементів. Вони можуть бути, наприклад: звичаним текстом, заголовком, списком,

зображенням, відео тощо. Приклад коду HTML-документу та його відображення у веб браузері покзано на рисунку 2.9.

Рисунок 2.9 – приклад відображення простого HTML-документу

Стандартно, для надання елементам та сторінці налаштувань відображення, використовують каскадну таблицу стилів на мові CSS. Налаштування можуть бути, наприклад: форматуванням тексту, його кольором, фоновим зображенням, анімацією, реакцією на дію користувача тощо [34].

CSS можна використовувати у HTML-документі, приписуючи налаштування окремим елментам, або в окремому доменті, призначаючи так звані «стилі» окремим елементам та групам елементів. Наприклад, зміни відображення документу з рисунку 2.9 після присвоєння стилів CSS (всередині HTML-документу) покзано на рисунку 2.10.

Рисунок 2.10 – приклад присвоєння стилів CSS

Так як клієнська частина вебсервісу, що розробляється – вебсайт, а HTML та CSS є стандартними мовами для створення вебсторінок, їх необхідно використати для клієнтської частини проекту, що розробляється.

2.2.2 Мова програмування JavaScript

JavaScript – високорівнева мова програмування, що є однією із основних технологій створення вебсторінок, разом з HTML та CSS. Близько 97% вебсайтів використовують JavaScript для програмування поведінки клієнтської частини вебсторінок та більшість веббраузерів на більшості ОС його підтримують. Є простим завдяки динамічній типізації, автоматичному управлінню пам’яттю та сбірщику сміття, проте підтримує різні парадигми програмування, у тому числі ООП. Також JavaScript має API для роботи з текстом, датами, стандартними струкурами даних, та елементами на HTML-сторінці [35].

Отже із вищесказаного, а також із того, що клієнська частина вебсервісу, що розробляється – вебсайт, мову програмування JavaScript доцільно використати для клієнтської частини проекту, що розробляється.

Простий приклад використання JavaScript на вебсторінці показано на рисунку 2.11.

Рисунок 2.11 – простий приклад використання JavaScript на вебсторінці

2.3 Середа розробки

2.3.1 Менеджер віртуальних середовищ Conda

Conda – менеджер пакетів, залежностей та віртуальних середовищ для різних мов програмування, у том числі Python [36].

Він дозволяє створювати та переключатися між віртуальними середовищами Python, у яких, незалежно один від одного, можна встановлювати різні версії Python та його модулів.

За допомогою цього можна сторювати ізольовані середовища для розробки та запуску різних застосунків та різних їх версій.

Приклад переключення між двома середовищами, «oldpython» та «newpython» з різними версіями Python наведео у рисунку 2.12.

Рисунок 2.12 – переключення між середовищами у Conda

Conda входить до Anaconda [36] – менеджеру пакетів та дистрибутиву Python [38].

Отже, для використання окремих ізольованих віртуальних середовищ Python для розробки, тестування та використання проекту, що розробляється, доцільно використати Conda у пакеті Anaconda.

2.3.2 Інтегроване середовище розробки PyCharm

PyCharm – сучасне інтегроване середовище розробки, направлене на програмування на Python [39].

З загальних функцій [40]:

- підтримка розробки на HTML, CSS та JavaScript;
- розумна корекція коду;
- розумна навігація по коду;
- розумний рефакторинг коду;
- інструменти налагодження, тесування та профілювання;
- система контролю версій;
- інструменти розгортання;
- дистанційна розробка;

- інструменти роботи з базами даних;
- кастомізація інтерфейсу;
- підтримка плагінів;
- кросплатформовість.

З функцій, призначених для розробки на Python [40]:

- підтримка фреймворків для розробки REST API;
- підтримка наукових бібліотек Python (рис 2.13) [41];
- вбудована інтерактивна консоль Python;
- інтеграція з Conda.

Рисунок 2.13 – наукові бібліотеки Python у PyCharm

Отже, інтегровану середу розробки PyCharm доцільно використати для розробки як серверної, так і клієнтської частини проекту.

2.3.3 Середовище розробки API Postman

Postman – середовище тестування, будування та використання API [42].

Postman дозволяє [43]:

- зберігати та каталогізувати створені документації API, запити та їх відповіді для тестування, метрики тощо у хмарному сховищі;
- розробляти API на різних стадіях: дизайну, тестування та документації;
- отримувати попередження про проблеми безпеки, звіти та повідомлення про помилки із запитами;
- створювати публічні робочі простори.

Робота із запитами REST API показана на рисунку 2.14.

Рисунок 2.14 – робота із запитом REST API у Postman

Із вищесказаного, випливає, що Postman доцільно використати для розробки, тестування та документації REST API проекту, що розробляється.

2.5 Висновки за розділом

У даному розділі було визначено та оглянуто стек технологій для проекту, що розробляється.

Визначено, що:

- для розробки та у якості серверної ОС буде використана Ubuntu Desktop;
- для розробки серверної частини сервісу буде використана мова програмування Python;
- у якості СУБД та бази даних буде викорстана MySQL;
- REST API проекту буде побудованио за допомогою фреймворку FastAPI;
- для розробки клієнтської частини будуть використані HTML, CSS та мова програмування JavaScript;
- для використання окремих ізольованих віртуальних середовищ Python для розробки, тестування та використання проекту, що розробляється, буде використано Conda у пакеті Anaconda;

- інтегровану середу розробки PyCharm буде використано для розробки як серверної, так і клієнтської частини проекту;
- середовище розробки API Postman буде використано для розробки, тестування та документації REST API проекту.

РОЗДІЛ 3
ПРОГРАМНИЙ МОДУЛЬ

3.1 Проектно-архітектурні застосунки UML

UML діаграма прецедентів показана на рисунку 3.1.

Рисунок 3.1 – UML діаграма прецедентів

UML діаграма розгортки вебсервісу показана на рисунку 3.2.

Рисунок 3.2 – UML діаграма розгортки вебсервісу

UML діаграма класів клієнтської частини показана на рисунку 3.3.

Рисунок 3.3 – UML діаграма класів клієнтської частини

UML діаграма класів серверної частини показана на рисунку 3.4.

Рисунок 3.4 – UML діаграма серверної частини

3.2 Структура проекту

3.2.1 Структура клієнтської частини

Файли клієнстської частини знаходяться у межах однієї діректорії. Схема файлової структури показана на рисунку 3.5.

Рисунок 3.5 – Схема файлової структури клієнтської частини

Вхідний документ – «index.html», що містить сторінку акторизації. Всі інші html-документи знаходяться у папці «views».

У папці «app» знаходяться документи JavaScript-коду.

У файлі «authentication_handler.js» визначено клас «Authentication_handler» з методами:

- «log_into_current_user» (якщо у браузері є активна сесія, авторизується під цим користувачем, не приймає параметрів, не повертає значень);
- «login» (викликає метод авторизації класу «request_handler», приймає об’єкт форми з даними, повертає результат авторизації);
- «signup» (викликає метод реєстрації класу «request_handler», приймає об’єкт форми з даними, повертає результат реєстрації);
- «log_out» (завершує активну сесію, не приймає параметрів, не повертає значень).

У файлі «request_handler.js» визначено клас «Request_handler» з методами:

- «login» (відправляє дані акторизації до API і авторизує користувача при позитивній відповіді API, приймає об’єкт форми з даними, повертає результат авторизації);
- «signup» (відправляє дані реєстрації до API, відкриває сторінку авторизації при позитивній відповіді API, приймає об’єкт форми з даними, повертає результат реєстрації);

- «copy_example_deck» (відправляє ім’я нового користувача до API для копіювання йому тестової колоди, приймає ім’я поточного користувача, не повертає значень);
- «add_card» (відправляє дані нової картики до API, приймає 2 форми з даними нової картки (передньої та зворотної сторін), не повертає значень);
- «post_deck_pic» (відправляє зображення колоди до API, приймає об’єкт елементу завантаження файлу та ім’я у вигляді UUID v4, не повертає значень);
- «get_all_cards» (віправляє запит на отримання даних карток користувача до API, потім викликає метод для відображення на сторінці, не приймає параметрів, не повертає значень);
- «get_profile_pic» (віправляє запит на зображення профіля користувача до API, потім викликає метод для відображення на сторінці, приймає необов’язковий параметр для відображення зображення у великому масштабі, не повертає значень);
- «delete_card» (відправляє дані для видалення картки до API, приймає UUID картки, не повертає значень);
- «learn_deck_redirect» (відправляє дані для перевірки чи пуста колода до API та переходить на сторінку вивчення колоди при позитивній відповіді API, інакше виводить повідомлення, приймає UUID колоди, не повертає значень);
- «add_card_redirect» (визначає поточну колоду сесії у браузері, оновлює значення останної активності колоди і переходить на сторінку створення картки, приймає UUID колоди, не повертає значень);
- «edit_deck_redirect» (визначає поточну колоду сесії у браузері, оновлює значення останної активності колоди і переходить на сторінку редагування картки, приймає UUID колоди, не повертає значень);

- «update_deck_activity» (відправляє дані для оновлення значення останньої активності колоди до API, приймає UUID колоди, не повертає значень);
- «delete_deck» (відправляє дані для видалення колоди до API, не приймає параметрів, не повертає значень);
- «delete_user» (відправляє дані для видалення користувача до API, не приймає параметрів, не повертає значень);
- «edit_deck» (відправляє дані для редагування колоди до API, приймає об’єкт форми з даними та необов’язковий параметр ідентифікатору елементу завантазення файлу, не повертає значень);
- «edit_user» (відправляє дані для редагування користувача до API, приймає об’єкт форми з даними та необов’язковий параметр ідентифікатору елементу завантазення файлу, не повертає значень);
- «add_deck» (відправляє дані для створення колоди до API, приймає об’єкт форми з даними та необов’язковий параметр ідентифікатору елементу завантазення файлу, не повертає значень);
- «get_all_decks» (відправляє запит на отримання даних до API, потім викликає метод для відображення на сторінці, не приймає параметрів, не повертає значень);
- «get_deck_cover» (віправляє запит на отримання зображення колоди до API, потім викликає метод для відображення на сторінці, приймає необов’язковий параметр для відображення зображення у великому масштабі, не повертає значень);
- «print_learn_pic» (відправляє дані для отримання зображення з картки до API, потім викликає метод для відображення на сторінці, приймає об’єкт з UUID картки та розміщення зображення, не повертає значень);
- «move_card» (відправляє дані для зміни черги картки у колоді до API, приймає об’єкт з UUID картки та оцінку, не повертає значень);

- «print_full_card_pic» (відправляє дані для отримання зображення з картки до API, потім викликає метод для відображення на сторінці, приймає об’єкт з UUID картки та розміщення зображення, не повертає значень).

У файлі «display_handler.js» визначено клас «Display_handler» з методами:

- «print_small_decks» (якщо колода пуста, виводить повідомлення, інакше – відображає все колоди, приймає об’єкт з всіма картками колоди, не повертає значень);
- «print_full_card» (якщо колода пуста, виводить повідомлення, інакше – відображає дві сторони кожної картки у колоді, приймає об’єкт з всіма картками колоди, не повертає значень);
- «print_card_side» (відправляє дані для отримання даних першої картки у черзі, в залежності від параметра відображає передню або зворотну сторону, приймає необов’язковий параметр відображення зворотної сторони, не повертає значень).

У файлі «authentication_handler.js» визначено клас «Authentication_handler» з методами:

- «log_into_current_user» (якщо у браузері є активна сесія, авторизується під цим користувачем, не приймає параметрів, не повертає значень);
- «login» (викликає метод авторизації, приймає об’єкт форми з даними, повертає результат авторизації);
- «signup» (викликає метод реєстрації, приймає об’єкт форми з даними, повертає результат реєстрації);
- «log_out» (завершає активну сесію у браузері та переходить на сторінку авторизації, не приймає параметрів, не повертає значень).

У файлі «env.js» визначено клас «Display_handler» з полями:

- «domain» (адреса та порт API);

- поля з закінченням на «_endpoint» (адреси методів API).

Методи класу «Display_handler»:

- «getTitle» (повертає заголовок сторінки, не приймає параметрів, повертає заголовок сторінки);
- «getFullUrl» (повертає повну адресу методу API, приймає адресу методу API, повертає повну адресу методу API).

У файлі «cookie.js» визначено клас «Cookie» з методами:

- «getCookie» (повертає значення cookie по назві, приймає ім’я cookie, повертає значення cookie);
- «setCookie» (створює cookie, приймає ім’я та значення cookie, не повертає значень);
- «deleteCookie» (видаляє cookie по назві, приймає ім’я cookie, не повертає значень).

У файлі «utilities.js» визначено клас «Utilities» з методами:

- «UUID4» (повертає сгенерований UUID v4, не приймає параметрів, повертає сгенерований UUID v4);
- «start_connection_checker» (при виклику починає робити запити до API для перевірки з’єднання з сервером, якщо відповідь негативна – відображає повідомлення, що перекриває всю сторінку, не приймає аргументів, не повертає значень).

У папці «assets» знаходяться необхідні зображення вебсайту.

У папці «styles» знаходяться каскадні таблиці стилів вебсайту.

3.2.2 Структура серверної частини

Файли серверної частини знаходяться в межах однієї директорії. Схема файлової структури показана на рисунку 3.6.

Рисунок 3.6 – Схема файлової структури серверної частини

Файл запуску серверу API – «main.py», у ньому запускається сервер та визначається скрипт-роутер «api.py» у папці «api», що визначає адреси методів API.

У файлі «flashcards.py» визначаються методи API:

- «/login» (метод аутентифікції, тип: post, приймає json-дані, повертає результат аутентифікації);
- «/signup» (метод реєстрації, тип: post, приймає json-дані, повертає результат реєстрації);
- «/get_profile_pic» (метод отримання зображення профілю, тип: post, приймає json-дані, повертає файл зображення);
- «/post_profile_pic» (метод завантаення зображення профілю, тип: post, приймає файл зображення, повертає булеве «True»);
- «/delete_user» (метод видалення профілю, тип: post, приймає json-дані, приймає json-дані, повертає результат видалення);
- «/add_deck» (метод додання колоди, тип: post, приймає json-дані, повертає булеве «True»);
- «/post_deck_cover» (метод завантаення зображення колоди, тип: post, приймає файл зображення, повертає результат завантаження);
- «/get_all_decks» (метод отримання всіх колод, тип: post, приймає json-дані, повертає словник всіх колод);
- «/get_deck_cover» (метод отримання зображення колоди, тип: post, приймає json-дані, повертає булеве «True»);
- «/get_queue» (метод отримання останнього номеру черги, тип: post, приймає json-дані, повертає номер черги);
- «/add_card» (метод додання картки, тип: post, приймає json-дані, повертає результат додання);

- «/edit_deck» (метод редагування колоди, тип: post, приймає json-дані, повертає результат редагування);
- «/edit_user» (метод редагування профілю, тип: post, приймає json-дані, повертає результат редагування);
- «/post_deck_pic» (метод завантаження зображення для картки, тип: post, приймає файл зображення, повертає булеве «True»);
- «/get_all_cards» (метод отримання всіх карток, тип: post, приймає json-дані, повертає словник карток);
- «/get_card_pic» (метод отримання зображення для картки, тип: post, приймає json-дані, повертає файл зображення);
- «/delete_card» (метод видалення картки, тип: post, приймає json-дані, повертає результат видалення);
- «/delete_deck» (метод аутентифікції, тип: post, приймає json-дані, повертає результат аутентифікації);
- «/update_deck_activity» (метод оновлення часу останньої активності колоди, тип: post, приймає json-дані, повертає булеве «True»);
- «/get_last_card» (метод отримання наступної картки в черзі, тип: post, приймає json-дані, повертає результат словник з даними картки);
- «/copy_example_deck» (метод копіювання тестової колоди новому користувачу, тип: post, приймає json-дані, повертає результат копіювання);
- «/move_card» (метод зміни номеру черги картки, тип: post, приймає json-дані, повертає результат оновлення);
- «/check_connection» (метод перевірки з’єднання з сервером, тип: post, приймає json-дані, повертає булеве «True»).

У файлі «logic_controller.py» у папці «internal» визначаються «Request_handler» з методами:

- «authenticate» (перевіряє наявність профіля та пароль, приймає словник з іменем та паролем, повертає результат аутентифікації);

- «get_profile_picture» (викликає метод для отримання шляху до зображення профілю з БД, приймає словник з іменем та паролем, повертає повний шлях до зображення);
- «add_user» (перевіряє зайнятість імені, наявність всіх полів та викликає метод запису профілю до БД, приймає словник з іменем, паролем, іменем зображення (опційно), не повертає значень);
- «copy_example_deck» (викликає метод додання тестової колоди до нового користувача у БД, приймає словник з іменем профілю, повертає булеве «True», або «False» при помилці);
- «delete_user» (викликає метод видалення профілю з БД, приймає словник з іменем профілю, повертає булеве «True», або «False» при помилці);
- «delete_deck» (викликає метод видалення колоди з БД та видаляє її директорію з серверу, приймає словник з UUID колоди, повертає булеве «True», або «False» при помилці);
- «update_deck_activity» (викликає метод оновлення часу останньої активності колоди у БД, приймає словник з UUID колоди, повертає булеве «True», або «False» при помилці);
- «add_deck» (викликає метод додання картки до БД та створює директорію для неї на сервері, приймає словник з даними нової колоди, повертає булеве «True», або «False» при помилці);
- «get_all_decks» (викликає метод отримання всіх колод користувача з БД, приймає словник з ім’ям користувача, повертає масив словників з даними колод, або «False» при помилці);
- «get_all_cards» (викликає метод отримання всіх карт колоди з БД, приймає словник з ім’ям користувача та ім’ям картки, повертає масив словників з даними карт, або «False» при помилці);

- «get_deck_cover» (викликає метод отримання пляху зображення колоди з БД, приймає словник з UUID колоди, повертає шлях, або «False» при помилці);
- «get_card_pic» (викликає методи отримання імені зображення та даних картки з БД та формує шлях до зображення з картки, приймає словник з UUID картки та розміщення зображення, повертає шлях, або «False» при помилці);
- «get_queue» (викликає метод отримання кількості карток у колоді з БД, приймає словник з UUID колоди, повертає число карток, або «False» при помилці);
- «get_last_card» (викликає метод отримання даних наступної картики з БД, приймає словник з UUID колоди, повертає словник з даними картки, або «False» при помилці);
- «add_card» (викликає метод додання картки до БД, приймає словник з UUID колоди, оригінальні та нові (попередньо сгенеровані у форматі UUID v4) назви зображень з карки та тексти, повертає булеве «True», або «False» при помилці);
- «delete_card» (викликає метод видалення картки з БД та методи видалення файлів зображень картки, приймає словник з UUID картки, повертає UUID видаленої картки, або «False» при помилці);
- «delete_file» (видаляє файл з серверу, приймає шлях до файлу, повертає булеве «True», або «False» при помилці);
- «get_new_queue» (обчислює новий номер черги для картки, приймає кількість карток та оцінку, повертає номер черги);
- «move_card» (викликає метод зміщення картки у черзі у БД, приймає словник з UUID колоди, ім’ям користувача та оцінкою, повертає булеве «True», або «False» при помилці);
- «edit_deck» (викликає метод оновлення даних колоди у БД, приймає словник з UUID колоди, нову назву колоди (опційно), ім’я зображення

(опційно), ім’ям користувача, повертає ім’я колоди, або «False» при помилці);

- «edit_user» (викликає метод оновлення даних користувача у БД, приймає словник з новим ім’ям (опційно), новим паролем (опційно), та назвою нового файлу зображення (опційно), і видаляє старе зображення, якщо завантажено нове, або вибрано стандартне, повертає булеве «True», або «False» при помилці).

У файлі «db_controller.py» у папці «internal» визначаються «Mysql_handler» з методами:

- «get_password» (робить запит на отримання паролю користувача, приймає ім’я користувача, повертає пароль);
- «get_profile_pic_path» (робить запит на отримання шляху до зображення профілю з БД, ім’я користувача, повертає шлях до зображення, або «False» при помилці);
- «does_user_exist» (робить запит до БД для визначення чи існує користувач, приймає ім’я користувача, повертає булеве «True», або «False» при помилці);
- «does_deck_exist» (робить запит до БД для визначення чи існує колода, приймає ім’я колоди та користувача, повертає булеве «True», або «False» при помилці);
- «add_user» (вставляє дані нового користувача до БД, приймає словник з оригінальним іменем, паролем, іменем зображення (опційно) та нове ім’я зображення у форматі UUID v4 (опційно), повертає булеве «True», або «False» при помилці);
- «copy_example_deck» (записує дані карток тестової колоди до БД, приймає ім’я користувача, повертає булеве «True», або «False» при помилці);
- «add_deck» (записує дані колоди до БД, приймає словник з даними нової колоди, повертає булеве «True», або «False» при помилці);

- «delete_user» (видаляє користувача, його колоди та картки з БД, приймає словник з іменем профілю, повертає булеве «True», або «False» при помилці);
- «get_all_decks» (робить запит до БД на всі колоди користувача, приймає ім’я користувача, повертає масив словників з даними колод, або «False» при помилці);
- «get_all_cards» (робить запит до БД на всі картки колоди, приймає словник з ім’ям користувача та ім’ям картки, повертає масив словників з даними карт, або «False» при помилці);
- «does_have_custom_cover» (робить запит до БД на наявність зображення профілю у користувача, приймає UUID колоди, повертає результат);
- «get_deck_by_uuid» (робить запит до БД на дані колоди, приймає UUID колоди, повертає словник з даними колоди);
- «get_last_card» (робить запить до БД для отримання даних наступної картики, приймає ім’я колоди та користувача, повертає словник з даними картки, або «False» при помилці);
- «get_card_by_uuid» (робить запит до БД на дані колоди, приймає UUID колоди, повертає словник з даними колоди);
- «get_cards_number» (робить запит до БД всі картки для визначення їх кількості, приймає ім’я колоди та користувача, повертає число карток, або «False» при помилці);
- «get_card_pic_name» (робить запит до БД дані картки та повертає необхідний вид (розташування) зображення, приймає UUID картки та розташування зображення, повертає ім’я зображення на сервері);
- «add_card» (додає дані картки до БД, приймає словник з UUID колоди, оригінальні та нові (попередньо сгенеровані у форматі UUID v4) назви зображень з карки та тексти, повертає словник нових імен зображень, або «False» при помилці);

- «get_deck_cover_new_name» (формує ім’я для зображення колоди, приймає оригінальне ім’я зображення та UUID v4 для назви (опційно), повертає ім’я для зображення);
- «delete_deck» (видаляє колоду та картки з неї з БД, приймає ім’я колоди та користувача, повертає булеве «True», або «False» при помилці);
- «delete_card» (видаляє картку з БД, приймає UUID картки, повертає булеве «True», або «False» при помилці);
- «update_deck_activity» (оновлює час останньої активності колоди у БД, приймає UUID колоди, повертає булеве «True», або «False» при помилці);
- «move_card» (оновлює у БД номер черги першої картки у черзі, та відповідно зміщує у черзі інші, приймає новий ноер черги, ім’я колоди та користувача, повертає булеве «True», або «False» при помилці);
- «edit_deck» (оновлює дані колоди у БД, приймає словник з UUID колоди, нову назву колоди (опційно), ім’я зображення (опційно), ім’ям користувача та старе ім’я колоди, повертає булеве «True», або «False» при помилці);
- «edit_user» (оновлює дані користувача у БД, приймає словник з новим ім’ям (опційно), новим паролем (опційно), та назвою нового файлу зображення (опційно), повертає словник з даними видаленого користувача, або «False» при помилці).

У файлі «.env» у кореневій папці API визначаються налаштування змінних середовища API:

- «PORT» (порт API);
- «MYSQL_HOST» (адреса MySQL);
- «MYSQL_USER» (ім’я користувача MySQL);
- «MYSQL_PASSWORD» (пароль MySQL);
- «MYSQL_DB» (база даних для API у MySQL);
- «PROJECT_PATH» (шлях до директорії API).

У файлі «config.py» у папці «core» підключаються налаштування змінних середовища з файлу «.env».

У папці «storage» зберігаються користувацькі файли для клієнтської частини. За шляхом «storage/profile_pictures» зберігаються зображення профілів, разом зі стандартним. За шляхом «storage/default» знаходяться стандартне зображення колоди та тестова колода. За шляхом «storage/data» зберігаються зображення з карток користувачів.

3.2.3 Структура бази даних

База даних складається з 3 таблиць:

- «users», що зверігає всіх користувачів;
- «decks», що зберігає всі колоди;
- «cards», що зберігає всі карти.

Поля «UUID» у всіх таблицях «users» – первинні ключі, вони зберігають унікальний ідентифікатор у форматі UUID версії 4, генерацію якого забезпечує API.

Поле «name» у таблиці «users» відноситься до поля «user» у таблиці «decks» як «один до багатьох». Поля «deck» та «user» у таблиці «cards» відносяться до полей «name» у таблицях «decks» та «users» відповідно, як «один до багатьох».

Поле «name» у таблиці «users», а також у «decks» в межах одного користувача (поля «user»), мають бути унікальним. Перевірку на унікальність забезпечує API.

Графічна схема бази даних, створена за допомогою сервісу dbdiagram.io [44] представлена на рисунку 3.7.

Рисунок 3.7 – Графічна схема бази даних

3.3 Розробка застосунку

3.3.1 Розробка клієнтської частини

Клієнтська частина представляє собою вебсайт для веббраузерів. Всі вебсторінки мають формат «html». Типовий html-документ проекту, на прикладі вхідної сторінки авторизації «index.html», має таку структуру (рис 3.8).

Рисунок 3.8 – Розбір коду типового html-докумену проекту, «index.html»

На сторінці авторизації «index.html» викликається метод «log_into_current_user», що переходить на сторінку з колодами, якщо у браузері активна сесія користувача. Якщо активної сесії немає – виконання скрипту продовжується і сайт залишається на сторінці авторизації (рис 3.9).

Рисунок 3.9 – Зовнішній вигляд сторінки авторизації «index.html»

Далі, як і на всіх сторінках вебсайту проекту, викликається метод «start_connection_checker», що робить запит до методу API «flashcards/check_connection» кожну секунду. Якщо відповіді від API

немає – з’являється повідомлення про розрив з’єднання з сервером, що закриває всю сторінку і не дає виконувати на ній дії до відновлення з’єднання (рис 3.10). Запити до API продовжуються, і коли відповідь – булеве «true», повідомлення зникає.

Рисунок 3.10 – Повідомлення про розрив з’єднання з сервером

Розбір коду методу «start_connection_checker», з прикладом відправлення запиту до API без даних у заголовку «body» та отриманням відповіді з простим булевим значенням, показано на рисунку 3.11.

Рисунок 3.11 – Розбір коду методу «start_connection_checker»

У формах вебсайту, наприклад формі авторизації (рис.3.9), поля мають обмеження на кількіть смволів за допомогою властивості «maxlength» (рис 3.12). Тобто за дотриманням розміру вхідних даних відповідає клієнтська частина.

Рисунок 3.12 – Властивість «maxlength»

Після введення даних та натискання кнопки «log in», метод «login» відправляє запит з даними форми до методу API «/flashcards/login/», який, у відповідь, повертає результат аутентифікації.

Розбір коду методу «login», з прикладом відправлення запиту до API з тектовими даними у заголовку «body» та отриманням відповіді з простим значенням, показано на рисунку 3.13.

Рисунок 3.13 – Розбір коду методу «login»

Якщо результат – «AUTHENTICATED», визначається файл cookie «current_user» зі значенням-іменем користувача для визначення поточної сесії у браузері, і далі переходить на сторінку з колодами «decks.html».

Якщо, наприклад, введено невірний пароль, показується відповідна помилка (рис 3.14).

Рисунок 3.14 – Невірний пароль

При натисканні кнопки «sign up» відбувається перехід на сторінку реєстрації «sign_up.html» (рис 3.15).

Рисунок 3.15 – Зовнішній вигляд сторінки авторизації «index.html»

У формі на цій сторінці доступне завантаження зображення. Це елемент «input» з обмеженням до форматів «png», «gif» та «jpeg» (рис 3.16).

```
<input id="imginput" type="file" name="myImage" accept="image/png, image/gif, image/jpeg" />
```

Рисунок 3.16 – Завантаження зображення

Розбір частини коду методу «signup» з прикладом відправлення запиту до API з файлом у заголовку «body» показано на рисунку 3.17.

Рисунок 3.17 – Розбір коду відправлення файлу до API з методу «signup»

Запит відправляється, якщо користувач був успішно зареєстрований, і зображення профілю взагалі було завантажено. Якщо зображення не було завантажено користувачем, йому присвоюється стандартне забраження профілю.

При успішній реєстрації з'являється повідомлення «welcome» та відбувається перехід на сторінку авторизації «index.html».

Після акторизації відбувається перехід на сторінку з колодами «decks.html» (рис 3.18). Ця сторінка є «домашньою» для користувача, на ній відображаються всі колоди у порядку часу останньої інтеракції. Після реєстрації новому користувачу додається колода «fun with flags» для прикладу і демонстрації можливостей сервісу.

Рисунок 3.18 – Зовнішній вигляд сторінки з колодами «decks.html»

При натисканні на заголовок сторінки «flashcards» відбувається перехід на домашню сторінку корситувача (з колодами).

При натисканні на кнопку «add deck» відбувається перехід на сторінку створення колоди «add_deck.html» (рис. 3.19).

Рисунок 3.19 – Зовнішній вигляд сторінки створення колоди «add_deck.html»

У верхньому правому куті знаходиться ім’я користувача та його зображення, при натисканні на них відбувається перехід на сторінку огляду профілю «profile.html» (рис. 3.20).

Рисунок 3.20 – Зовнішній вигляд сторінки огляду профілю «profile.html»

Розбір коду методу «get_profile_pic» з прикладом відправлення запиту до API для отримання зображення з серверу показано на рисунку 3.21.

Рисунок 3.21 – Розбір коду методу «get_profile_pic»

На сторінці огляду профілю (рис. 3.20), при натисканні на кнопку «log out» браузер завершає поточну сесію користувача, присвоюючи cookie-файлу «current_user_name» значення «unauthorized».

При натисненні на кнопку «edit profile» відбувається перехід на сторінку редагування користувача «edit_profile.html» (рис. 3.22).

Рисунок 3.22 – Зовнішній вигляд сторінки редагування профілю «edit_profile.html»

На домашній сторінці з колодами (рис. 3.18), при наведенні на колоду з’являються кнопки «learn», «add» та «edit» (рис. 3.23).

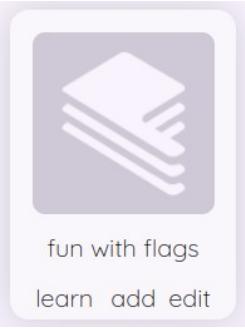


Рисунок 3.23 – Варіанти взаємодії з колодою

При натисненні на кнопки взаємодії з колодами, у колоди оновлюється час останньої інтеракції, тому на сторінці з іншими колодами вона стане першою в списку.

При натисненні на кнопку «edit» відбувається перехід на сторінку редагування колоди «edit_deck.html» (рис. 3.24).

Рисунок 3.24 – Зовнішній вигляд сторінки редагування колоди «edit_deck.html»

У верхній частині сторінки можна змінити дані колоди або видалити її, а нижче відображаються усі картки у колоді, відсортовані за чергою відображення у режимі вивчення. Смуга на картці розділяє передню та зворотну сторони, а також кожну окрему картку можно видалити з колоди кнопкою «delete card».

При натисненні на кнопку «add» на колоді (рис. 3.23) відбувається перехід на сторінку додання картки «add_card.html» (рис. 3.25).

Рисунок 3.25 – Зовнішній вигляд сторінки створення картки «add_card.html»

На одній стороні картки може міститись до 2 зображень та текст до 5000 символів.

При натисненні на кнопку «learn» на колоді (рис. 3.23) відбувається перехід на сторінку вивчення колоди «learn.html». Після переходу на сторінці відображається передня сторона першої картки в черзі на вивчення (рис. 3.26).

Рисунок 3.26 – Зовнішній вигляд сторінки вивчення колоди «learn.html» (передня сторона картки)

Після натискання кнопки «flip card» відображається зворотна сторона цієї картки (рис. 3.27).

Рисунок 3.27 – Зовнішній вигляд сторінки вивчення колоди «learn.html» (зворотна сторона картки)

У нижній частині зворотної сторони є 5 кнопок, пронумерованих від 1 до 5, де 1 означає, что користувачу було дуже важко пригадати зміст зворотної сторони, а 5 – що це було дуже легко. При натисканні однієї із кнопок, картка змінить номер черги в залежності від відповіді, а на сторінці відобразиться передня сторона вже нової першої в черзі картки, і так далі (цей режим є нескінченним).

3.3.2 Розробка серверної частини

Скрипт запуску API – «main.py» у корені папки API. Розбір його коду показано на рисунку 3.28.

Рисунок 3.28 – Розбір коду скрипту «main.py»

Шляхи методів API підключаються у скрипті «api.py» у папці «api». Розбір його коду показано на рисунку 3.29.

Рисунок 3.29 – Розбір коду скрипту «api.py»

Методи API визначаються скрипті «flashcards.py» за шляхом «api/endpoints». Розбір його коду показано на рисунку 3.30.

Рисунок 3.30 – Розбір коду скрипту «flashcards.py»

Більшість методів API приймають дані у форматі JSON, викликають необхідний метод класу «request_handler» та повертають результат до клієнтської частини. Результатом у різних методів, виходячи із призначення, може бути булеве «True» або «False», рядок з повідомленням або тектом помилки, словник з даними або файл. Всі методи мають тип «post», який дозволяє приймати і повертати дані.

Наприклад, метод «/signup» приймає JSON-дані з даними реєстрації та повертає рядок з результатом реєстрації. Для прикладу методу такого типу, розбір його коду показано на рисунку 3.31.

визначення шляху post-методу
визначення функції
конвертація JSON-даних у словник Python
створення екземпляру службового класу
виклик методу службового класу з вхідними даними
повернення результату до клієнтської частини

```
@router.post("/signup")
async def signup(arbitrary_json: JSONStructure = None):
    data = arbitrary_json
    request_handler = Request_handler()
    result = request_handler.add_user(data)
    return result
```

Рисунок 3.31 – Розбір коду методу «/signup»

З методу «/signup» викликається метод «request_handler.add_user». Розбір його коду показано на рисунку 3.32.

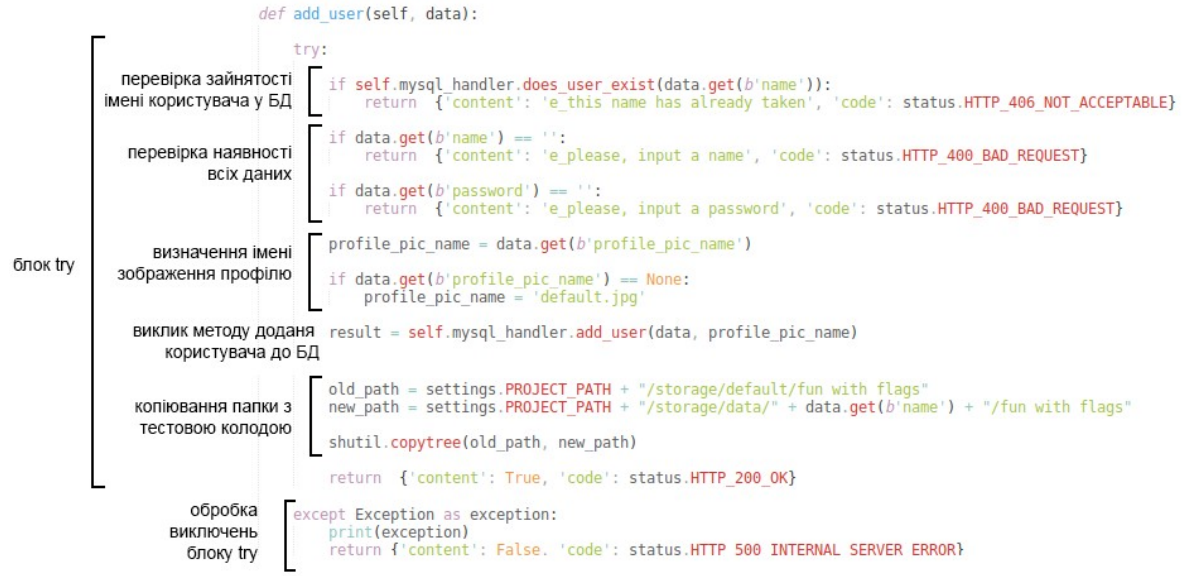


Рисунок 3.32 – Розбір коду методу «request_handler.add_user»

З методу «request_handler.add_user» викликається метод «mysql_handler.add_user». Розбір його коду, для прикладу відправлення SQL-запиту до бази даних MySQL показано на рисунку 3.33.

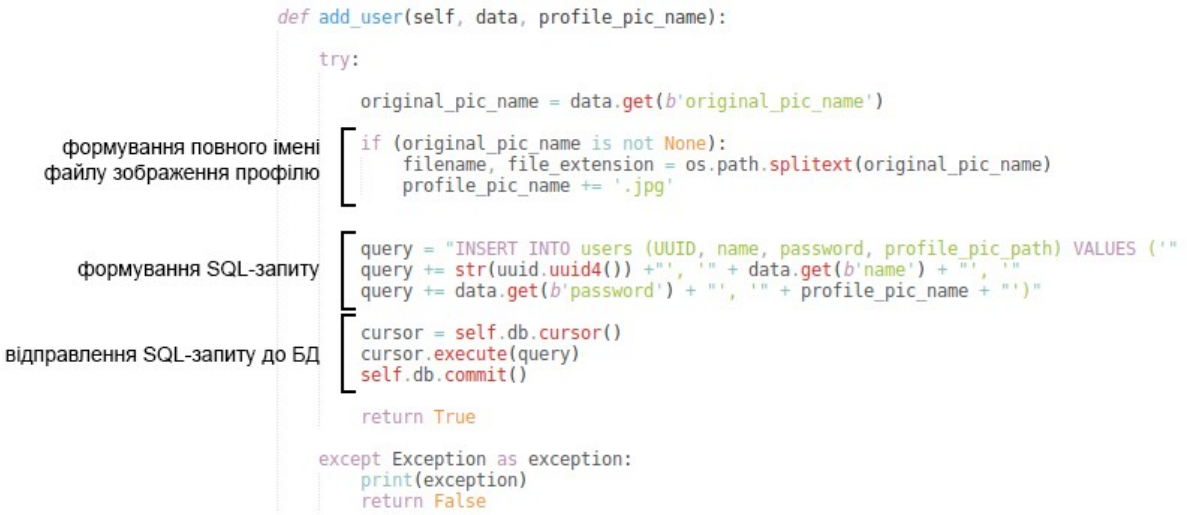
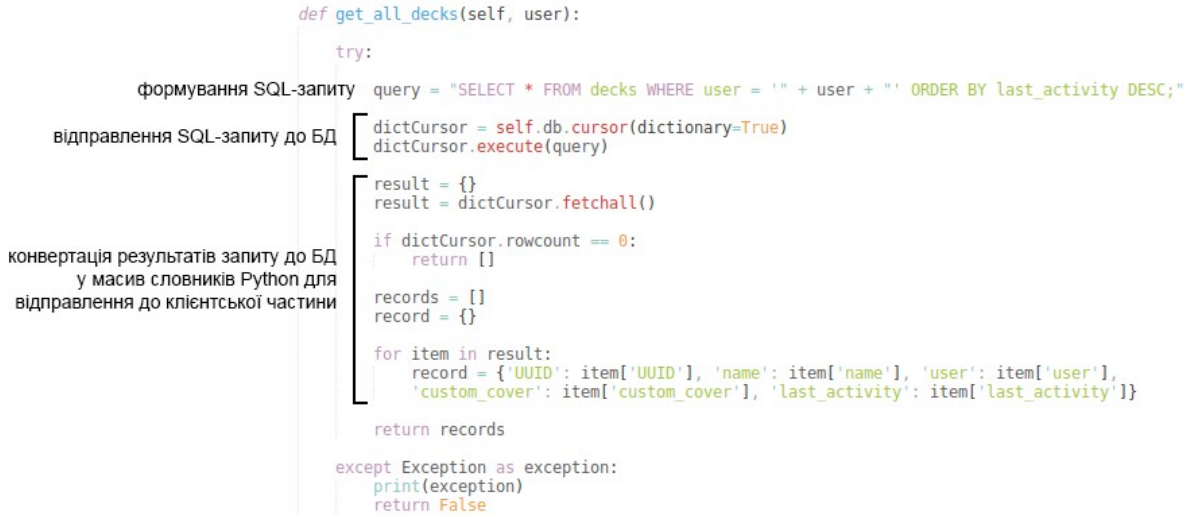


Рисунок 3.33 – Розбір коду методу «mysql_handler.add_user»

SQL-запити з інших методів відправляються аналогічно, відрізняються вони текстом запиту, вхідними даними та формуванням вихідних.

Для прикладу, розбір коду методу отримання масиву з даними всіх колод «mysql_handler.get_all_decks» показано на рисунку 3.34.




```
def get_all_decks(self, user):
    try:
        query = "SELECT * FROM decks WHERE user = '" + user + "' ORDER BY last_activity DESC;"
        dictCursor = self.db.cursor(dictionary=True)
        dictCursor.execute(query)
        result = {}
        result = dictCursor.fetchall()

        if dictCursor.rowcount == 0:
            return []

        records = []
        record = {}

        for item in result:
            record = {'UUID': item['UUID'], 'name': item['name'], 'user': item['user'],
            'custom_cover': item['custom_cover'], 'last_activity': item['last_activity']}

        return records

    except Exception as exception:
        print(exception)
        return False
```

Рисунок 3.34 – Розбір коду методу «mysql_handler. get_all_decks»

Приклад SQL-запитів на видалення записів, з методу видалення користувача «mysql_handler. delete_user» показано на рисунку 3.35.


```
cursor = self.db.cursor()
cursor.execute("DELETE FROM users WHERE name = '" + name + "';")
self.db.commit()
cursor.execute("DELETE FROM decks WHERE user = '" + name + "';")
self.db.commit()
cursor.execute("DELETE FROM cards WHERE user = '" + name + "';")
self.db.commit()
```

Рисунок 3.35 – SQL-запити на видалення записів

Приклад SQL-запитів на оновлення (редагування) записів, з методу редагування користувача «mysql_handler.edit_user» показано на рисунку 3.36. У ньому формується складений запит, у який додаються нові дані, виходячи з того, чи були вони відправлені до API.

початок запиту на оновлення користувача

```
query_user = "UPDATE users SET "
```

додання до запиту на оновлення корситувача нового імені, якщо воно було відправлене до API

```
if (new_user_name != ''):
    query_user += "name = '"
    query_user += new_user_name
    query_user += "'"
    name_inputted = True
```

оновлення імені користувача у записах про колоди

```
    query_decks = "UPDATE decks SET user = '" + new_user_name
    query_decks += "' WHERE user = '" + old_user_name + "';"
    cursor = self.db.cursor()
    cursor.execute(query_decks)
    self.db.commit()
```

оновлення імені користувача у записах про картки

```
    query_cards = "UPDATE cards SET user = '" + new_user_name
    query_cards += "' WHERE user = '" + old_user_name + "';"
    cursor = self.db.cursor()
    cursor.execute(query_cards)
    self.db.commit()
```

додання до запиту на оновлення корситувача нового паролю, якщо він був відправлений до API

```
password = data.get(b'password')

if (password != ''):
    password_inputted = True
    if (name_inputted): query_user += ", "
    query_user += "password = '"
    query_user += password
    query_user += "'"
```

додання до запиту на оновлення корситувача імені нового зображення профілю, якщо воно було відправлене до API

```
custom_profile_pic = data.get(b'profile_pic_path')

if (custom_profile_pic is not None):
    if (name_inputted or password_inputted): query_user += ", "
    query_user += "profile_pic_path = '"
    query_user += custom_profile_pic
    query_user += "'"
```

кінець запиту на оновлення корситувача

```
query_user += " WHERE name = '" + old_user_name + "';"
```

відправлення запиту на оновлення корситувача

```
cursor = self.db.cursor()
cursor.execute(query_user)
self.db.commit()
```

Рисунок 3.36 – SQL-запити на оновлення записів

Після сторення нового користувача, у базі даних зберігаються дані про нього, а також його колоди та картки (рис. 3.37).

mysql> SELECT * FROM users;

UUID	name	password	profile_pic_path
30f921d4-15f0-4432-bf9c-e7f75230c96e	mio	1	02db6e50-0e61-46c3-8be2-024bf800f5f2.jpg

1 row in set (0.02 sec)

mysql> SELECT * FROM decks WHERE user = 'mio';

UUID	name	user	custom_cover	last_activity
e7a1d1be-0869-4002-bcb1-cacac66ac5a4	fun with flags	mio	1	1654136586.2494605

1 row in set (0.08 sec)

mysql> SELECT * FROM cards WHERE user = 'mio';

UUID	deck	user	queue	f_pic1_path	f_pic2_path	f_text	b_pic1_path	b_pic2_path	b_text
0583d7e2-b8eb-4a6f-854a-136f47d3db94	fun with flags	mio	4	549a6435-05e1-41dc-885c-1127ebf71458.png		Name the country	01aac522-435e-44cf-b7dd-196ae481055c.png	609705f2-8279-4428-b566-e8647ce7cb96.png	Kyrgyzstan
154a873a-a5fe-4627-a9fc-abe6c4fa2d6c	fun with flags	mio	7	2d8a6a7b-17e0-4ae9-b3e3-ec0acb8d2755.png	0140419b-95e3-44d1-8495-7a4cba664061.png	Name the country			Greenland
1ad2538e-a6ec-4b9f-accc-300d976cb5bc	fun with flags	mio	6			Which flag is designed by Lafayette?	81c3bb42-f1d8-4423-97c5-bfe0e550e708.png		France flag
7151affd-ff1b-4e4c-8056-27913fefa2b2	fun with flags	mio	2	c3326784-d8cd-4485-a0ea-200919473824.png		What is the traditional English name for the flags flown to identify a pirate ship about to attack?			Jolly Roger
8e374ce5-d843-4ae8-90c9-b9c01a5ed215	fun with flags	mio	5	89890c99-5f60-4e82-b458-ea794d181217.png	6b4fd2e3-b97d-4a54-93e2-3a0fae1c9c7e.png	Name the continent			Africa
a13dee9f-2733-4db1-ab9e-cc325fb21c3a	fun with flags	mio	10			How many human legs are on a Sicily flag?	74eec214-4acd-4b86-8712-2d3f73370629.png	609705f2-8279-4428-b566-e8647ce7cb96.png	3 legs
ad6b9da7-18f0-4d8e-9f2e-b886363eb6aa	fun with flags	mio	8	a76f0650-296f-41cb-b810-5085683e4c55.png		Name the country	281f1eae-7089-4fed-92fb-c63343a41797.png	b493c2af-e209-4184-8aac-f97a55646161.jpg	Quatar
c2cc4268-8099-4a99-9158-29ad57593883	fun with flags	mio	3			The only flag in the world that does not have a rectangular shape	98cc0e1d-90b3-4908-811d-fe16fcbbe10d.png	81a60fc4-43a2-4ec9-bd5b-45925c258e7b.png	The flag of Nepal
eabf9b3e-cf75-4415-8856-8f9015cd65ac	fun with flags	mio	9	541c7cc6-79f1-4217-82ae-b87ad72e108d.png	0e763d8d-1d4a-4992-8930-80b5dab1d104.png	Name the continent			South America
f143b7fa-832e-4f79-b755-3c4a3ee1c100	fun with flags	mio	1			Which flag has a red dragon? (name the country)	418a014c-b10d-4b8d-8dff-d5f0bbae2bbc.png	7d411c45-7c18-4438-b0d9-80c8e0705fcd.png	Wales

10 rows in set (0.02 sec)

Рисунок 3.37 – Дані користувача у базі даних

Файли зображень профілю, колод та карток зберігаються у папці «storage» (рис 3.38). Кожне зображення має унікальне ім'я у форматі UUID версії 4 для уникнення конфліктів імен файлів у файловій системі.

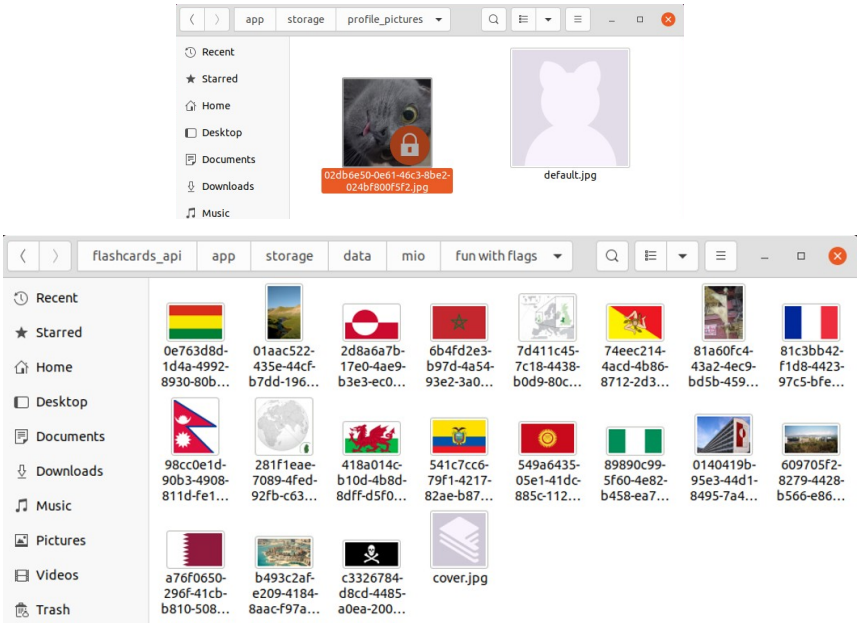


Рисунок 3.38 – Файли зображень на сервері

Приклад завантаження зображення до серверу наведено у рисунку 3.39 на прикладі методу API для завантаження зображення профілю «/post_profile_pic».

приймає файл зображення
розділення імені файлу та розширення
визначення повного шляху до файлу
завантаження файлу до серверу

```python
@router.post("/post_profile_pic")
async def post_profile_pic(request: Request, file: UploadFile = File(...)):
    filename, file_extension = os.path.splitext(file.filename)
    file.filename = 'storage/profile_pictures/' + request.headers.get('uuid_name') + '.jpg'
    with open(f'{file.filename}', "wb") as buffer:
        shutil.copyfileobj(file.file, buffer)
    return True
```

Рисунок 3.39 – Розбір коду методу «/post_profile_pic»

Приклад отримання зображення з серверу наведено у рисунку 3.40 на прикладі методу API для завантаження зображення профілю «/get_profile_pic».

приймає JSON з іменем користувача
отримання повного шляху до файлу зображення
відправлення файлу до клієнтської частини

```
@router.post("/get_profile_pic")
async def get_profile_picture(arbitrary_json: JSONStructure = None):
    data = arbitrary_json
    request_handler = Request_handler()
    result = request_handler.get_profile_picture(data)
    return FileResponse(result)
```

Рисунок 3.40 – Розбір коду методу «/get_profile_pic»

Головний режим сервісу – вивчення карток, працює за принципом черги. У колоді всі картки мають значення «queue» – номер у черзі на вивчення (зберігається у базі даних). Нові картки, що створюються, отримують номер черги «1», а інші картки «зміщуються», збільшуючи свої значення номеру черги на 1 (рис 3.41).

```
query = "UPDATE cards SET queue = queue+1 WHERE user = '" + data.get(b'user') + "' AND deck = '" + deck.get('name') + "';"
```

Рисунок 3.41 – Зміщення черги карток у колоді при сторенні нової

Після перегортання картки у режимі вивчення, користувач має оцінити, наскільки легко було згадати зміст зворотної сторони, за шкалою від 1 до 5. Виходячи із оцінки та кількості карт у колоді, обчислюється новий номер черги для цієї картки (рис 3.42).

приймає кількість карток та оцінку
позиція через одну картку
проміжні позиції
позиція у кінці колоди

```
def get_new_queue(self, cards_number, rate):
    if rate == 1:
        return 2
    if rate == 2:
        return math.floor(cards_number * 0.25)
    if rate == 3:
        return math.floor(cards_number * 0.5)
    if rate == 4:
        return math.floor(cards_number * 0.75)
    if rate == 5:
        return cards_number
```

Рисунок 3.42 – Розбір коду методу «request_handler.get_new_queue»

Номер черги цієї картки оновлюється, а картки до позиції нового номеру черги включно зменшують свій номер черги на 1 (рис. 3.43).

Рисунок 3.43 – Оновлення черги з методу «mysql_controller.move_card»

3.4 Тестування

Для тестування застосунку було виконано наступний перелік дій:

- на всіх сторінках вимикався та вмикався сервер: сторінки реагували відображенням повідомлення про розрив з’єднання;
- на сторінці створення нового користувача було не введено ім’я, пароль, невірний пароль-підтвердження, та вже зайняте ім’я: при цих умовах на сторінці відображалися відповідні повідомлення;
- створено користувачів із завантаженим зображенням профілю і без нього: операції проходили коректно;
- на сторінці аутентифікації було не введено ім’я, пароль, введено неіснуюче ім’я користувача та невірний пароль: при цих умовах на сторінці відображалися відповідні повідомлення, при правильних умовах аутентифікація користувача проходила коректно;
- новим користувачам успішно додавалася тестова колода;
- на сторінці перегляду профілю після натиснення кнопки «log out» завершення сесії користувача відбувалося коректно;
- на сторінці редагування профілю було не введено даних, введено часткові дані: оновлення даних відбувалося коректно;
- на сторінці редагування профілю при натисненні «delete user» було введено невірне підтвердження: відображалося відповідне повідомлення,

після успішного підтвердження видалення користувача відбувалося коректно;

- на сторінці створення колоди було не введено ім’я та введено вже зайняте ім’я: при цих умовах на сторінці відображалися відповідні повідомлення, при правильних умовах створення колоди (із завантаженням зображенням і без) проходили коректно;
- на сторінці з колодами коректно відображались усі створені колоди у порядку часу останньої інтеракції;
- при взіємодії з колодою час останньої інтеракції оновлювався коректно;
- при натисненні кнопки «learn» на пустій колоді відображалося відповідне повідомлення;
- на сторінці редагування колоди коректно відображались усі картки колоди у порядку черги для вивчення;
- на сторінці редагування колоди було не введено даних, введено часткові дані: оновлення даних відбулося коректно;
- на сторінці редагування колоди при натисненні «delete deck» після підтвердження видалення колоди відбувалося коректно;
- на сторінці редагування колоди при натисненні «delete card» після підтвердження видалення картки з колоди відбувалося коректно;
- при переході на сторінку вивчення картки передня сторона відображалася коректно;
- при натисненні кнопки «flip card» зворотна сторона картки відображалася коректно;
- при виборі оцінки під зворотною стороною картки, вона коректно зміщувалася у черзі на вивчення, далі на сторінці відображалася передня сторона вже наступної картки, і далі цикл працював коректно.

Після виконання переліку дій не виявлено помилок, виключень та критичних помилок. На основі результатів визначено, що програмний продукт є стабільним та пригодним для практичного використання.

3.5 Вимоги до програмного та апаратного забезпечення користувача та власника сервера

На основі тестування програмного продукту на декількох ПК з різним апаратним забезпеченням та у віртуальній машині з різними параметрами, а також на основі мінімальних системних вимог ОС Ubuntu [45] визначено мінімальні системні вимоги для власника сервера та користувача програмного продукту.

Мінімальні системні вимоги для власника сервера:

- процесор: AMD FX-8300 або Intel Core i5-3570;
- оперативна пам’ять: 8 гігабайт;
- накопичувач: твердотільний від 120 гігібайт;
- відеоадаптер: з підтримкою VGA та роздільної здатності від 1024x768;
- наявність DVD-приводу або USB;
- наявність монітору, миші, клавіатури;
- вільний доступ до мережі інтернет.

Мінімальні системні вимоги для користувача:

- процесор: AMD Athlon II X2 240 або Intel Core i3-10100;
- оперативна пам’ять: 4 гігабайти;
- накопичувач: 25 гігібайт;
- відеоадаптер: з підтримкою VGA та роздільної здатності від 1024x768;
- наявність монітору, миші, клавіатури;

- вільний доступ до мережі інтернет.

3.6 Інструкція до розгортки програмного продукту

Наступна інструкція до розгортки середи розробки призначена для ОС Ubuntu Desktop з командним рядком (терміналом) bash. Всі команди виконуються у bash від імені суперкористувача (адміністратора). Цей режим активується командою «sudo -s». У лістингах команди для bash від імені суперкористувача позначені символом «$». Ім’я користувача-адміністратора Ubuntu у прикладах команд та шляхів визначається як «ubuntu_user».

Для встановлення та налаштування вебсерверу Apache необхідно виконати команди з лістингу 3.1 [46]. Ім’я домену у прикладах визначено як «flashcards.io».

Лістинг 3.1 – Встановлення Apache Web Server

```
$ apt update
$ apt install apache2
$ ufw allow ‘Apache’
$ sudo mkdir /var/www/flashcards.io
$ chown -R $USER:$USER /var/www/flashcards.io
$ -R 755 /var/www/flashcards.io
$ nano /etc/apache2/sites-available/flashcards.io.conf
```

Далі, для налаштування віртуального хосту Apache, необхідно вставити код з лістингу 3.2 та зберегти файл через «Control+X», «Y» та «Enter».

Лістинг 3.2 – Налаштування віртуального хосту Apache

```
<VirtualHost *:80>
    ServerAdmin webmaster@localhost
    ServerName flashcards.io
    ServerAlias www.flashcards.io
    DocumentRoot /var/www/flashcards.io
    ErrorLog ${APACHE_LOG_DIR}/error.log
    CustomLog ${APACHE_LOG_DIR}/access.log combined
</VirtualHost>
```

Для завершення налаштування вебсерверу Apache необхідно виконати команди з лістингу 3.3.

Лістинг 3.3 – Встановлення Apache Web Server

```
$ a2ensite flashcards.io.conf
$ a2dissite 000-default.conf
$ apache2ctl configtest
$ systemctl restart apache2
```

На диску до пояснювальної записки випускної роботи, у папці «застосунок» знаходиться архів «flashcards.io.zip», який необхідно розархівувати за шляхом «/var/www/» з відповідним іменем (рис 3.44).

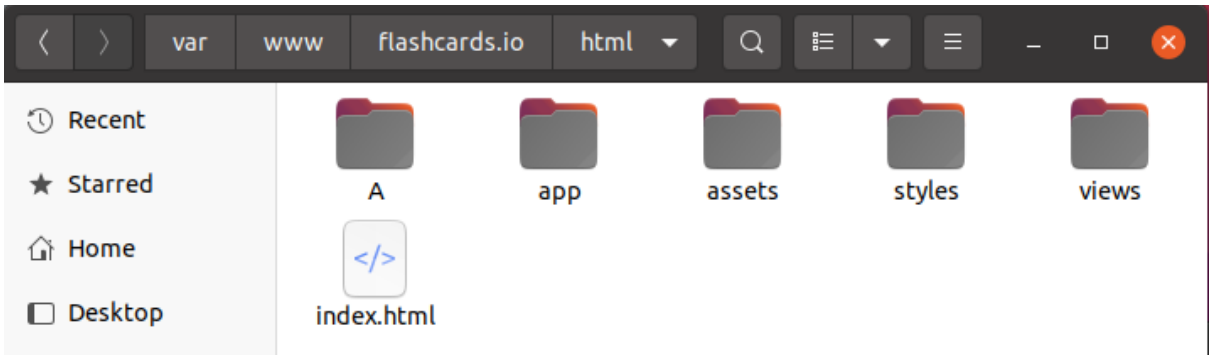


Рисунок 3.44 – файли вебсерверу

Для встановлення MySQL необхідно виконати команди з лістингу 3.4 [47].

Лістинг 3.4 – Встановлення MySQL

```
$ apt update
$ apt install mysql-server
$ mysql_secure_installation
```

Для налаштування MySQL необхідно виконати команди з лістингу 3.5 зі своїми параметрами у полях «username», «password» та «db». У лістингах команди для командноного інтерфейсу MySQL позначені символом «>».

Лістинг 3.5 – Налаштування MySQL

```
$ mysql
> SELECT user,authentication_string,plugin,host FROM
mysql.user; ALTER USER 'root'@'localhost' IDENTIFIED WITH
caching_sha2_password BY 'password';
> FLUSH PRIVILEGES;
> exit
$ mysql -u root -p
```

```
> CREATE USER 'username'@'localhost' IDENTIFIED BY
'password';
> GRANT ALL PRIVILEGES ON *.* TO 'username'@'localhost'
WITH GRANT OPTION;
> CREATE DATABASE db;
> exit
```

Для встановлення Anaconda та Conda необхідно виконати команди з лістингу 3.6 [48]. Під час встановлення буде необхідно вказати шлях до Anaconda: «/home/anaconda».

Лістинг 3.6 – Встановлення Anaconda та Conda

```
$ sudo apt-get update
$ cd /home
$ wget https://repo.anaconda.com/archive/Anaconda3-2020.11-
Linux-x86_64.sh
$ bash Anaconda3-2020.11-Linux-x86_64.sh
$ rm Anaconda3-2020.11-Linux-x86_64.sh
$ conda update conda
$ conda update anaconda
```

Для створення віртуального середовища, його активації та встановлення необхідних пакетів Python необхідно виконати команди з лістингу 3.7.

Лістинг 3.7 – Налаштування віртуального середовища

```
$ conda create -n flashcards python=3.9.2
$ conda activate flashcards
$ pip install mysql-connector-python
```

```
$ pip install uvicorn
$ pip install fastapi
$ pip install pydantic[dotenv]
$ pip install python-multipart
$ pip install httpx
$ pip install aiofiles
```

На диску до пояснювальної записки випускної роботи, у папці «застосунок» знаходиться архів «flashcards_api.zip, який необхідно розархівувати за шляхом «/home/ubuntu_user» з відповідним іменем (рис 3.45).

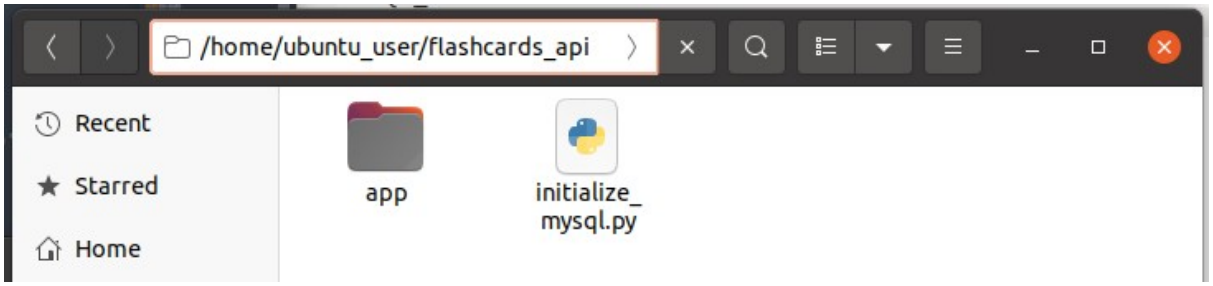


Рисунок 3.45 – файли API

Для ініціалізації бази даних MySQL необхідно виконати команди з лістингу 3.8.

Лістинг 3.8 – Ініціалізація бази даних

```
$ conda activate flashcards
$ cd /home/ubuntu_user/flashcards_api
$ python3 initialize_mysql.py
```

Далі необхідно вказати дані MySQL у командному рядку, або залишити поле пустим для вибору за замовчуванням (рис. 3.46). Про успішне виконання скрипту сигналізує повідомлення «success».

```
input mysql host: (default = 'localhost')

input mysql user: (default = 'username')
tommy
input mysql input_password: (default = 'password')
ijf39f$LKfe21
input mysql database for flashcards: (default = 'db')
flashcards_db
success
```

Рисунок 3.46 – Ініціалізація бази даних MySQL

Для налаштування змінnх API необхідно відредагувати файл налаштувань змінних середовища «.env» (лістинг 3.9).

Лістинг 3.9 – Налаштування змінних API

```
$ cd /home/ubuntu_user/flashcards_api/app
$ nano .env
```

Далі, при необхідності, замінити значення «PORT» (порт для API), дані від MySQL та шлях до файлів API (рис. 3.47) та зберегти файл.

```
  GNU nano 4.8                          .env                          Modified
PORT = 1987

MYSQL_HOST = "localhost"
MYSQL_USER = "user"
MYSQL_PASSWORD = "password"
MYSQL_DB = "db"

PROJECT_PATH = "/home/ubuntu_user/flashcards_api/app"
```

Рисунок 3.47 – Налаштування API за замовчуванням

Для налаштування змінnх вебсерверу необхідно відредагувати файл налаштувань змінних середовища «env.js» (лістинг 3.10).

Лістинг 3.10 – Налаштування змінних вебсерверу

$ cd /var/www/flashcards.io/html/app

$ nano env.js

Далі, при необхідності, замінити значення «domain» (ip та порт API), та зберегти файл (рис. 3.48).

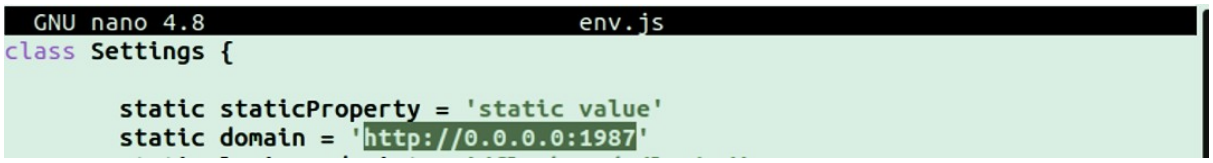


Рисунок 3.48 – Налаштування вебсерверу за замовчуванням

Для встановлення API як служби Ubuntu, яка запускається разом із системою, необхідно скопіювати у директорію служб файл служби API та відредагувати його (лістинг 3.11).

Лістинг 3.11 - редагування файлу служби API

$ cp /home/a/flashcards_api/flashcards_api.service
/lib/systemd/system/flashcards_api.service

$ nano /lib/systemd/system/flashcards_api.service

Далі, при необхідності, замінити значення «WorkingDirectory» (папка з файлами API), та зберегти файл (рис. 3.49).

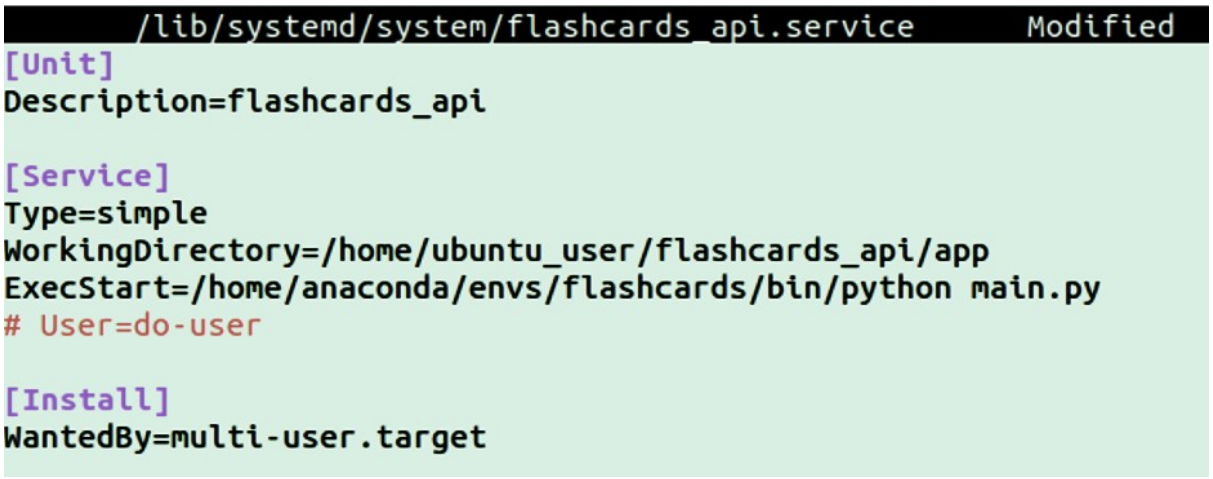


Рисунок 3.49 – Налаштування служби API

Для завершення налаштування необхідно виконати команди з лістингу 3.12.

Лістинг 3.12 - редагування файлу служби API

```
$ sudo systemctl daemon-reload
$ sudo systemctl enable flashcards_api.service
$ sudo systemctl start flashcards_api.service
```

3.7 Висновки за розділом

У даному розділі було:

- побудовано UML діаграми, визначено архітектуру проекту;
- визначено та оформлено файлову та кодову структуру проекту;
- розроблено проект, задокументовано процес та деталі розробки та реалізації;
- протестовано проект, визначено його стабільність та пригодність для практичного використання;

- визначено вимоги до програмного та апаратного забезпечення користувача та власника сервера;
- оформлено інструкцію до розгортки програмного продукту на сервері.

ВИСНОВОК

У повсякденному житті постійно необхідно щось запам’ятовувати. Для цього були розроблені спеціальні техніки: «картки для запам’ятовування» у зв’язці з технікою «інтервального повторення».

Метою роботи було створення програми для роботи з картками для запам’ятовування та їх ефективного вивчення.

У ході виконання випускної роботи було:

- проаналізовано актуальність теми: визначено, що тема є актуальною, а розробка програми за цією темою є доцільною.
- оглянуто предметну область: основні визначення, поняття, та техніки мнемоніки, зокрема зв’язку технік «флешкарток» та «інтервального повторення»;
- розглянуто існуючі аналоги, виділено їхні переваги та недоліки;
- на основі аналізу аналогів сформовано список задач проекту;
- визначено та оглянуто стек технологій для проекту;
- побудовано UML діаграми, визначено архітектуру проекту;
- визначено та оформлено файлову та кодову структуру проекту;
- розроблено проект, задокументовано процес та деталі розробки та реалізації;
- протестовано проект, визначено його стабільність та пригодність для практичного використання;
- визначено вимоги до програмного та апаратного забезпечення користувача та власника сервера;
- оформлено інструкцію до розгортки програмного продукту на сервері;
- за результатами виконання оформлено звіт.

Результатом виконання випускної роботи молодшого спеціаліста є вебсервіс «Картки для запам’ятовування».

ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ

1 Флешкартка – Вікіпедія. [Електронний ресурс]. – Режим доступу: www. URL: https://en.wikipedia.org/wiki/Flashcard (дата звернення 30.05.22).

2 Інтервальне повторення – Вікіпедія. [Електронний ресурс]. – Режим доступу: www. URL: https://en.wikipedia.org/wiki/Spaced_repetition (дата звернення 30.05.22).

3 Фавелл Лі Мортімер – Вікіпедія. [Електронний ресурс]. – Режим доступу: www. URL: https://en.wikipedia.org/wiki/Favell_Lee_Mortimer (дата звернення 30.05.22).

4 Навчальні картки – Аналіз – Google Trends. [Електронний ресурс]. – Режим доступу: www. URL: https://trends.google.com/trends/explore?date=all&q=%2Fm%2F029ktr (дата звернення 30.05.22).

5 Сфери навчання Флешкартки Quizlet. [Електронний ресурс]. – Режим доступу: www. URL: https://quizlet.com/166742548/fields-of-study-flash-cards/ (дата звернення 30.05.22).

6 Сфери навчання Флешкартки Quizlet [Електронний ресурс]. – Режим доступу: www. URL: https://quizlet.com/gb/516956909/fields-of-study-flash-cards/ (дата звернення 30.05.22).

7 Мнемоніка – Вікіпедія. [Електронний ресурс]. – Режим доступу: www. URL: https://uk.wikipedia.org/wiki/Мнемоніка (дата звернення 30.05.22).

8 Ефект тестування – Вікіпедія. [Електронний ресурс]. – Режим доступу: www. URL: https://en.wikipedia.org/wiki/Testing_effect (дата звернення 30.05.22).

9 Ефект інтервалу – Вікіпедія. [Електронний ресурс]. – Режим доступу: www. URL: https://en.wikipedia.org/wiki/Spacing_effect (дата звернення 30.05.22).

10 Список програм з флешкатками – Вікіпедія. [Електронний ресурс]. – Режим доступу: www. URL: https://en.wikipedia.org/wiki/List_of_flashcard_software (дата звернення 30.05.22).

11 Застосунки з флешкартками для покращення навчання. [Електронний ресурс]. – Режим доступу: www. URL: https://collegeinfogeek.com/flashcard-apps/ (дата звернення 30.05.22).

12 Офіційний сайт Anki. [Електронний ресурс]. – Режим доступу: www. URL: https://apps.ankiweb.net/ (дата звернення 30.05.22).

13 Офіційний сайт Brainscape. [Електронний ресурс]. – Режим доступу: www. URL: https://www.brainscape.com/ (дата звернення 30.05.22).

14 Офіційний сайт Mnemosyne Project. [Електронний ресурс]. – Режим доступу: www. URL: https://mnemosyne-proj.org/ (дата звернення 30.05.22).

15 Особливості Mnemosyne Project. [Електронний ресурс]. – Режим доступу: www. URL: https://mnemosyne-proj.org/features (дата звернення 30.05.22).

16 Офіційний сайт Quizlet. [Електронний ресурс]. – Режим доступу: www. URL: https://quizlet.com/ (дата звернення 30.05.22).

17 Офіційний сайт Ubuntu. [Електронний ресурс]. – Режим доступу: www. URL: https://ubuntu.com/ (дата звернення 30.05.22).

18 Вступ до Ubuntu – The Sec Master. [Електронний ресурс]. – Режим доступу: www. URL: https://thesecmaster.com/introduction-to-ubuntu/ (дата звернення 30.05.22).

19 Статистика Statista популярності ОС для розробки ПЗ . [Електронний ресурс]. – Режим доступу: www. URL:

https://www.statista.com/statistics/869211/worldwide-software-development-operating-system/ (дата звернення 30.05.22).

20 Microsoft Windows – Вікіпедія. [Електронний ресурс]. – Режим доступу: www. URL: https://en.wikipedia.org/wiki/Microsoft_Windows (дата звернення 30.05.22).

21 Статистика TrueList популярності дистрибутивів Linux. [Електронний ресурс]. – Режим доступу: www. URL: https://truelist.co/blog/linux-statistics/ (дата звернення 30.05.22).

22 Офіційний сайт Python. [Електронний ресурс]. – Режим доступу: www. URL: https://www.python.org/ (дата звернення 30.05.22).

23 Огляд Python – Tutorialspoint. [Електронний ресурс]. – Режим доступу: www. URL: https://www.tutorialspoint.com/python/python_overview.htm (дата звернення 30.05.22).

24 Статистика Tiobe популярності мов програмування. [Електронний ресурс]. – Режим доступу: www. URL: https://www.tiobe.com/tiobe-index/ (дата звернення 30.05.22).

25 Офіційний сайт MySQL. [Електронний ресурс]. – Режим доступу: www. URL: https://www.mysql.com/ (дата звернення 30.05.22).

26 Оглад MySQL. [Електронний ресурс]. – Режим доступу: www. URL: https://dev.mysql.com/doc/refman/8.0/en/what-is-mysql.html (дата звернення 30.05.22).

27 Статистика Statista популярності СУБД. [Електронний ресурс]. – Режим доступу: www. URL: https://www.statista.com/statistics/809750/worldwide-popularity-ranking-database-management-systems/ (дата звернення 30.05.22).

28 Офіційний сайт СУБД Oracle. [Електронний ресурс]. – Режим доступу: www. URL: https://www.oracle.com/database/ (дата звернення 30.05.22).

29 Документація модулю Python MySQL Connector. [Електронний ресурс]. – Режим доступу: www. URL: https://dev.mysql.com/doc/connector-python/en/ (дата звернення 30.05.22).

30 Огляд технології REST API – Red Hat. [Електронний ресурс]. – Режим доступу: www. URL: https://www.redhat.com/en/topics/api/what-is-a-rest-api (дата звернення 30.05.22).

31 Огляд фремворків Python для розробки REST API – RapidAPI. [Електронний ресурс]. – Режим доступу: www. URL: https://rapidapi.com/blog/best-python-api-frameworks/ (дата звернення 30.05.22).

32 Офіційний сайт FastAPI. [Електронний ресурс]. – Режим доступу: www. URL: https://fastapi.tiangolo.com/ (дата звернення 30.05.22).

33 HTML – Вікіпедія. [Електронний ресурс]. – Режим доступу: www. URL: https://en.wikipedia.org/wiki/HTML (дата звернення 30.05.22).

34 CSS – Вікіпедія. [Електронний ресурс]. – Режим доступу: www. URL: https://en.wikipedia.org/wiki/CSS (дата звернення 30.05.22).

35 JavaScript – Вікіпедія. [Електронний ресурс]. – Режим доступу: www. URL: https://en.wikipedia.org/wiki/JavaScript (дата звернення 30.05.22).

36 Офіційний сайт Conda. [Електронний ресурс]. – Режим доступу: www. URL: https://docs.conda.io/en/latest/ (дата звернення 30.05.22).

37 Офіційний сайт Anaconda. [Електронний ресурс]. – Режим доступу: www. URL: https://www.anaconda.com/ (дата звернення 30.05.22).

38 Документація Anaconda. [Електронний ресурс]. – Режим доступу: www. URL: https://docs.anaconda.com/#anaconda-distribution (дата звернення 30.05.22).

39 Офіційний сайт PyCharm. [Електронний ресурс]. – Режим доступу: www. URL: https://www.jetbrains.com/pycharm/ (дата звернення 30.05.22).

40 Особливості PyCharm. [Електронний ресурс]. – Режим доступу: www. URL: https://www.jetbrains.com/pycharm/features/ (дата звернення 30.05.22).

41 Огляд наукових інструментів Python у PyCharm. [Електронний ресурс]. – Режим доступу: www. URL: https://www.jetbrains.com/pycharm/features/scientific_tools.html (дата звернення 30.05.22).

42 Офіційний сайт Postman. [Електронний ресурс]. – Режим доступу: www. URL: https://www.postman.com/ (дата звернення 30.05.22).

43 Огляд Postman. [Електронний ресурс]. – Режим доступу: www. URL: https://www.postman.com/product/what-is-postman/ (дата звернення 30.05.22).

44 Конструктор діаграм баз даних dbdiagram. [Електронний ресурс]. – Режим доступу: www. URL: https://dbdiagram.io/home (дата звернення 30.05.22).

45 Системні вимоги Ubuntu. [Електронний ресурс]. – Режим доступу: www. URL: https://help.ubuntu.com/community/Installation/SystemRequirements (дата звернення 30.05.22).

46 Інструкція до встановлення Apache Web Server. [Електронний ресурс]. – Режим доступу: www. URL: https://www.digitalocean.com/community/tutorials/how-to-install-the-apache-web-server-on-ubuntu-20-04 (дата звернення 30.05.22).

47 Інструкція до встановлення MySQL. [Електронний ресурс]. – Режим доступу: www. URL: https://www.digitalocean.com/community/tutorials/how-to-install-mysql-on-ubuntu-20-04 (дата звернення 30.05.22).

48 Інструкція до встановлення Anaconda. [Електронний ресурс]. – Режим доступу: www. URL:

https://www.digitalocean.com/community/tutorials/how-to-install-the-anaconda-python-distribution-on-ubuntu-20-04 (дата звернення 30.05.22).

ДОДАТКИ

ДОДАТОК А
НАЗВА

ДОДАТОК Б
НАЗВА