



Міністерство освіти і науки України
Національний технічний університет України
“Київський політехнічний інститут імені Ігоря Сікорського”
Факультет інформатики та обчислювальної техніки
Кафедра інформаційних систем та технологій

Лабораторна робота №2

З дисципліни «**Технології розроблення програмного
забезпечення**»

Тема: «**ДІАГРАМА ВАРІАНТІВ ВИКОРИСТАННЯ. СЦЕНАРІЇ
ВАРІАНТІВ ВИКОРИСТАННЯ. ДІАГРАМИ UML. ДІАГРАМИ
КЛАСІВ. КОНЦЕПТУАЛЬНА МОДЕЛЬ СИСТЕМИ**»

Варіант №6

Виконав
студент групи ІА–13:
Костенко П.С.

Перевірив:
Мягкий М. Ю.

Київ 2023

Тема:**..6 Web-browser (proxy, chain of responsibility, factory method, template method, visitor, p2p)**

Веб-браузер повинен мати можливість зробити наступне: мати адресний рядок для введення адреси сайту, переміщатися і відображати структуру html документа, переглядати підключений javascript та css файли, перегляд всіх підключених ресурсів (зображень), коректна обробка відповідей з сервера (коди відповідей HTTP) - переходи при перенаправленнях, відображення сторінок 404 і 502/503.

Завдання:

1. Ознайомитися з короткими теоретичними відомостями.
2. Проаналізуйте тему та намалюйте схему прецеденту, що відповідає обраній темі лабораторії.
3. Намалюйте діаграму класів для реалізованої частини системи.
4. Виберіть 3 прецеденти і напишіть на їх основі прецеденти.
5. Розробити основні класи і структуру системи баз даних.
6. Класи даних повинні реалізувати шаблон Репозиторію для взаємодії з базою даних.
7. Підготувати звіт про хід виконання лабораторних робіт.

Звіт, що подається повинен містити: діаграму прецедентів, діаграму класів системи, вихідні коди класів системи, а також зображення структури бази даних.

Хід роботи:

Діаграма прецедентів:

1. Введення адресного рядка

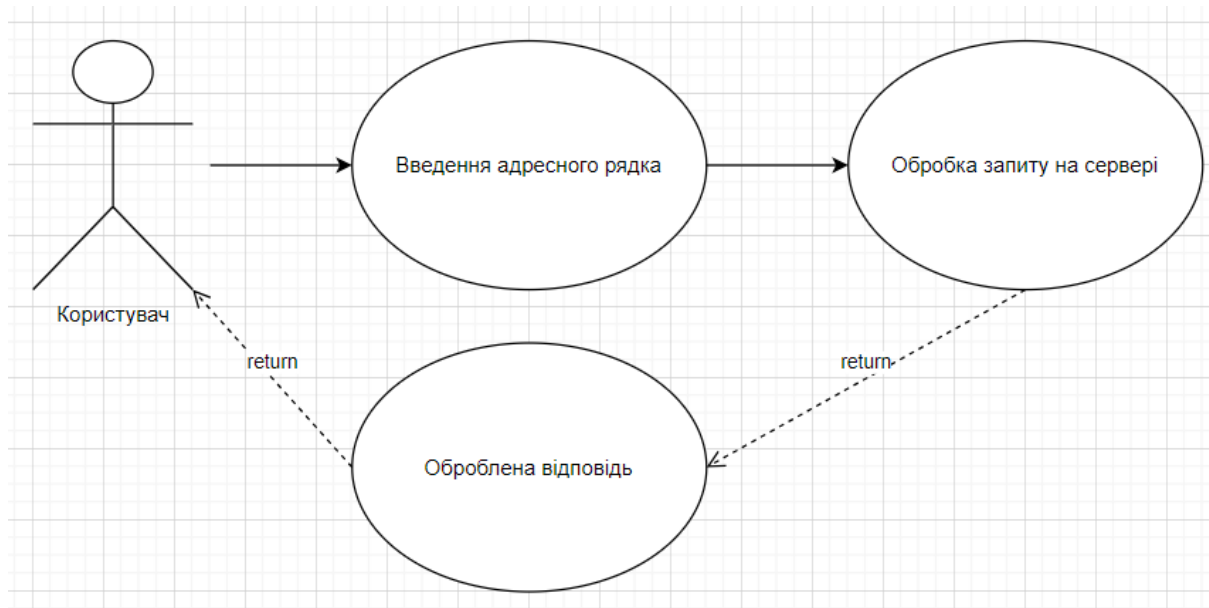


Рис.1.1 Прецедент введення адресного рядка

У першому прецеденті наводиться приклад введення адресного рядка користувачем. Після цього на сервері відбувається обробка даних та оброблена відповідь у вигляді шуканої сторінки з відображенням її вмісту виводиться користувачеві.

2. Перегляд підключених ресурсів:

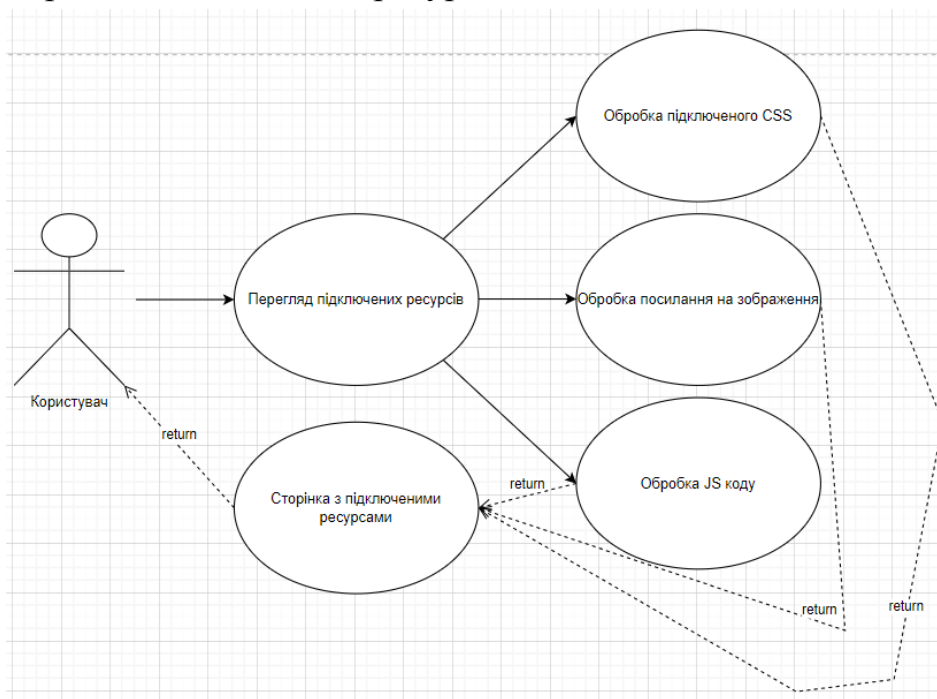


Рис.1.2 Прецедент перегляду підключених ресурсів

У другому прецеденті наводиться приклад спроби перегляду користувачем сторінки з підключеними ресурсами. При цьому на сервері відбувається обробка та повертається сторінка з коректно відображеними підключеними ресурсами.

3. Перегляд сторінки з помилкою:

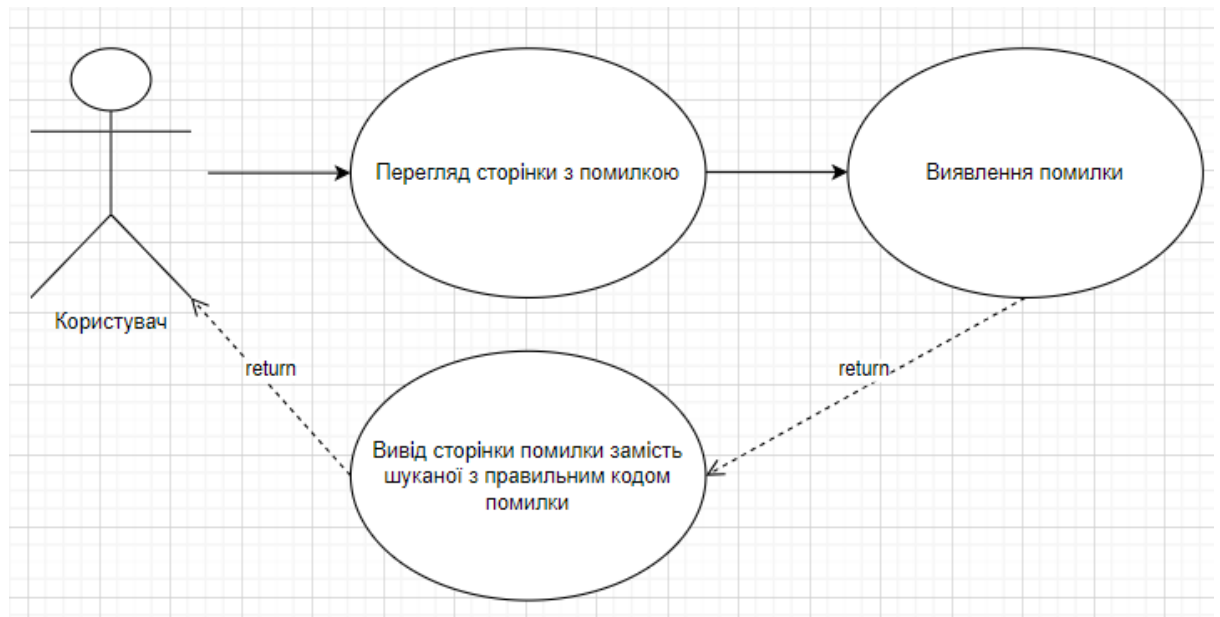


Рис.1.3 Прецедент перегляду сторінки з помилкою

У третьому прецеденті наводиться приклад перегляду сторінки, яка містить помилку. При цьому відбувається виявлення помилки та повертається сторінка помилки з кодом помилки.

Діаграма класів системи:

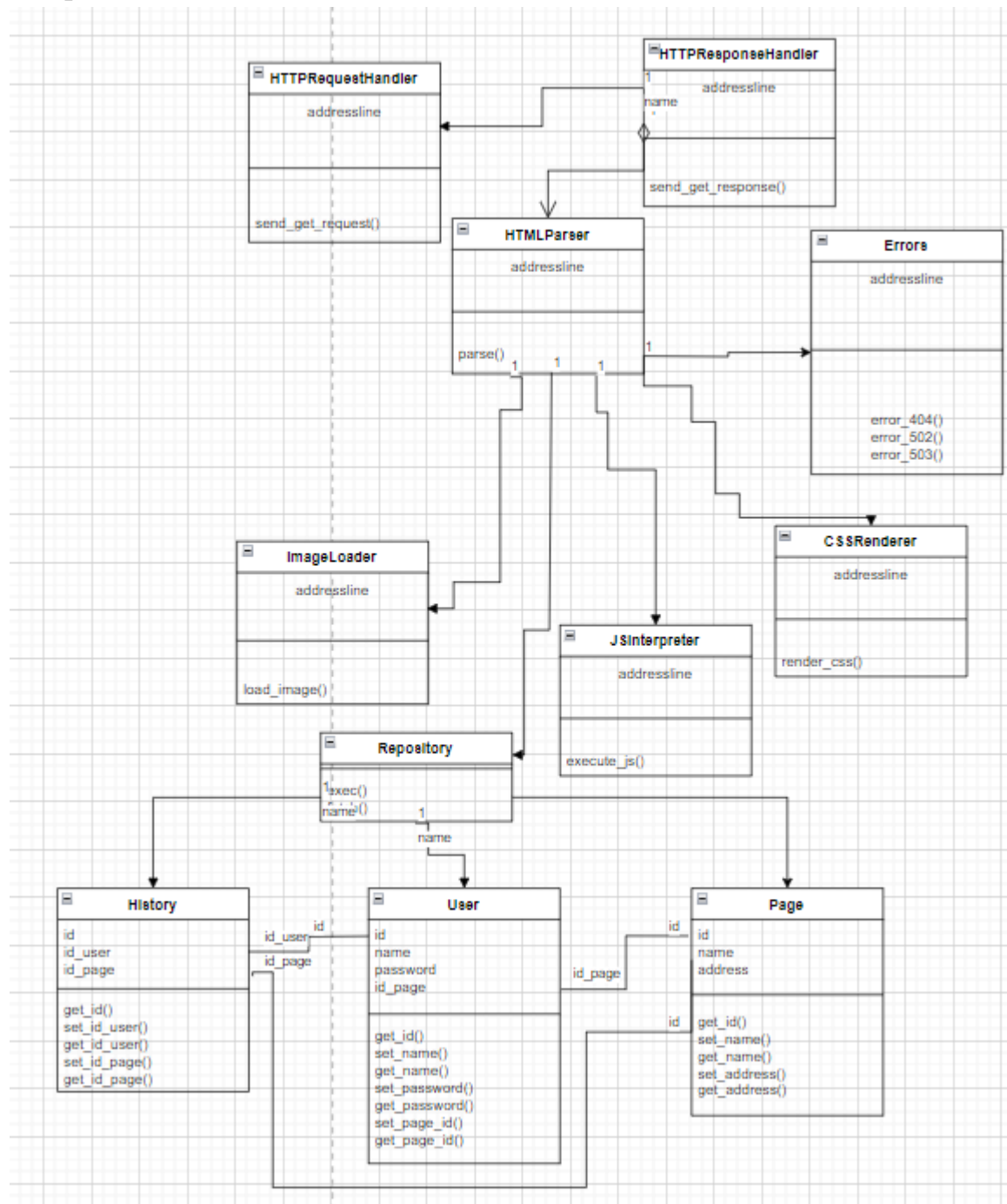


Рис.2.1 Діаграма класів системи

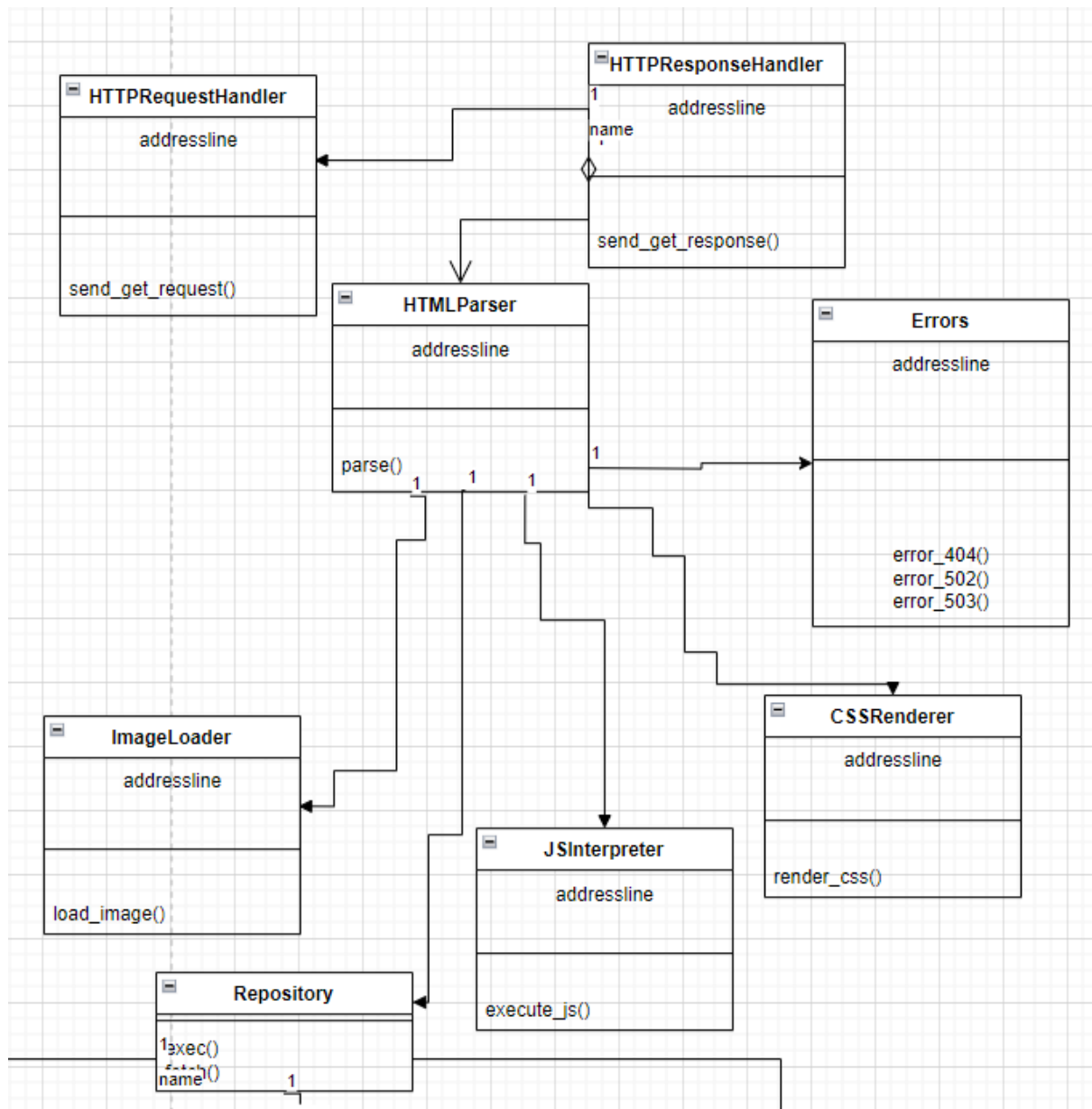


Рис. 2.2 Перша частина діаграми класів

Для кращого огляду діаграму класів було розділено на 2 частини (Рис.2.2 та Рис.2.3).

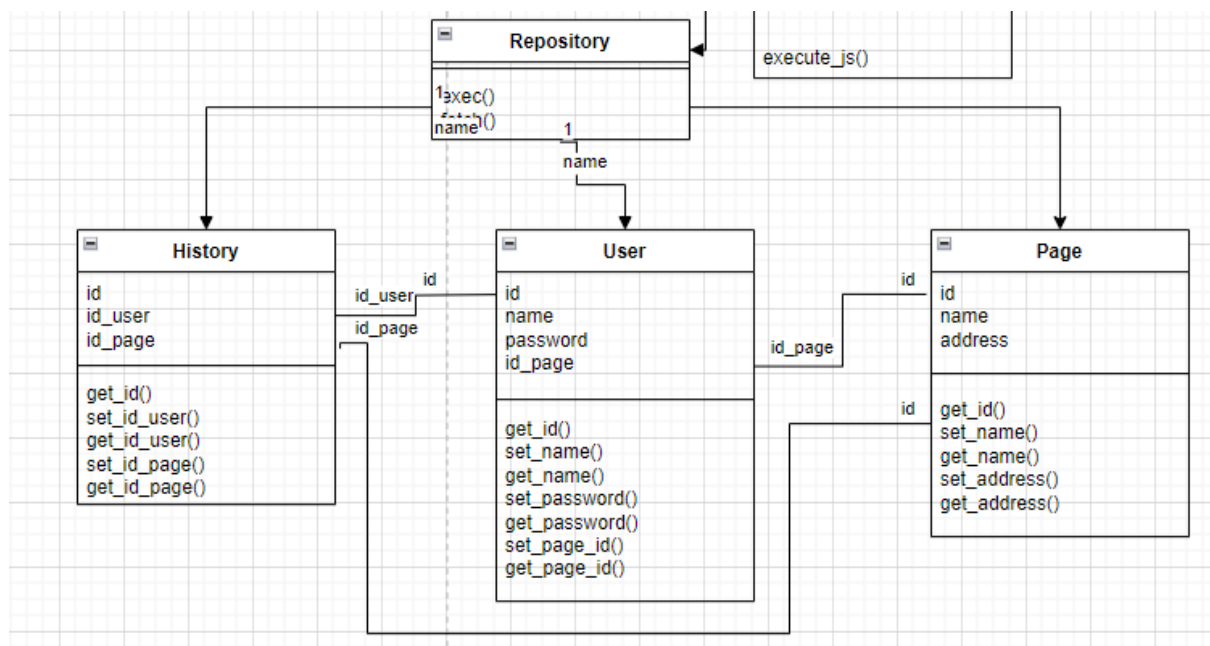


Рис. 2.3 Друга частина діаграми класів

Зображення структури бази даних:

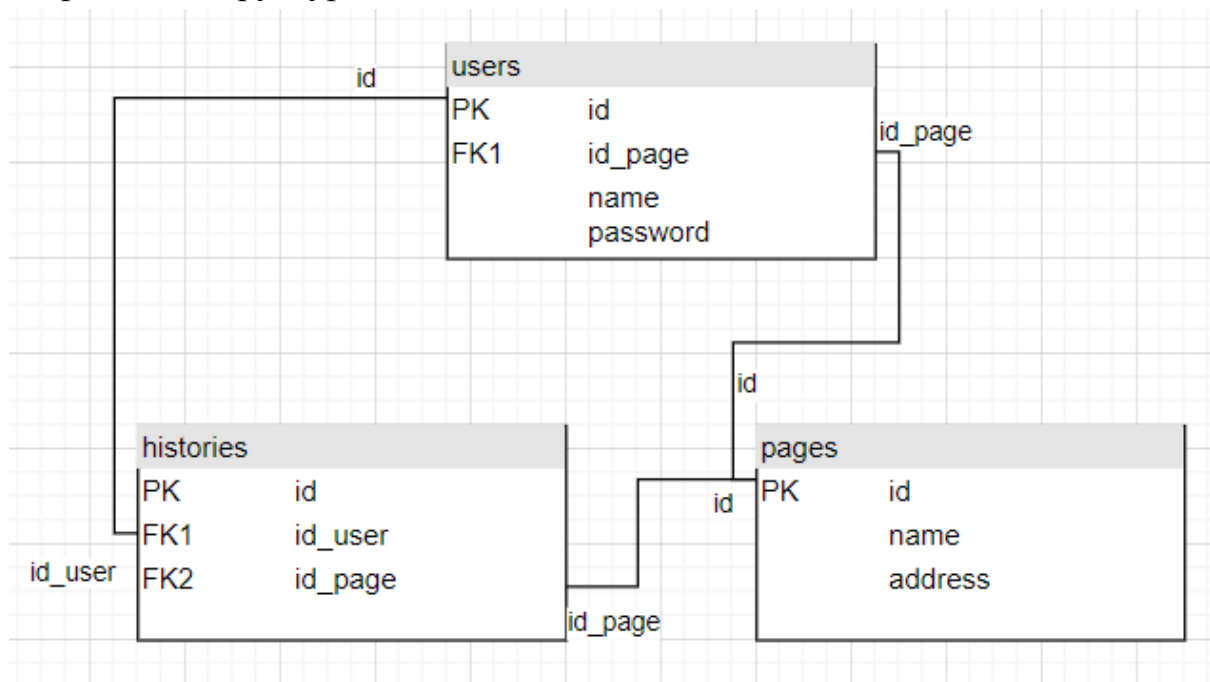


Рис.3.1 Зображення структури бази даних

Вихідні коди класів системи:

```
Python HTTPRequestHandler.py × JSInterpreter.py
1 import requests
2
3
4   Павло Kostenko *
5 class HTTPRequestHandler:
6     #змінна для адресного рядка
7     addressline = ''
8
9     #основний метод для реквестів
10    Павло Kostenko
11    def send_get_request(self, url):
12        response = requests.get(url)
13        return response.text
```

```
Python HTTPResponseHandler.py × JSInterpr
1   Павло Kostenko *
2 class HTTPResponseHandler:
3     addressline = ''
4
5     #основний метод для респонс
6    Павло Kostenko
7    def send_get_response(self):
8        pass
```



```

Pavlo Kostenko *
1 class CSSRenderer:
2     addressline = ''
3
4     #метод для обробки CSS
Pavlo Kostenko
5     def render_css(self, css_content):
6         pass
7

```

```

Pavlo Kostenko *
class Errors:
    addressline = ''

    Pavlo Kostenko *
    def error_404(self):
        pass

    new *
    def error_502(self):
        pass

    new *
    def error_503(self):
        pass
#методи для обробки помилок

```

```
new *
class History:
    id = ''
    id_user = ''
    id_page = ''

new *
def __init__(self, id, id_user, id_page):
    self.id = id
    self.id_user = id_user
    self.id_page = id_page

new *
def get_id(self):
    return self.id

new *
def set_id_user(self, id_user):
    self.id_user = id_user

new *
def get_id_user(self):
    return self.id_user

#гетери та сетери, будуть дописані
```

```
from bs4 import BeautifulSoup
```

👤 Pavlo Kostenko *

```
class HTMLParser:
```

```
    addressline = ''
```

💡 #метод для парсингу HTML вмісту

👤 Pavlo Kostenko

```
    def parse(self, html_content):
```

```
        soup = BeautifulSoup(html_content, 'html.parser')
```

```
        pass
```

```
1 from PIL import Image
```

```
2 from io import BytesIO
```

```
3 import requests
```

```
4
```

```
5
```

👤 Pavlo Kostenko *

```
6 class ImageLoader:
```

```
7     addressline = ''
```

```
8
```

9 💡 #метод для завантаження зображень на сторінку

👤 Pavlo Kostenko

```
10    def load_image(self, image_url):
```

```
11        response = requests.get(image_url)
```

```
12        img = Image.open(BytesIO(response.content))
```

```
13        img.show()
```

```

Pavlo Kostenko *
1 class JSInterpreter:
2     addressline = ''
3
4     #Метод для відображення(запуску) JS коду
Pavlo Kostenko
5     def execute_js(self, js_code):
6         pass

```

```

new *
class Page:
    id = ''
    name = ''
    address = ''

    new *
    def __init__(self, id, name, address):
        self.id = id
        self.name = name
        self.address = address
    #Також будуть гетери та сетери


```

```

new *
class User:
    id = ''
    name = ''
    password = ''
    page_id = ''

new *
def __init__(self, id, name, password, page_id):
    self.id = id
    self.name = name
    self.password = password
    self.page_id = page_id

```

 #Також будуть гетери та сетери

Repository.py ×

```

1  import psycopg2
2
3
4  2 usages
5  class Repository:
6
7      def exec(self, dbname, user, password, host, port, query):
8          conn = psycopg2.connect(dbname=dbname, user=user, password=password, host=host, port=port)
9          conn.autocommit = True
10         cursor = conn.cursor()
11         cursor.execute(query)
12         conn.commit()
13         conn.close()
14
15     1 usage
16     def fetch(self, dbname, user, password, host, port):
17         conn = psycopg2.connect(dbname=dbname, user=user, password=password, host=host, port=port)
18         conn.autocommit = True
19         cursor = conn.cursor()
20         cursor.execute('SELECT * from histories')
21         result = cursor.fetchall()
22         conn.commit()
23         conn.close()
24         print(result)

```

Висновок: на цій лабораторній роботі я ознайомився з теоритичними відомостями, проаналізував тему, склав до неї діаграму прецедентів, діаграму класів системи, а також зображення структури бази даних, написав вихідні початкові коди класів системи.