**Зміст**

[1 МОВА ПРОГРАМУВАННЯ «РНР»: ХАРАКТЕРНІ ОСОБЛИВОСТІ ТА СПОСОБИ ПРЕДСТАВЛЕННЯ ВХІДНИХ ДАНИХ 6](#_Toc504034814)

[1.2 Обґрунтування вибору програмного забезпечення для вирішення поставленої задачі 6](#_Toc504034815)

[2 ПРОЕКТУВАННЯ БАЗИ ДАНИХ 8](#_Toc504034816)

[3 РОЗРОБКА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ 9](#_Toc504034817)

[3.1 Драйвер для роботи з базою даних 9](#_Toc504034818)

[3.2 Користувацький інтерфейс 14](#_Toc504034819)

[3.3 CGI-сервер 21](#_Toc504034820)

[Висновки 22](#_Toc504034821)

[Список використаних джерел 23](#_Toc504034822)

Додаток А Код програми-драйвера для бази даних (файл engine.py)………………

Додаток Б Код програми головного файлу користувацького інтерфейсу main.py..

Додаток В Код програми додавання операторів та конструкцій мови РНР (файл add\_operator.py)………………………………………………………………………..

Додаток Г Код програми змінення записів в базі даних (файл edit\_operator.py)….

Додаток Д Код каскадної таблиці стилів style.css………………………………….

Додаток Е Скріншоти користувацького інтерфейсу………………………………..

# МОВА ПРОГРАМУВАННЯ «РНР»: ХАРАКТЕРНІ ОСОБЛИВОСТІ ТА СПОСОБИ ПРЕДСТАВЛЕННЯ ВХІДНИХ ДАНИХ

* 1. **Мова програмування РНР**

Мета курсового проекту – спроектувати реляційну базу даних довідкового типу для операторів та конструкцій мови програмування PHP.

PHP (англ. PHP: Hypertext Preprocessor — PHP: гіпертекстовий препроцесор), попередня назва: Personal Home Page Tools — скриптова мова програмування, була створена для генерації HTML-сторінок на стороні веб-сервера. PHP є однією з найпоширеніших мов, що використовуються у сфері веб-розробок (разом із Java, .NET, Perl, Python, Ruby). Дана мова підтримується переважною більшістю хостинг-провайдерів і вона з відкритими сирцевими кодами та відноситься до відкритого програмного забезпечення.

PHP — мова, у код якої можна вбудовувати безпосередньо html-код сторінок, які, у свою чергу, коректно оброблюватимуться PHP-інтерпретатором. Обробник PHP просто починає виконувати код після відкриваючого тегу (<?php) і продовжує виконання до того моменту, поки не зустріне закриваючий тег (?>).

Також дана мова містить велику кількість вже реалізованих вбудованих функцій, що полегшує роботу при розробці і робить мову більш доступною для широких мас.

## Обґрунтування вибору програмного забезпечення для вирішення поставленої задачі

1.2.1 CGI-скрипти

Для вирішення поставленої задачі було взято за основу CGI-сценарії, та скрипти мовою Python. Інтерфейсом обміну інформацією між користувачем та базою даних є веб-інтерфейс.

CGI (від англійського Common Gateway Interface — «загальний інтерфейс шлюзу») — стандарт інтерфейсу, який використовується для організації взаємодії програми веб-сервера із зовнішньою програмою. Програму, яка працює за таким інтерфейсом спільно з веб-сервером, прийнято називати шлюзом. Також вживаються терміни «скрипт» (сценарій) або «CGI-програма».

Сам інтерфейс розроблений таким чином, щоб можна було використовувати будь-яку мову програмування. Для обміну даними використовуються стандартні інтерфейси вводу/виводу.

1.2.2 СУБД SQLite3

Дана задача вирішується на локальному комп’ютері, томе не потребує повноцінної бази даних типу MySQL, PostgreSQL та ін., тому, було вирішено використати компактну реляційну вбудовану базу даних SQLite3, що не потребує постійної роботи в окремому процесі а взаємодіє з програмою у вигляді бібліотеки.

# ПРОЕКТУВАННЯ БАЗИ ДАНИХ

Для вирішення поставленої задачі було прийнято використати три таблиці, в яких зберігатимуться наступні дані:

* Оператори мови РНР (таблиця *operators*)
* Приклади використання операторів мови РНР (*example*)
* Опис до них (*description*)

Всі таблиці об’єднані зв’язками «один до одного» за допомогою додаткових полів «ID». Дані поля мають тип даних INTEGER, та є первинними ключами, а також автоматично нумеруватимуться (AUTOINCREMENT).

В таблиці *operators* існує поле “Operator”, тип якого VARCHAR, і призначене воно для зберігання текстової інформації, а саме, операторів мови РНР.

В таблиці *example* існує поле “text”, типу VARCHAR, і призначене воно для зберігання текстової інформації, а саме прикладів використання операторів мови програмування РНР.

В таблиці *description* існує поле “text”, типу VARCHAR, призначене для зберігання текстових описів до операторів (*operators*.Operator) та прикладів їхнього використання (*example*.text)

На Рисунку 2.1 представлено UML-діаграму структури бази даних.

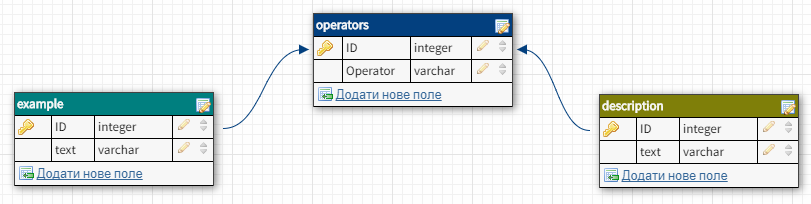


Рисунок 2.1 – UML-діаграма структури бази даних

# РОЗРОБКА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

## Драйвер для роботи з базою даних

Спочатку нам потрібно створити клас з відповідними методами для роботи з базою даних. Він повинен містити в собі методи підключення до бази даних, створення таблиць у випадку, якщо вони ще не створені, додавання нових записів, видалення, редагування, їх вибірки: всіх записів або за умовою.

Такий клас ми створили, його назва «DataBase» і розміщений він у файлі engine.py.

Перш за все, при ініціалізації об’єкту класу потрібно підключитись до бази даних (так як в даному курсовому проекті використовується Sqlite3, при підключенні файл бази даних за його відсутності автоматично створюється) та створити потрібні таблиці з відповідними полями відповідних типів. За все це відповідає метод-ініціалізатор класу для роботи з базою даних (Рисунок 3.1).

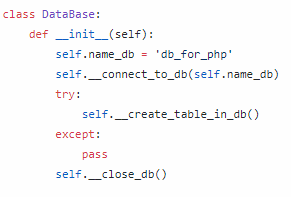


Рисунок 3.1 - Фрагмент коду класу DataBase з методом \_\_init\_\_

Даний метод містить один атрибут name\_db, в якому зберігається назва файлу бази даних. Наступний рядок містить в собі виклик методу \_\_connect\_to\_db, який в якості аргументу приймає атрибут назви бази даних, до якої потрібно підключитись. З міркувань безпеки, даний метод (як і багато наступних) названо з двома символами підкреслення на початку («\_\_»), так як це єдиний спосіб інкапсулювати внутрішні методи класу від зовнішнього використання в мові Python.

В наступних чотирьох рядках викликається метод створення таблиць в базі даних. Сам виклик методу загорнуто в програмну конструкцію try-except, так-званий відловлювач помилок. Зроблено це з метою уникнути припинення роботи CGI-скрипта у випадку виникнення конфліктів при підключенні до бази даних. Останній рядок коду закриває робочу сесію з базою даних. Постійне правильне завершення робочої сесії призводить до економії оперативної пам’яті за рахунок зменшення кількості використовуваних потоків.

Самі ж методи підключення до бази даних, створення таблиць та відключення від неї зображені на Рисунку 3.2.

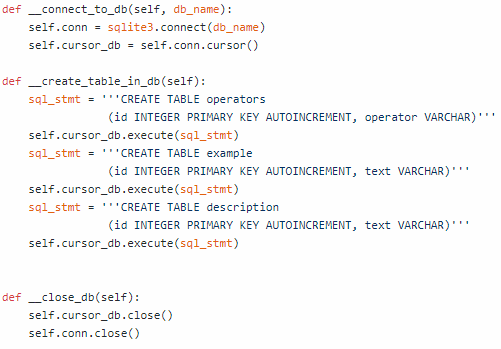


Рисунок 3.2 – Основні методи роботи з базою даних

Для додавання нових записів в базу даних було створено чотири методи, три з яких працюють з кожною таблицею окремо, а один – загальний для всіх. В ньому викликаються три попередні методи і відловлюються помилки в разі виникнення проблеми при додаванні записів. При виникненні помилок в головну програму повертається значення «1», що символізує про некоректну роботу програми при додаванні нових записів. Якщо ж все добре, повертається значення «0» і перед користувачем виводиться повідомлення про успішно-виконану дію. Фрагменти коду даних методів представленні на Рисунку 3.3 та Рисунку 3.4.

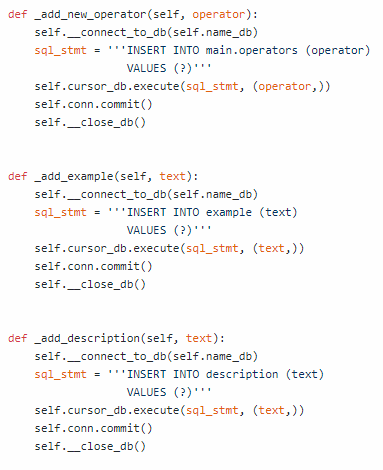


Рисунок 3.3 – Методи для роботи з кожною таблицею окремо

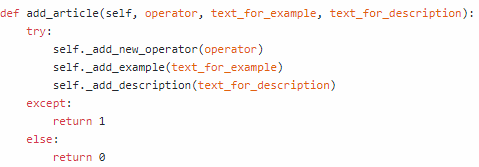


Рисунок 3.4 – Загальний метод додавання записів в базу даних

Наступні методи створені для вибірки інформації з бази даних. За допомогою метода get\_all\_articles ми отримуємо всі записи з таблиці у вигляді списку, що містить в собі списки-записи (Рисунок 3.5).

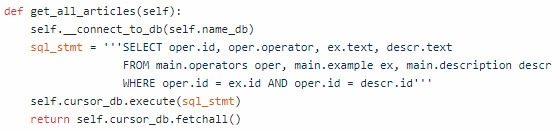


Рисунок 3.5 – Метод вибірки всіх записів з таблиці

Для отримання одного потрібного запису існує метод get\_article, що приймає в якості аргумента id запису і повертає список значень: оператор, приклад використання та текст опису (Рисунок 3.6)

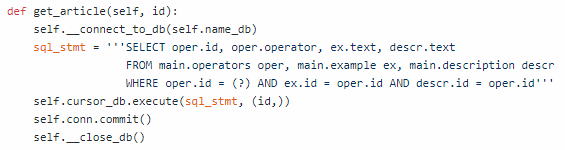


Рисунок 3.6 – Метод вибірки одного потрібного запису з таблиці

Для вибірки записів за умовою використовується метод search\_operators, в якому реалізовано конструкцію запиту SELECT з використанням оператору LIKE в умові (Рисунок 3.7).

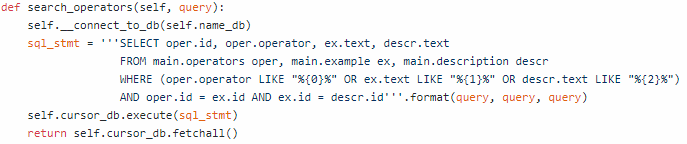


Рисунок 3.7 – Метод вибірки даних за умовою

Для змінення та видалення запитів створено два методи:

* edit\_article (Рисунок 3.8), що приймає вхідні аргументи «operator», «text\_for\_example», «text\_for\_description»
* delete\_article (Рисунок 3.9), що приймає id запису в базі даних.

В першому методі використовується SQL оператор UPDATE, що дозволяє обновити наявний запис в базі даних, в другому відповідно використовується оператор DELETE.

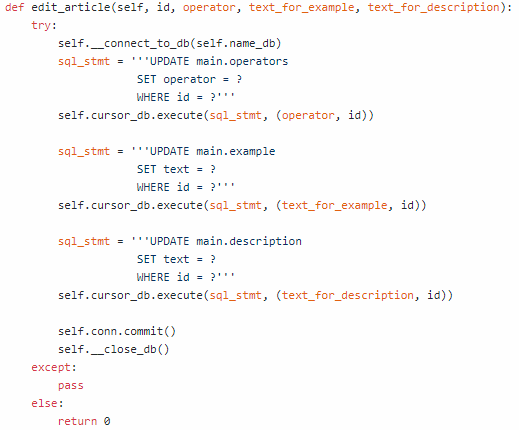


Рисунок 3.8 – Метод зміни запису в базі даних

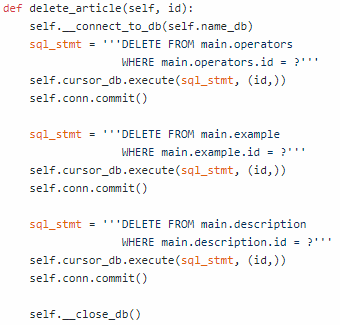


Рисунок 3.9 – Метод видалення запису з бази даних

## Користувацький інтерфейс

3.2.1 main.py

Main.py – це основний файл користувацького інтерфейсу. За його допомогою виводяться всі існуючі записи з бази даних і саме він містить кнопки для редагування та видалення даних (Додаток Е.1, Рисунок Е.1). Також при пошуку знайдені записи виводяться в цей файл.

Сам вигляд головної сторінки написано за допомогою мови веб-розмітки HTML та каскадної таблиці стилів CSS. Розміщені ці файли в папці /html/ та /html/css/, та називаються pattern.html і style.css. Перший файл зчитується двома рядками коду в main.py і зчитаний текст присвоюється змінній «pattern» (Рисунок 3.10)



Рисунок 3.10 – Зчитування файлу-шаблону вигляду головної сторінки

Файл pattern.html містить в собі підключення таблиці стилів, а також розділені блоки з вказаними місцями для форматного виводу даних (Рисунок 3.11)



Рисунок 3.11 – Pattern.html

Вивід і форматна заміна блоків проводиться в кінці коду, коли все сформовано. На Рисунку 3.12 зображено два рядки коду: в першому ми виводимо повідомлення браузеру що це – html код і його потрібно інтерпретувати а не виводити у вигляді тексту, в другому рядку ми виводимо раніше зчитаний шаблон (зберігається в змінній «pattern»), в якому методом «format» замінюємо вказані блоки з pattern.html (Рисунок 3.11, блоки які потрібно замінити виділено зеленим маркером) на потрібний нам текст. Потрібний нам текст міститься в змінних «header», «out» та «notific». Перша змінна містить в собі назву сторінки для відображення в «title» pattern.html. Третя змінна – notific (від англійського слова notification - сповіщення), містить в собі додаткові повідомлення. Використовується при пошуку, додаванні та редагуванні даних і виводить сповіщення про стан дії що відбулась.



Рисунок 3.12 – Форматна заміна блоків та вивід даних

Змінна out містить в собі основну інформацію, яку потрібно вивести користувачеві (Рисунок 3.13). Спочатку ми ініціалізуємо дану змінну статичними даними, такими як навігаційні посилання, форма пошуку і верх таблиці, після цього в циклі «for» ми доповнюємо змінну даними для виводу окремих записів.

Дані для доповнення змінної out ми беремо зі списку articles, який отримуємо з раніше створеного об’єкту класу DataBase – wall. Перед тим як отримати дані, ми можемо видалити запис з бази даних, і отримати вже обновлений список, або ввести пошуковий запит і отримати замість всіх записів тільки ті які відповідають запиту, або не отримати нічого. Якщо не виконується ні перший ні другий сценарій, то викликається метод get\_all\_articles і ми отримуємо список з усіма записами (Рисунок 3.14)



Рисунок 3.13 – Ініціалізація та доповнення змінної «out»

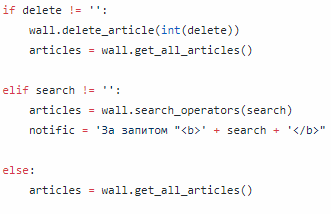


Рисунок 3.14 – Отримання даних для виводу на сторінку

Приклад виведення всіх записів зображено на Рисунку 3.15.

Приклад виведення пошукового запиту зображено на Рисунку 3.16

Приклад виведення всіх записів за винятком видаленого запису «if» зображено на Рисунку 3.17

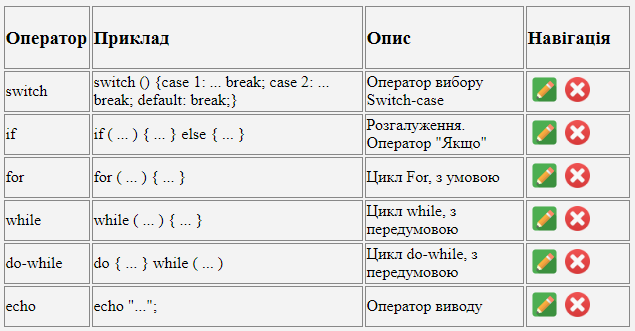


Рисунок 3.15 – Виведення на сторінку всіх записів

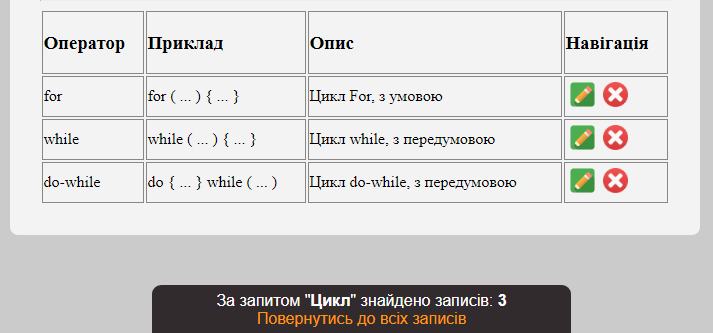


Рисунок 3.16 – Виведення пошукового запиту

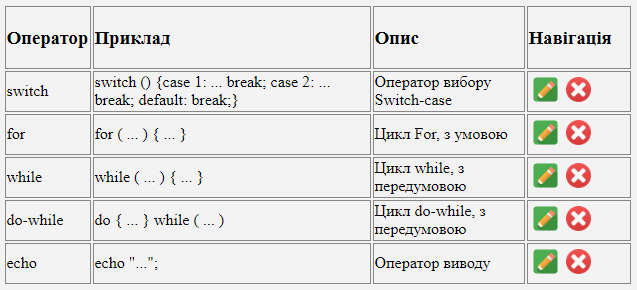


Рисунок 3.17 – Виведення всіх записів за винятком видаленого «if»

Самі ж дані для пошуку та видалення записів ми отримуємо з GET-запитів, що приходять на дану ж сторінку. Форма пошуку відправляє запит вигляду «/main.py?search=<пошуковий запит>», і фраза <пошуковий запит> заноситься в змінну search, а дальше відправляється в метод класу DataBase – search-operators, звідки повертається вибірка даних, або порожній масив, що присвоюється змінній articles (Рисунок 3.14).

Видалення відбувається за допомогою натиску кнопки «Х» в блоці «Навігація» на сторінці. При натиску на цю ж сторінку відправляється GET-запит вигляду «main.py?delete=<number>», де ціле число <number> присвоюється змінній delete, а потім передається в метод класу DataBase – delete\_article, де відбувається видалення потрібного запису (Рисунок 3.14). Наступним рядком ми отримуємо обновлений список записів і присвоюємо його змінній articles.

3.2.2 add\_operator.py

Даний файл дуже схожий до попереднього. В його коді теж використовується зчитування pattern.html, форматний вивід в блоки файлу потрібних даних, також ініціалізований об’єкт класу DataBase – wall. Єдина відмінність даного файлу від попереднього – відсутність виводу будь-яких даних з бази даних та присутність форми для додавання нових записів (Додаток Е.1, Рисунок Е.2).



Рисунок 3.18 – Змінна out в файлі add\_operator.py

При заповненні форм дані відправляються на цей же файл GET-запитом у вигляді «/add\_operator.py?add\_operator=<operator>&add\_example=<example&

add\_desc=<description>», де присвоюються змінним «add\_operator», «add\_example» та «add\_desc», які в свою чергу передаються у вигляді аргументів в метод класу DataBase – add\_article (Рисунок 3.19). При успішному додаванні запису на екран користувача виведеться сповіщення «Оператор успішно додано» (Рисунок 3.20), в противному випадку сповіщення матиме вигляд «Помилка, спробуйте ще раз».

Також після кожного сповіщення виводиться посилання на головну сторінку, де користувач зможе переглянути зміни.

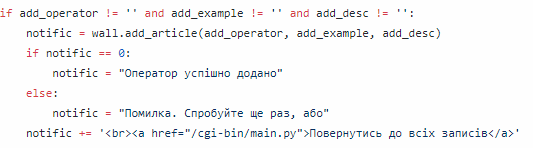


Рисунок 3.19 – Додавання нового запису в базу даних

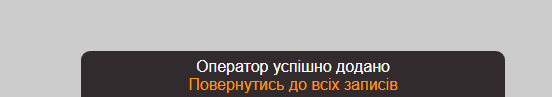


Рисунок 3.20 – Сповіщення про успішне додавання нового оператора

3.2.3 edit\_operator.py

Даний файл майже нічим не відрізняється від попереднього, він містить ту саму форму, тільки в даному випадку форма зразу заповнюється потрібними даними, які потрібно редагувати (Рисунок 3.21) (Вигляд сторінки – Додаток Е.2, Рисунок Е.3). Самі ж дані файлу передаються GET-запитому у вигляді:

«edit\_operator.py?id=<id>&operator=<operatr>&example=<exmple>&desc=<desc>»

і присвоюються змінним «old\_operator», «old\_example» та «old\_desc». Потім зі змінні заносяться в атрибут HTML-тегу Input – value (Рисунок 3.22)

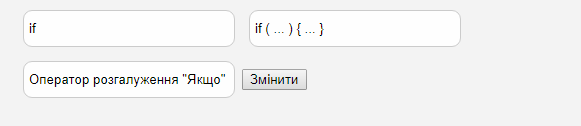


Рисунок 3.21 – Форма редагування запису

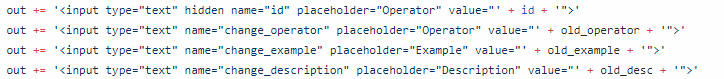


Рисунок 3.22 – Занесення даних які потрібно змінити в форми

Після змін нові дані відправляються на цю ж сторінку GET-запитом у вигляді:

«edit\_operator.py?id=<id>&change\_operator=<operator>&change\_example=<example>&change\_description=<desc>», де присвоюються змінним «change\_operator», «change\_example» та «change\_desc», які відправляються в якості аргументів в метод класу DataBase – edit\_article (Рисунок 3.23)

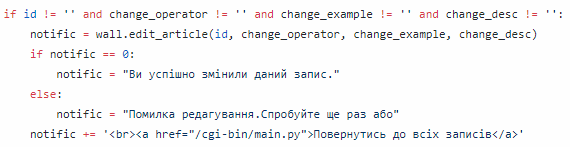


Рисунок 3.23 – Зміна запису в базі даних

У випадку успішного виконання операції, користувачу виводиться сповіщення (Рисунок 3.24).

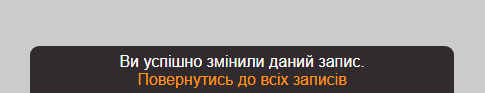


Рисунок 3.24 – Сповіщення про успішну зміну запису в базі даних

## CGI-сервер

Основним файлом, без якого не запрацює жоден вищеописаний є звичайно файл CGI-сервера – serv.py. Він містить в собі основні команди для запуску сервера. Весь код даного файлу зображено на Рисунку 3.25. В даному коді є змінна server\_address, в яку ми заносимо ІР адресу та порт, за яким буде доступний результат виконання скриптів. Якщо поле ІР-адреси залишити пустим, то встановиться стандартна адреса на локальний хост даного комп’ютера (127.0.0.1).

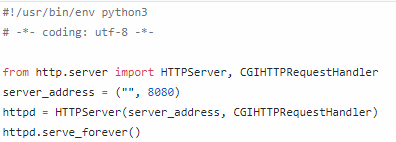


Рисунок 3.25 – Код запуску CGI-сервера

# Висновки

Метою даної курсової роботи було спроектувати базу даних потрібного типу для збереження інформації про оператори та конструкції мови програмування РНР.

Для вирішення поставленої задачі було використано комплексний підхід до проблеми і створено систему роботи з записами операторів та конструкцій даної мови програмування. Вирішено ряд проблем, серед яких автоматичне створення таблиць в базі даних за їх відсутності, а також автоматичне створення самої бази даних у випадку її не існування. Створено клас та методи для роботи з базою даних, а також користувацький інтерфейс за допомогою якого можна додавати, переглядати, редагувати та видаляти записи. Також додано можливість пошуку по всіх полях бази даних текстових полях бази даних.

Даний варіант виконання програми можна виставляти на готовий сервер і таким чином створити загальнодоступний у використанні довідник для мови програмування РНР по її операторах та умовних конструкціях.

# Список використаних джерел

1. Г. Россум, Ф.Л.Дж. Дрейк, Д.С. Откидач. Язык программирования Python - 2001 — 454 c.
2. Shishir Gundavaram. CGI Programming on the World Wide Web - O'Reilly Media: 1996 —450 с.
3. Tutorials Point (I) Pvt. Ltd. SQLite tutorials point.: 2017.-191 с.
4. PHP // Вікіпедія – вільна енциклопедія– 2017 [Електронний ресурс]. Режим доступу: <https://uk.wikipedia.org/wiki/PHP>, вільний – Назва з екрану