Міністерство освіти і науки України

Економіко-технологічний інститут ім. Роберта Ельворті

Кафедра інформаційних технологій

КУРСОВА РОБОТА

ПОЯСНЮВАЛЬНА ЗАПИСКА

з дисципліни «Об’єктно-орієнтоване програмування»

«Бібліотека класів для розробки

родового (Генеалогічного) дерева»

Керівник:

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Студент:

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Комісія:

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Кропивницький 2024

Економіко-Технологічний інститут

ім. Роберта Ельворті

Кафедра *інформаційних технологій*

Рівень вищої освіти *перший (бакалаврський)*

Дисципліна *Об’єктно-орієнтоване програмування*

Спеціальність *122 Комп’ютерні науки*

Освітня програма: *Комп’ютерні науки*

Курс *4*

Група *КН-20*

Семестр *8*

ЗАВДАННЯ на курсовий проект студента

Васильєвої Мар’яни Вікторівни

1 Тема проекту: Спроектувати та реалізувати бібліотеку класів для розробки родового (Генеалогічного) дерева. Паспортні дані членів деякого родового клану; посилання на дітей (або на батьків). Пошук всіх нащадків або всіх предків для вказаної особи. Ієрархічне відображення генеалогічного дерева обраної людини.

2 Термін здачі студентом закінченого проекту: “16” квітня 2024 р.

3 Вихідні дані до проекту: *Специфікація програми, методичні*

*вказівки до виконання курсової роботи*

4 Зміст розрахунково-пояснювальної записки:

*Вступ, специфікація програми, проектна специфікація, інструкція*

*користувача, висновки\_*\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

5 Перелік графічного матеріалу:

*Схема об’єктної моделі, алгоритми, приклади екранних форм, знімки екрану*

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

РЕФЕРАТ

Пояснювальна записка до курсової роботи: 44 с., 10 рис., 3 табл., 2

додатки, 7 джерел.

ЛОКАЛЬНИЙ СЕРВЕР РОЗРОБКИ, MVC, SQL, PHP, СУБД SQLITE, COMPOSER, HTML, CSS, ІНКАПСУЛЯЦІЯ, КЛАС, ОБ'ЄКТ, МЕТОД, ВЛАСТИВІСТЬ, КОНСТРУКТОР, SINGLETON, FACTORY.

Метою роботи є розробка програми «Family Tree» на засадах об'єктно-орієнтованого програмування. Методи розробки базуються на використанні локального середовища розробки PHP, мови програмування PHP та принципів ООП.

В результаті отриманий веб-додаток під назвою «Family Tree», що дозволяє зберігати, опрацьовувати інформацію про осіб та їх взаємозв’язки в контексті створення генеалогічного дерева.

Реалізовано можливості користувацького інтерфейсу:

- додавати, редагувати, видаляти осіб в таблицю “People”;

- додавати, редагувати, видаляти інформаційні записи на головну сторінку.

Реалізовано можливості вебдодатку:

- формувати родове дерево для обраної особи;

- виводити загальну інформацію по всім особам у вигляді таблиці.

Реалізовано додаткові серверні можливості

- консольні команди для демонстрації роботи проєкту.

ЗМІСТ

|  |  |  |
| --- | --- | --- |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

ВСТУП

У сучасному світі інформаціних технологій, зокрема в веб-розробці (програмування веб додатків), особливо значущим стає вміння застосувати об’єктно-орієнтовані принципи для створення масштабованих і легко модифікованих програмних рішень. Дана курсова робота, виконана в рамках дисципліни "Об’єктно-орієнтоване програмування", має на меті не лише закріпити базові поняття і принципи об’єктно-орієнтованого програмування, а й демонструє їх практичне застосування через розробку спеціалізованої бібліотеки класів.

Головна задача даної курсової роботи — спроектувати та реалізувати бібліотеку класів для розробки генеалогічного (родового) дерева, здатну зберігати паспортні дані членів деякого родового клану, включно з посиланнями на дітей та батьків. Окрім того, бібліотека дозволить здійснювати пошук всіх нащадків або всіх предків для вказаної особи та ієрархічно відображати генеалогічне дерево обраної людини.

Реалізація проекту здійснена з використанням мови програмування PHP, шаблонізатора Blade та системи управління базами даних SQLite, що забезпечує гнучкість та зручність управління даними. Застосування цих технологій дозволяє ефективно інтегрувати об’єктно-орієнтовані підходи у веб-розробку, створюючи міцну основу для подальших інноваційних розробок у цій галузі.

Таким чином, дана курсова робота відображає не тільки теоретичне осмислення об’єктно-орієнтованих принципів, а й їх практичне застосування в реалізації конкретного програмного продукту, що має значний потенціал застосування у сфері родових досліджень та управління генеалогічною інформацією.

РОЗДІЛ 1. Специфікація програми

1.1. Визначення робочої середи розробки та обрання інструментів розробки.

Програмна реалізації бібліотеки класів для обраної теми (проєктування та реалізація бібліотеки класів для розробки родового дерева) необхідно чітко визначити робочу середу розробки та обрати відповідні інструменти.

Для розробки бібліотеки класів, призначеної для створення родового дерева, вибрано наступну технологічну стеку:

**Мова програмування:** Обрано PHP через її широкі можливості у веб-розробці та зручність інтеграції з різними базами даних та шаблонізаторами (template engine).

**Інтегроване середовище розробки (IDE)**: Вибрано JetBrains PhpStorm, оскільки це одне з найбільш потужних та зручних середовищ для роботи з PHP. PhpStorm надає розширену підтримку синтаксису PHP, інтеграцію з системами керування версіями, засоби для дебагінгу та рефакторингу коду, що забезпечує високу продуктивність розробки.

**Пакетний менеджер**: Використовуємо Composer для керування залежностями PHP проєкту. Composer дозволяє легко встановлювати, оновлювати та керувати бібліотеками, що використовуються у проєкті, що є важливим для підтримки коду в актуальному стані.

**Система контролю версій**: Для керування версіями коду обрано Git. Git — це найпопулярніша система контролю версій, яка дозволяє розробникам з легкістю відстежувати та управляти змінами в коді проєкту. Використання Git сприяє кращій координації роботи команди, дозволяє здійснювати паралельну розробку, швидко відкатувати зміни та інтегрувати роботу з віддаленими репозиторіями, такими як GitHub або Bitbucket. Інтеграція Git з PhpStorm також підвищує продуктивність роботи, оскільки дозволяє виконувати всі основні операції безпосередньо з IDE.

Це забезпечує високу гнучкість та ефективність у процесі розробки, що є критично важливим для успішного ведення сучасних IT-проектів.

**Локальний сервер розробки (вбудований вебсервер РНР)** - це середовище для розробки вебдодатків на локальному комп'ютері. Локальний сервер дозволяє запускати вебсервер, базу даних, та інші сервіси для розробки та тестування вебдодатків. Локальний сервер дозволяє розробникам працювати над вебдодатками без доступу до Інтернету та забезпечує швидкий доступ до ресурсів.

**База даних**: Використовується SQLite через її простоту інтеграції та використання без необхідності налаштування складних серверів. SQLite ідеально підходить для розробки додатків середнього та малого розміру, де не передбачається великий обсяг даних або високе навантаження.

**Інтерфейс користувача**: Інтерфейс реалізований за допомогою шаблонізатора Blade та фреймворка Bootstrap. Blade надає зручні інструменти для створення динамічного контенту, дозволяючи ефективно управляти HTML шаблонами і сприяє швидкій розробці. Bootstrap використовується для швидкого та адаптивного дизайну веб-сторінок, що дозволяє забезпечити гарний вигляд інтерфейсу на будь-яких пристроях.

Ця комбінація технологій дозволяє ефективно проєктувати та реалізовувати бібліотеку класів для створення родового дерева, забезпечуючи високу продуктивність розробки та зручність подальшої підтримки проєкту.

1.2. Обрання архітектурного шаблону.

Для проєктування та реалізації бібліотеки класів, що спрямована на розробку родового дерева, ключовим є вибір архітектурного шаблону, який визначатиме структуру та методи взаємодії компонентів системи. Враховуючи специфіку проєкту, обрано наступні архітектурні рішення:

**Модель-вид-контролер (MVC)**: MVC є стандартним шаблоном проектування, що розділяє додаток на три основні компоненти: модель (дані), вид (інтерфейс користувача), та контролер (логіка обробки). Цей шаблон сприяє чіткому розділенню логіки та інтерфейсу, що полегшує тестування, розширення та підтримку коду. Для родового дерева, MVC дозволить ефективно управляти даними користувачів, їхніми взаємозв'язками та представленням цієї інформації.

**Singleton**: Патерн Singleton застосовується для забезпечення існування лише одного екземпляра класу в усьому додатку. Це корисно для управління глобальними ресурсами, такими як з'єднання з базою даних. Використання Singleton для SQLite бази даних забезпечить, що з'єднання буде створено один раз та використовуватиметься повсюдно в рамках проєкту.

**Factory**: Патерн Factory використовується для створення об'єктів без специфікації точних класів об'єктів, які будуть створені. Цей підхід ідеально підходить для створення об'єктів класів у нашій бібліотеці, де може бути кілька типів елементів родового дерева, таких як користувачі, зв'язки, події тощо. Factory забезпечить гнучкість у створенні цих об'єктів, дозволяючи легко впроваджувати нові типи.

**Принципи об'єктно-орієнтованого програмування (ООП)**: Основні принципи ООП, такі як інкапсуляція, наслідування, поліморфізм і абстракція, відіграватимуть центральну роль у проектуванні та реалізації нашої бібліотеки. Інкапсуляція забезпечує, що дані класу захищені та сховані від зовнішнього доступу, наслідування дозволяє створювати нові класи на базі вже існуючих, поліморфізм сприяє використанню одного інтерфейсу для різних базових форм даних, а абстракція дозволяє мінімізувати складність, виділяючи ключові характеристики об'єкта.

Використання цих шаблонів та принципів забезпечить розробку гнучкої, масштабованої та легко підтримуваної системи для родового дерева.

1.3. Програмна реалізація бібліотеки класів

Для розробки бібліотеки класів проекту родового дерева використовується наступна конфігурація у файлі **composer.json**, що забезпечує керування залежностями та автозавантаження класів (рис 1.1).



Рисунок 1.1 - конфігураційний файл composer.json

**Автозавантаження (Autoload**): Використання стандарту PSR-4 для автозавантаження дозволяє організувати структуру директорій та простори імен у відповідності до стандартів сучасної PHP-розробки. Всі класи бібліотеки розміщені у директорії src/ та належать до простору імен Mariana\FamilyTree\, що сприяє чистоті коду та легкості його управління.

Автозавантаження класів у PHP проекті, який використовує Composer як інструмент для управління залежностями, є ключовим компонентом для ефективної розробки та підтримки чистоти коду.

**Підключення Composer автозавантажувача:**

Для ініціації автозавантажувача, який Composer генерує, перш ніж використовувати будь-які класи, потрібно підключити файл autoload.php, який знаходиться у директорії vendor. Це робиться шляхом додавання наступного рядка коду на початку вашого скрипта або конфігураційного файлу:

**require '../vendor/autoload.php';**

Коли використовуємо клас, наприклад Mariana\FamilyTree\Tree, PHP автоматично знаходить і завантажує відповідний файл класу з директорії src/, використовуючи налаштування, визначені в Composer. Це значно спрощує управління залежностями і організацію коду, знижуючи ризик помилок та забезпечуючи більшу ефективність розробки.

**Залежності (Require)**: Проєкт вимагає PHP версії 8.3 або вище, забезпечуючи сумісність із сучасними функціональними можливостями мови. Включення розширення PDO дозволяє взаємодіяти з базами даних через об'єктно-орієнтований підхід. Бібліотека nikic/fast-route використовується для ефективного маршрутизування запитів у веб-додатку. Шаблонізатор jenssegers/blade дозволяє створювати гнучкі та зручні шаблони інтерфейсу. fakerphp/faker використовується для генерації випадкових даних під час розробки та тестування (рис 1.2).

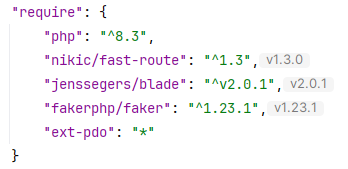


Рисунок 1.2 - залежності проєкту.

Ця конфігурація відображає використання сучасних інструментів та практик у розробці веб-додатків, спрямованих на високу продуктивність, масштабованість та легкість у підтримці.

**Опис Класу Application**

Клас Application у системі родинного дерева є центральним компонентом для управління ініціалізацією та взаємодією з різними частинами програми. Використання патерну "Singleton" гарантує, що в програмі існує лише один екземпляр цього класу, що є стандартним підходом для реалізації центрального керування компонентами (рис. 1.3).

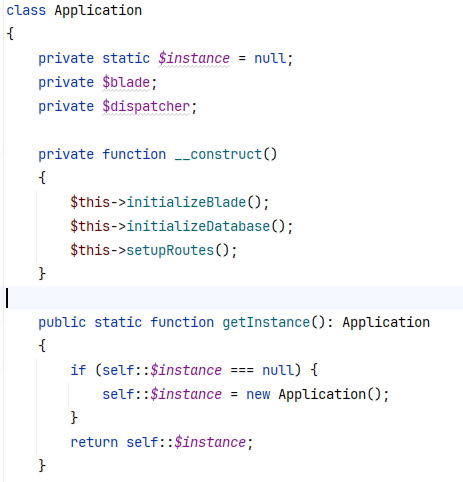


Рисунок 1.3 - Використання патерну "Singleton"

**Ініціалізація системи шаблонів Blade:** Використання шаблонізатора Blade для рендерингу HTML сторінок на основі PHP коду дозволяє легко інтегрувати динамічний контент у веб-сторінки:

private function initializeBlade(): void

{

$this->blade = new Blade('/../cache');

}

**Налаштування маршрутизації:** Через бібліотеку FastRoute, клас управляє маршрутами HTTP-запитів, направляючи їх до відповідних контролерів (рис 1.4)



Рисунок 1.4 - визначення маршрутів

**Ініціалізація бази даних:** Завдяки Singleton патерну інстанс бази даних ініціалізується одноразово та використовується у всіх частинах програми:

private function initializeDatabase(): void

{

Database::getInstance(); // Singleton Database

}

**Методи класу Application**:

**getInstance**(): Метод Singleton для отримання інстансу класу.

**run**(): Основний метод, який запускає веб-додаток, обробляючи HTTP запити та направляючи їх до відповідних маршрутів.

**initializeBlade**(): Відповідає за ініціалізацію і налаштування шаблонізатора Blade.

**initializeDatabase**(): Відповідає за ініціалізацію з'єднання з базою даних.

**setupRoutes**(): Визначає маршрути у програмі, зв'язуючи шляхи з обробниками запитів.

Приклад реалізації:

$app = Application::getInstance();

$app->run();

Цей код демонструє як клас Application використовується для запуску програми. Метод getInstance() гарантує, що використовується тільки один інстанс класу, а run() ініціює обробку вхідних запитів.

Програма також підтримує консольні команди для інтеракції з базою даних, наприклад, для показу інформації про конкретну особу за ID. Це дозволяє ефективно інтегрувати функціональність веб-додатку з консольним управлінням, що розширює можливості використання програми.

**Опис класів:**

Клас Person - відповідає за роботу з людьми. Він відповідає за завантаження, збереження, видалення та редагування даних про людину.

Клас Tree - необхідний для побудови фамільного дерева.

Singleton клас Database - відповідає за роботу з базою даних SQLite.

Клас Table - відповідає за роботу з таблицями. Він відповідає за завантаження, збереження, видалення та редагування даних в таблицях.

Методи даних класів забезпечують формування інтерфейсу взаємодії з базою даних та дозволяють реалізувати складні логічні операції з даними, що необхідні для керування інформацією про осіб, управління фамільним деревом, а також ефективне виконання операцій над таблицями бази даних.

РОЗДІЛ 2. Проєктна специфікація

2.1 Структура бази даних

**Таблиця "people"**

Основна таблиця "people" призначена для зберігання інформації про осіб у контексті фамільного дерева. Вона включає наступні поля (рис 2.1):

id: Унікальний ідентифікатор особи в базі даних. Використовується як первинний ключ, що автоматично інкрементується.

first\_name: Ім'я особи.

last\_name: Прізвище особи.

gender: Стать особи. Значення поля обмежене переліком можливих статей.

birth\_date: Дата народження.

death\_date: Дата смерті, яка може бути нульовою, якщо особа ще жива.

father\_id: Ідентифікатор батька в базі даних. Зовнішній ключ, що посилається на id в тій самій таблиці.

mother\_id: Ідентифікатор матері. Аналогічно є зовнішнім ключем.

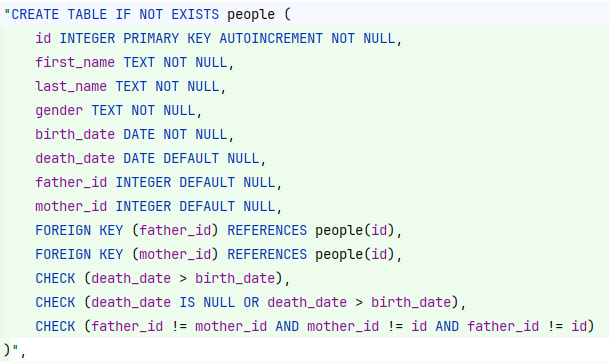


Рис 2.1 - SQL запит на створення таблиці.

**Обмеження таблиці:**

Обмеження на CHECK (death\_date > birth\_date) гарантує, що дата смерті завжди пізніша за дату народження.

CHECK (father\_id != mother\_id AND mother\_id != id AND father\_id != id) запобігає встановленню батька і матері як однієї й тієї ж особи, а також самопосилання.

2.2 Об'єктна структура програми

Програма має кілька ключових класів для взаємодії з базою даних та управління бізнес-логікою:

**Клас Person**

Клас Person відповідає за завантаження, збереження, видалення та редагування даних про осіб. Він дозволяє створювати нові об'єкти осіб, зберігати їх у базі, оновлювати існуючі записи та видаляти їх за необхідності.

**Клас Tree**

Клас Tree використовується для побудови фамільного дерева, забезпечуючи методи для визначення стосунків між особами, таких як батьки, діти, родичі.

**Клас Database**

Singleton клас Database відповідає за управління підключенням до бази даних SQLite. Це гарантує, що всі запити до бази виконуються через одне підключення, зменшуючи навантаження та використання ресурсів.

**Клас Table**

Клас Table спеціалізується на операціях з таблицями, як-от завантаження даних, їх збереження, редагування та видалення. Цей клас може використовуватися для управління будь-якими таблицями в базі даних, забезпечуючи широкий спектр утиліт для роботи з даними.

Ці класи разом формують основу для бізнес-логіки додатку, забезпечуючи всі необхідні інструменти для управління даними та їх представлення у зручному для користувача вигляді.

РОЗДІЛ 3. Інструкція користувача

Цей розділ надає інструкції щодо встановлення та використання веб-додатку "Фамільне дерево", а також описує доступні консольні команди для роботи з програмою.

3.1 Встановлення

**Клонування репозиторію:**

Перш за все, вам потрібно клонувати репозиторій з кодом додатку на ваш комп'ютер. Відкрийте термінал і введіть наступну команду:

git clone https://github.com/Maryana232/family\_tree.git

Ця команда створить копію проєкту у вказаній папці.

**Встановлення залежностей:**

Після клонування репозиторію перейдіть у кореневу директорію проєкту і встановіть необхідні залежності за допомогою Composer:

cd family\_tree

composer install

**Запуск локального сервера:**

Для запуску локального сервера і перегляду веб-додатку в браузері виконайте наступну команду:

php -S localhost:8000 -t public

Відкрийте веб-браузер і перейдіть за адресою http://localhost:8000, щоб перевірити роботу додатку.

3.2 Консольні команди

Додаток "Фамільне дерево" також підтримує ряд консольних команд, які дозволяють керувати даними без використання веб-інтерфейсу. Ось основні команди:

**Вивести дерево для конкретної особи:**

Щоб побачити фамільне дерево для особи з певним ID, використовуйте наступну команду:

php console.php show-ancestors 6

Ця команда покаже всіх предків особи з ID 6.

**Вивести всіх нащадків для конкретної особи:**

Для виведення інформації про всіх нащадків певної особи скористайтесь командою:

php console.php show-descendants 12

Ця команда виведе список всіх нащадків особи з ID 12.

Ці команди є важливими інструментами для швидкого доступу та управління даними про осіб у базі даних вашого фамільного дерева.

3.3. Інтерфейс користувача

Інтерфейс користувача веб-додатку "Фамільне дерево" складається з низки інтуїтивно зрозумілих елементів, які забезпечують зручне управління даними та перегляд інформації. Ось основні елементи інтерфейсу, що використовуються на веб-сторінках:

Форми:

Форма додавання особи: Дозволяє користувачам вводити дані нової особи, такі як ім'я, прізвище, стать, дати народження та смерті. Ця форма також містить поля для вибору батьків з існуючих записів, що дозволяє зв'язувати родинні відносини.

Форма редагування особи: Користувачі можуть оновити існуючі дані про осіб. Форма автоматично заповнюється існуючими даними особи, яку обрано редагувати, що спрощує процес коригування.

Форма видалення особи: Проста форма з можливістю підтвердження перед видаленням даних особи, для уникнення випадкового видалення важливих даних.

Таблиці:

Таблиця перегляду осіб: Відображає всіх осіб, зареєстрованих у базі даних, з основними деталями, такими як ім'я, прізвище, дати народження та смерті. Ця таблиця також має кнопки дій для кожного запису, такі як редагування чи видалення.

Таблиця фамільного дерева: Візуалізує структуру родинних зв'язків осіб. Показує як прямі зв'язки (наприклад, батьки і діти), так і розширені (наприклад, онуки, брати і сестри).

Навігаційна панель:

Верхня навігаційна панель дозволяє користувачам легко переміщатися між різними сторінками додатку, такими як головна сторінка, сторінка додавання особи, перегляд таблиці осіб тощо.

Інтерактивні елементи:

Кнопки: Для виконання різних дій, таких як подання форм, видалення записів, або перехід до інших частин інтерфейсу.

Посилання: Використовуються для навігації або як частина інтерфейсу для швидкого доступу до пов'язаних дій або інформації.

Ці елементи разом створюють користувацький інтерфейс, який є не тільки функціональним, але й приємним для користувача, сприяючи ефективній роботі з даними і легкому доступу до необхідної інформації в контексті фамільного дерева.

ВИСНОВКИ

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

Додаток А