1. Descripció del projecte

Aquest document descriu els requeriments per a una aplicació web desenvolupada en PHP que permet el manteniment de frases celebres. L'aplicació seguirà el patró de disseny MVC (Model-Vista-Controlador) i utilitzarà el patró Singleton per gestionar la configuració de l'aplicació mitjançant un arxiu XML.

L'objectiu principal d'aquesta pràctica és desenvolupar una aplicació web en PHP que permeti gestionar frases celebres, autors i temes mitjançant un sistema estructurat seguint el patró MVC.

L'aplicació ha de:

- Permetre la creació, lectura, actualització i eliminació (CRUD) de frases celebres.
- Gestionar autors i temes associats a cada frase.
- Utilitzar una base de dades MySQL per emmagatzemar la informació.
- Implementar el patró Singleton per carregar la configuració des d'un fitxer XML.

L'accés a aquesta part de l'aplicació serà a partir de la zona privada, havent d'estar identificat mitjançant un usuari i password.

2. Estructura del Projecte

L'aplicació estarà organitzada segons el patró MVC, amb les següents carpetes:

- models/ Contindrà les classes que representen els objectes de negoci (Phrase, Author, Theme).
- controllers/ Gestionarà la lògica d'aplicació i la interacció entre el model i la vista.
- views/ Contindrà els fitxers que generen la interfície d'usuari.
- config/ Contindrà el fitxer de configuració XML.
- core/ Contindrà classes auxiliars, incloent-hi la classe Singleton per gestionar la configuració.
- public/ Contindrà els arxius accessibles des del navegador (CSS, JavaScript, etc.).

3. Models (Objectes de Negoci)

L'accés a la base de dades es realitzarà mitjançant **PDO** per millorar la seguretat i portabilitat.

3.1. Classe Phrase

Representa una frase cèlebre i conté els següents atributs:

- text: Contingut de la frase.
- author: Autor de la frase.
- theme: Rema relacionat.
- created_at: Data de creació.
- updated_at: Data d'última modificació.

3.2. Classe Author

Representa un autor i conté els següents atributs:

- nom: Nom de l'autor.
- descripcio: Descripció de l'autor.
- url: Url de l'autor.

3.3. Classe Theme

Representa un tema associat a una frase i conté els següents atributs:

- nom: Nom del tema.
- descripcio: Descripció del tema.

4. Base de Dades

L'aplicació utilitzarà una base de dades MySQL amb les següents taules:

Hauràs de definir les taules corresponents amb el prefix tbl_
El nom de la base de dades serà frases_Cognom_Nom…. en el meu cas sería frases_Aguilar_Toni.

5. Patrons de Disseny Implementats

5.1. Patró MVC

L'aplicació seguirà el patró Model-Vista-Controlador per garantir una separació clara de responsabilitats:

- Model: Gestiona l'accés a la base de dades i representa les dades.
- Vista: Presenta la informació a l'usuari.
- Controlador: Gestiona la lògica de negoci i les interaccions amb l'usuari.

5.2. Patró Singleton per a la Configuració

La configuració de l'aplicació es gestionarà mitjançant un fitxer XML.
La classe Config implementarà el patró Singleton per assegurar que només es carrega una única instància de la configuració.

Exemple de fitxer config.xml:

```
<config>
        <database>
        <host>localhost</host>
        <user>root</user>
        <password>password</password>
        <dbname>phrases_db</dbname>
        </database>
</config>
```

5.3 Ús del patró DataMapper

En aquesta segona part aplicarem el patró de disseny DataMapper per realitzar l'edició de la taula Autor. Primer de tot, farem que la clau primària sigui autonumerica.

DataMapper, és una Capa d'accés a Dades que realitza una transferencia bidireccional de dades entre un magatzem de dades persistent (normalment una base de dades relacional) i una representació de dades en memoria (la capa de domini). L'objectiu del patró és mantenir la representació en memòria i el magatzem de dades persistent separades entre si i del propi data mapper. La capa està formada per un o més mappers (o Capa d'Accés a Dades),realitzant la transferència de dades. La implementació de mappers varien en abast. Els mapejadors genèrics manejaran molts tipus d'entitats de domini diferents, els mapejadors dedicats manejaran un o pocs.

Aquest patró s'utilitza:

- Classes i objectes de negoci. Els objectes de negoci no tenen comportament, simplement alberguen dades. Generalment, la definició de la classe de negoci es correspon amb la definició d'una taula de la base de dades i un objecte de negoci emmagatzema les dades d'una fila d'aquesta taula. La classe bàsicament disposa dels getters i setters necessaris i opcionalment d'un constructor.

- Classes i objectes DAO. Les classes DAO contenen mètodes per accedir a l'origen de dades i reben els paràmetres i retornen els resultats en forma d'objectes de negoci. Generalment realitzen funcions CRUD (create, read, update, delete). Segons la implementació pot haver-hi una sola classe DAO o una per cada taula o entitat.

- Interfícies. Les interfícies s'utilitzen per fer possible diferents implementacions de les classes DAO. En aquesta pràctica no les utilitzarem.

L'existència de relacions entre les entitats o taules complica l'escriptura de classes DAO. Per això ens limitarem a realitzar el manteniment de la taula autors. Quan hi relacions es recomana l'ús d'una eina ORM. Hi ha moltes disponibles: Doctrine, PHPDataMapper, Propel, etc ...

6. Funcionalitats Principals

6.1. CRUD de Frases Célebres

- Afegir una nova frase.
- Editar una frase existent.
- Esborrar una frase.
- Llistar totes les frases.

6.2. Gestió d'Autors

- Afegir, editar i eliminar autors.
- Llistar autors disponibles.

6.3. Gestió de Temes

- Afegir, editar i eliminar temes.
- Llistar temes disponibles.