Documentation pour un Système ORM en PHP

I. Overview

Ce projet est un système ORM simple développé en PHP 8 et MySQL. Il fournit une manière simplifiée d'interagir avec une base de données MySQL en utilisant des objets PHP, facilitant la création, la lecture, la mise à jour et la suppression des enregistrements de la base de données.

Le système inclut des fonctionnalités pour la création dynamique de tables et l'application de contraintes uniques sur les colonnes de la base de données.

II. Fonctionnalité

Le système ORM permet aux développeurs de :

- Interagir avec la base de données via des objets PHP.
- Automatiser la création et la mise à jour des tables de la base de données basées sur des classes PHP.
- Gérer dynamiquement les contraintes uniques comme les clés primaires et les clés uniques.
- Effectuer des opérations CRUD (Create, Read, Update, Delete) en utilisant des methods cohérentes et facile à utiliser.

III.Composants du Code

1) config.php et Database.php

```php
<?php
// config.php
return [
    'host' => 'localhost',
    'dbname' => 'orm',
    'username' => 'root',
    'password' => '',
    'options' => [
        PDO::ATTR_ERRMODE => PDO::ERRMODE_EXCEPTION,
        PDO::ATTR_DEFAULT_FETCH_MODE => PDO::FETCH_ASSOC,
    ],
];
?>
```

```php
<?php
// Database.php
class Database {
    private $pdo;

    public function __construct() {
        $config = require 'config.php';
        if (!isset($this->pdo)) {
            try {
                $this->pdo = new PDO('mysql:host=' . $config['host'] . ';dbname=' . $config['dbname'], $config['username'], $config['password'], $config['options']);
            } catch (PDOException $e) {
                throw new PDOException($e->getMessage());
            }
        }

        return $this->pdo;
    }

    public function getConnection() {
        return $this->pdo;
    }
}
?>
```

2.1) Objectif:

Le fichier ‘config.php ’contient les paramètres de configuration de la base de données. Il retourne un tableau avec les détails nécessaires pour établir une connexion PDO.

La fichier ‘Database.php’ contient une class qui gère la connexion à la base de données en utilisant PDO. Elle lit la configuration depuis config.php et établit une connexion à la base de données MySQL.

3) ORMInterface.php et ORM.php

```php
<?php
// ORMInterface.php

interface ORMInterface {
    public function find($id);
    public function all();
    public function create($attributes);
    public function update($id, $attributes);
    public function save($attributes);
    public function delete($id);
    public function createTable($columns);
    public function alterTable($choice, $columns = []);
}
?>

```

3.1) Objectif:

Le fichier ‘ORMInterface.php ’ définit l'interface pour la classe ORM, spécifiant les méthodes qui doivent être implémentées pour gérer les interactions avec la base de données

La fichier ‘ORM.php’ Implémente ORMInterface pour fournir les fonctionnalités réelles d'interaction avec la base de données. Elle inclut des méthodes pour créer, lire, mettre à jour et supprimer des enregistrements, ainsi que pour créer des tables dynamiquement.

4) User.php et Product.php

```php
<?php
// User.php

class User {
    protected $table = 'users';
    protected $attributes = [
        'id' => 'id',
        'user_name' => 'string',
        'first_name' => 'string',
        'last_name' => 'string',
        'email' => 'string',
        'password' => 'string'
    ];

    public function getTable() {
        return $this->table;
    }

    public function getAttributes() {
        return $this->attributes;
    }
}
```

Le fichier ‘User.php’ définit le modèle User avec des attributs spécifiques pour une entité utilisateur, ainsi qu’une method pour la récupération de ses attributs.

La méme logic s’applique au Product.php

5) Les scripts de test (testCreate, testRead, testUpdate testDelete etc...)

```php
<?php
// testCreate.php

require 'ORM.php';

try {
    $orm = new ORM('users');
    $success = $orm->create([
        'user_name' => 'sife69',
        'first_name' => 'Sife',
        'last_name' => 'Yassine',
        'email' => 'sife@example.com',
        'password' => password_hash('qwert123', PASSWORD_BCRYPT)
    ]);

    if ($success) {
        echo "User created successfully.";
    } else {
        echo "Failed to create user.";
    }
} catch (Exception $e) {
    echo "Error creating user: " . $e->getMessage();
}
```

```php
<?php
// testRead.php

require 'ORM.php';

try {
    $orm = new ORM('users');
    $user = $orm->find(1);

    if ($user) {
        echo "User found: " . $user['id'] . " | " . $user['user_name'] . " | " . $user['first_name'] . " | "
    } else {
        echo "User not found.";
    }
} catch (Exception $e) {
    echo "Error reading user: " . $e->getMessage();
}
?>

```

5.1) Objectif:

Le fichier ‘testCreate.php’ est un script pour tester la création des enregistrements User et Product dans la base de données en utilisant le système ORM.

Le fichier ‘testRead.php’ est un script pour tester la lecture des enregistrements User et Product depuis la base de données en utilisant le système ORM.

Les autres scripts (testUpdate.php, testDelete.php & testReadAll.php) ont la méme logic

6) createTable.php

Ce script PHP est responsable de la création des tables d'une facon dynamique dans la base de données en utilisant le système ORM

```php
<?php
// createTable.php

require 'ORM.php';
require 'User.php';

try {
    $user = new User();
    $orm = new ORM($user->getTable());
    $orm->createTable($user->getAttributes());

    echo "Table '{$user->getTable()}' created successfully.";
} catch (Exception $e) {
    echo "Error creating table '{$user->getTable()}': " . $e->getMessage();
}
?>
```