Actividad de desarrollo en Servidor.

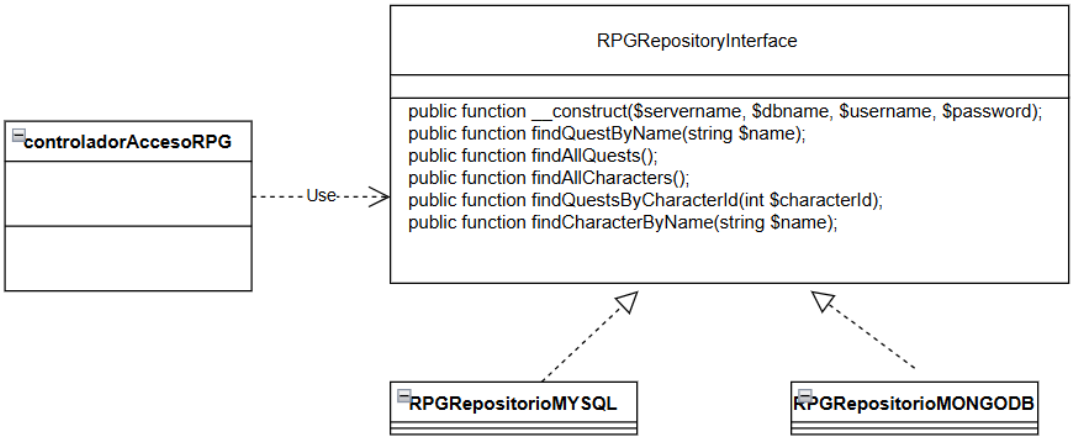
Acceso a datos desde PHP a bases de datos MYSQL y MONGODB(II)

Índice de apartados

Descripción

Vamos a modificar el laboratorio práctico anterior(parte i) para organizar mejor el acceso a los repositorio de otra forma más conveniente.

El diagrama de clases del nuevo diseño se muestra a continuación:

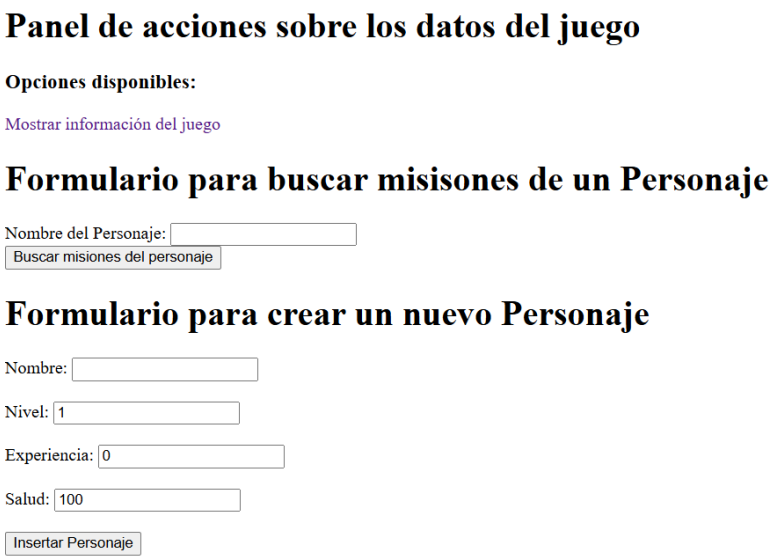


Componente del diagrama:

- **controladorAccesoRPG:** Organiza el acceso al repositorio en función de las ordenes de la interfaz gráfica
- **RPGRepositoryInterface:** Abstraer el comportamiento de cualquier repositorio. **Es una interfaz sbtracta**
- **RPGRepositorioMYSQL:** Implementa la interfaz RPGRepositoryInterface para acceder a MYSQL
- **RPGRepositorioMONGODB** Implementa la interfaz RPGRepositoryInterface para acceder a MONGODB

Componentes de la actividad práctica

Index.html



Implementación en HTML

```
<!DOCTYPE html>

<html lang="es">


<head>

    <meta charset="UTF-8">

    <meta name="viewport" content="width=device-width, initial-
scale=1.0">

    <title>Panel de acciones sobre los datos del juego</title>

</head>


<body>

    <h1>Panel de acciones sobre los datos del juego</h1>

    <h3>Opciones disponibles:</h3>

    <ul>

        <li><a href="controladorAccesoRPG.php">Mostrar información del
juego</a>

        <li><h1>Formulario para buscar misisones de un
Personaje</h1></li>

        <li>
```

```
            <form action=".." method="get">
                <label>Nombre del Personaje:</label>
                <input type="text" name="characterName"><br>

                <input type="submit" value="Buscar misiones del
personaje">
            </form>
        </li>
        <li>

            <h1>Formulario para crear un nuevo Personaje</h1>
            <form action="...." method="post">
                <label for="name">Nombre:</label>
                <input type="text" id="name" name="name"
required><br><br>

                <label for="level">Nivel:</label>
                <input type="number" id="level" name="level" value="1"
min="1"><br><br>

                <label for="experience">Experiencia:</label>
                <input type="number" id="experience" name="experience"
value="0" min="0"><br><br>

                <label for="health">Salud:</label>
                <input type="number" id="health" name="health"
value="100" min="0"><br><br>

                <input type="submit" value="Insertar Personaje">
            </form>

        </li>

    </ul>
```

```
</body>


</html>
```

controladorAccesoDB.php

Módulo php encargado de procesar la solicitudes desde la interface de usuario (html)y trasladarlas hasta la implementación del repositorio.

NOTA: El código está incompleto debes completarlo

```
<?php


require_once ‘???????????’;

require_once '??????????';


try {


    // Crear una instancia del repositorio

    $repository = new ¿???


    echo " <h1>Información del juego : LISTA DE PERSONAJES Y LISTA DE
DESAFIOS:</h1>";


    // Mostrar  todos los personajes

    $characters = $repository->????

    echo " <h3>Lista de personajes del juego: </h3>";

    foreach ($characters as $character) {

        echo ¿???

        echo "------\n";  // Separador entre personajes

    }



    //Mostrar todas las quests/desafios

    ¿?????
```

```
} catch (PDOException $e) {
    echo 'Error de conexión: ' . $e->getMessage();
}
```

RPGRepositoryInterface

Esta es la **interfaz que representa el comportamiento común** que se espera de todos los repositorios que se usen (en nuestro caso MYSQL y MONGODB).

NOTA :Fíjate bien en todos los métodos que define y sus parámetros

```
<?php


interface RPGRepositoryInterface
{
    public function __construct($servername, $dbname, $username, $password);


    // Método para encontrar una quest por su nombre
    public function findQuestByName(string $name);


    // Método para obtener todas las quests
    public function findAllQuests();


    // Método para obtener todos los personajes
    public function findAllCharacters();


    // Método para obtener las quests asociadas a un personaje por su ID
    public function findQuestsByCharacterId(int $characterId);


    // Método para encontrar un personaje por su nombre
    public function findCharacterByName(string $name);
}
```

RPGRepositoryMONGODB

Esta es la implementación de la interfaz abstracta para la base de datos de Mongo DB.

```
<?php
require 'vendor/autoload.php';
require_once 'RPGRepositoryInterface.php';
class RPGRepositoryMONGODB implements RPGRepositoryInterface {
    private $client;
    private $db;

    public function __construct($servername, $dbname, $username, $password) {
        // Conectar a la base de datos MongoDB

        $this->client = new MongoDB\Client("mongodb://$servername:27017");

        //$this->client = new MongoDB\Client("mongodb://$username:$password@$servername/$dbname");
        $this->db = $this->client->selectDatabase($dbname);
    }

    // Método para encontrar una quest por su nombre
    public function findQuestByName(string $name) {
        $quest = $this->db->quests->findOne(['title' => $name]);
        return $quest ? $quest : null;
    }

    // Método para obtener todas las quests
    public function findAllQuests() {
        $quests = $this->db->quests->find();
        return iterator_to_array($quests);
    }
```

```
    // Método para obtener todos los personajes
    public function findAllCharacters() {
        $characters = $this->db->characters->find();
        return iterator_to_array($characters);
    }

    // Método para obtener las quests asociadas a un personaje por su ID
    public function findQuestsByCharacterId(int $characterId) {
        $character = $this->db->characters->findOne(['_id' => new MongoDB\BSON\ObjectId((string)$characterId)]);
        return $character ? $character['quests'] : null;
    }

    // Método para encontrar un personaje por su nombre
    public function findCharacterByName(string $name) {
        $character = $this->db->characters->findOne(['name' => $name]);
        return $character ? $character : null;
    }
}
```

RPGRepositoryMYSQL

Esta es la implementación de la interfaz abstracta para la base de datos de Mongo DB.

NOTA : Se han omitido partes de la implementación

```
<?php
require 'vendor/autoload.php';
require_once 'RPGRepositoryInterface.php';

class RPGRepositoryMySQL implements RPGRepositoryInterface
{
```

```
    private $conn; // Conexión a MYSQL


    // Constructor con los parámetros de conexión
    public function __construct($servername, $dbname, $username, $password)
    {
        // Crear conexión


        $this->conn = new PDO("mysql:host=$servername;dbname=$dbname", $username, $password);
        // Establecer el modo de error de PDO a excepción
        $this->conn->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);
    }


    // Implementación para encontrar una quest por su nombre
    public function findQuestByName(string $name): mixed
    {
        ??????????
    }


    // Implementación para obtener todas las quests
    public function findAllQuests(): array
    {
        $sql = "SELECT * FROM quests";
        $stmt = $this->conn->query($sql);
        return $stmt->fetchAll(PDO::FETCH_ASSOC);
    }


    // Implementación para obtener todos los personajes
    public function findAllCharacters(): array
    {
        $sql = "SELECT * FROM characters";
```

```
        $stmt = $this->conn->query($sql);
        return $stmt->fetchAll(PDO::FETCH_ASSOC);
    }

    // Implementación para obtener las quests asociadas a un personaje por su ID
    public function findQuestsByCharacterId(int $characterId): array
    {
        $sql = "
            SELECT q.*
            FROM quests q
            JOIN character_quests cq ON q.id = cq.quest_id
            WHERE cq.character_id = :characterId
        ";
        $stmt = $this->conn->prepare($sql);
        $stmt->execute(['characterId' => $characterId]);
        return $stmt->fetchAll(PDO::FETCH_ASSOC);
    }

    // Implementación para encontrar un personaje por su nombre
    public function findCharacterByName(string $name): mixed
    {

        ????????????????
    }
}
```

Desarrollo de la actividad

Cuestiones

1. Implementa la aplicación y pruébala. Debes completar las partes de cada módulo (se han indicado con la secuencia ??????)

2. **La interfaz no define los tipos devueltos en cada uno de sus métodos**. Actualiza la definición de la interfaz para que comprenda también los tipos devueltos por cada función **`RPGRepositoryInterface`**

```
    public function findAllCharacters();
```

3. Como ves la interfaz permite hacer tres acciones básicamente. Cambia el controladorAccesoPHP para que use **un parámetro denominado action (se pasará por GET). A partir de este parámetro**
   Si es 1 se hará la primera acción, si es 2 la segunda, etc

4. Si te fijas en el método **`findQuestByCharactersID(int $characterID) de la clase RPGRepositoryMYSQL`**

El método busca un personaje y sus misiones pero a partir del ID. Sin embargo en la interfaz (index.html) se buscará por un nombre de personaje no por el id

Fijate por que en el siguiente fragmento se han indicado las partes donde se muestra esta circunstancia

```
1. public function findQuestsByCharacterId(int $characterId): array
2.     {
3.         $sql = "
4.             SELECT q.*
5.             FROM quests q
6.             JOIN character_quests cq ON q.id = cq.quest_id
7.             WHERE cq.character_id = :characterId
8.         ";
9.         $stmt = $this->conn->prepare($sql);
10.              $stmt->execute(['characterId' => $characterId]);
11.              return $stmt->fetchAll(PDO::FETCH_ASSOC);
12.          }
```

Cambia la interfaz e implementación del método para que el método sea

findQuestByCharactersID(string $characteraname) y la implementación busque el personaje por el nombre no por el id

¿ Qué cambios tendrías que hacer y en qué módulos de la aplicación

5. La interfaz tiene una opción "Insertar un nuevo personaje". Sin embargo esta opción no está contemplada en el código.
   ¿ Que partes de la aplicación tienes que añadir para que está nueva funcionalidad se implemente en cualquier repositorio que se utilice? Implementa la solución
6. ¿Cómo se podría implementar que sólo se pudiese insertar elementos un usuario que tuviera el rol de admin? ¿ Qué cosas nuevas necesitas?