Actividad de desarrollo en Servidor.

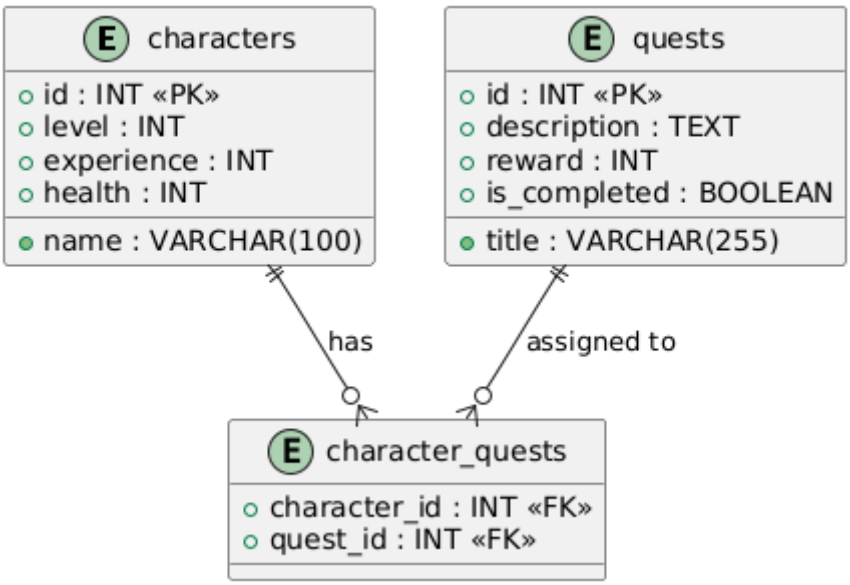
Acceso a datos desde PHP a bases de datos MYSQL y MONGODB(I)

Índice de apartados

Descripción

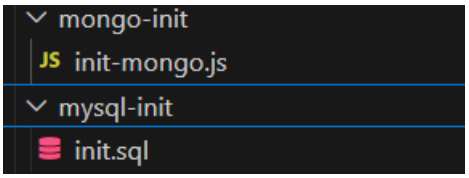
En esta actividad vamos a desarrollar varias cuestiones importantes acerca del modelado y acceso a datos desde el servidor en PHP. Esta actividad practica es la primera de una serie.

El siguiente diagrama E/R muestra un diseño de base de datos sobre un juego de rol (centrado en la mitología griega). Este modelo representa personajes y desafíos relacionados con la Odisea.



NOTA: Recuerda que este diagrama modela datos y sus interrelaciones. **Los scripts de creación de este modelo de datos están en la carpeta dentro del proyecto** (ver más adelante).Lo importante es que **estos scripts se lanzarán sobre dos sistemas gestores de datos** :MYSQL y MONGODB **de forma automatizada** creando la base de datos en ambos sistemas. Es decir, vamos a crear el mismo modelo de datos en dos sistemas de gestión de datos diferentes.

En tú proyectos debes ver dos carpetas relacionadas con los scripts de creación de las bases de datos.



Requisitos previos para la realización de la práctica

Antes de seguir vamos a instalar una seria de herramientas que nos ayudarán a poner todo en funcionamiento

Realiza o comprueba lo siguiente:

```
sudo apt install php-cli php-mbstring unzip curl -y
```

Descarga el instalador de Composer usando curl:

NOTA: cURL (Client URL) es una herramienta de línea de comandos y una biblioteca para transferir datos con URLs.

```
curl -sS https://getcomposer.org/installer -o composer-setup.php
```

ahora instala composer

```
sudo php composer-setup.php --install-dir=/usr/local/bin --
filename=composer
```

verifica la versión de la instalación

```
composer –version
```

Otra forma más sencilla de instalar composer

```
curl -sS https://getcomposer.org/installer | php -- --install-
dir=/usr/local/bin --filename=composer
```

Ahora vamos con otra herramienta muy importante (SEGURAMENTE LA TENGAS YA INSTALADA)

docker-compose

```
sudo apt update

sudo apt install docker.io

sudo systemctl start docker

sudo systemctl enable docker
```

descarga docker-compose

```
sudo curl -L
"https://github.com/docker/compose/releases/latest/download/dock
er-compose-$(uname -s)-$(uname -m)" -o /usr/local/bin/docker-
compose
```

Después de descargar Docker Compose, debes aplicar permisos ejecutables al binario:

```
sudo chmod +x /usr/local/bin/docker-compose
```

Descripción de la infraestructura

Tú proyecto necesita una infraestructura compuesta de tres servidores:

- Apache -php8
- MYSQL
- MongoDB

Para definir y levantar toda esta infraestructura vamos a usar contenedores Docker

IMPORTANTE :Algo muy importante es que necesitamos que los contenedores se comuniquen entre si. Desde el servidor de Apache el código php debe hacer llamadas a los servidores de bases de datos. Para comunicar los servidores Docker **debemos crear un red docker, si no, no podrán hablar entre si.**

Podemos definir todos los tipos de contenedores, sus dependencias y la red que los une en un archivo especial denominado **docker-compose.yml**

Archivo docker-compose.yml

Este archivo es clave, ya que define toda la infraestructura y sus dependencias

```
version: '3.8'
services:
  php-apache:
    build:
      context: .
      dockerfile: Dockerfile # Usar el Dockerfile personalizado
    container_name: php-apache8
    volumes:
      - C:\NuevosProyectosPHP:/var/www/html
    ports:
      - "8081:80"
    depends_on:
      - mongo
      - mysql
    networks:
```

```
      - app-network

  mongo:
    image: mongo:latest
    container_name: mongo
    volumes:
      - ./dataMongoDB:/data/db
      - ./mongo-init:/docker-entrypoint-initdb.d
    ports:
      - "27017:27017"
    networks:
      - app-network

  mysql:
    image: mysql:latest
    container_name: mysql
    environment:
      MYSQL_ROOT_PASSWORD: 1234
    volumes:
      - ./dataMYSQL:/var/lib/mysql
      - ./mysql-init:/docker-entrypoint-initdb.d
    ports:
      - "3306:3306"
    networks:
      - app-network

networks:
  app-network:
    driver: bridge
```

Archivo Dockerfile

A veces necesitamos que cuando se crea un contendor se instalen en él determinadas librerías . Por ejemplo, en el contendor de Apache necesitamos que se instalen todas las librerías de MYSQL y Mongo DB.

Además, necesitamos definir una archivo php.ini donde activaremos las extensiones de mongo y mysql como módulos.

```
# Utiliza la imagen oficial de PHP con Apache como base
FROM php:8.2-apache

# Instalar las extensiones de MongoDB y MySQL
RUN apt-get update && apt-get install -y \
    libpng-dev \
    libjpeg-dev \
    libfreetype6-dev \
    libonig-dev \
    libxml2-dev \
    && docker-php-ext-install mysqli pdo_mysql \
    && pecl install mongodb \
    && docker-php-ext-enable mongodb

# Habilitar mod_rewrite en Apache
RUN a2enmod rewrite

# Copia los archivos de configuración (opcional si tienes alguno)
 COPY ./php.ini /usr/local/etc/php/

# Exponer el puerto por defecto de Apache
EXPOSE 8081
```

Levantar la infraestructura del laboratorio

Debes situarte ahora en la carpeta del proyecto `ProyectosPHP/p_accesoBBDD` que te has descargado del AulaVirtual

Ejecuta

```
docker-compose up -d
```

```
✔Network p_accesobbdd_app-network  Created
✔Container mysql                   Started
✔Container mongo                   Started
✔Container php-apache8             Started
```

Y ejecuta ahora:

```
docker ps
```

```
CONTAINER ID   IMAGE                     COMMAND                  CREATED         STATUS         PORTS
                NAMES
c2d5ee5bb051   p_accesobbdd-php-apache   "docker-php-entrypoi…"   3 minutes ago   Up 2 minutes   8081/tcp, 0.0.0.0:8081->80/tcp      php-apache8
1de4071ce425   mysql:latest              "docker-entrypoint.s…"   3 minutes ago   Up 3 minutes   0.0.0.0:3306->3306/tcp, 33060/tcp   mysql
bddf0da40301   mongo:latest              "docker-entrypoint.s…"   3 minutes ago   Up 3 minutes   0.0.0.0:27017->27017/tcp            mongo
```

Si has obtenido este resultado(ver arriba) es que la infraestructura que necesitamos se ha desplegado correctamente.

Vamos a ver detalles de los script de creación de datos

Detalle del archivo de creación del modelo de datos mysql-init/init.sql

```
CREATE DATABASE IF NOT EXISTS rpg_game;


USE rpg_game;


-- Crear tabla characters
CREATE TABLE characters (
    id INT AUTO_INCREMENT PRIMARY KEY,
    name VARCHAR(100) NOT NULL,
    level INT DEFAULT 1,
    experience INT DEFAULT 0,
    health INT DEFAULT 100
);


-- Insertar datos de ejemplo en la tabla characters (mitología griega)
INSERT INTO characters (name, level, experience, health) VALUES
```

```
('Ulises', 5, 500, 100),
('Penélope', 4, 300, 80),
('Telemaco', 3, 250, 70),
('Circe', 6, 400, 90),
('Calipso', 5, 350, 85),
('Atenea', 7, 600, 95);
```

Detalle del archivo de creación del modelo de datos mongo-init/init.sql

```
b = db.getSiblingDB('rpg_game'); // Cambia a la base de datos rpg_game


// Crear la colección characters y agregar un documento
db.characters.insertMany([
    {
        name: "Ulises",
        level: 5,
        experience: 1200,
        health: 100,
        quests: [
            {
                quest_id: ObjectId(), // Puedes llenar más tarde con el id real
                title: "Regresar a Ítaca",
                is_completed: false
            },
            {
                quest_id: ObjectId(), // Puedes llenar más tarde con el id real
                title: "Derrotar a los Ciclopios",
                is_completed: true
            }
```

```
        ]
    },
    {
        name: "Telémaco",
        level: 3,
        experience: 500,
        health: 90,
        quests: []
    }
]);
```

Fíjate en quest_id: ObjectId(), pues la manera que tiene una base de datos orientada a documentos de guardar la referencia al desafío quest con el que está relacionado el personaje.

IMPORTE: Mongodb no tiene medidas de comprobar la integridad referencial

Dependencias del proyecto (relación con composer)

Tú proyecto necesita dos librerías importantes en el cliente PHP para conectarse a los SGBD .En concreto, aquellas relacionadas con la conexión a MYSQL y las relacionadas con MongoDB. **Estas dependencias se indican en el archivo composer.json** y serán resueltas cuando invoquemos a composer desde el directorio del proyecto.

Detalle del archivo composer.json

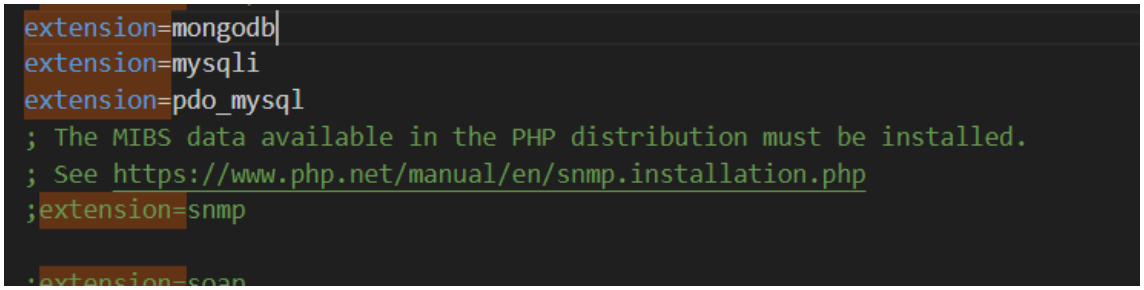
```
{
    "require": {
        "ext-pdo_mysql": "*",
        "mongodb/mongodb": "^1.9"
    },
    "config": {
        "platform": {
            "php": "8.2"
```

```
        }
    }
}
```

NOTA: Lo que descargue composer se ubicará en la carpeta vendor de tú proyecto

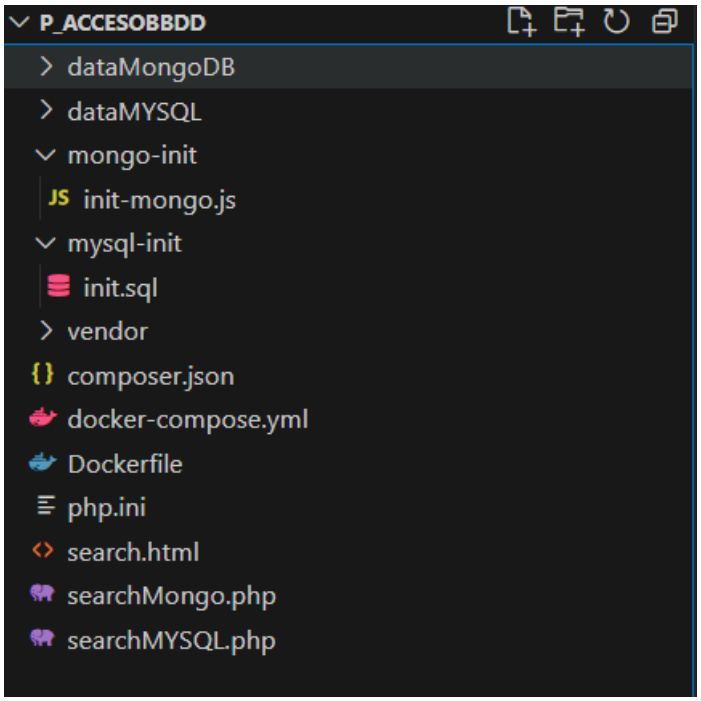
Detalle del archivo php.ini

En tú carpeta de proyecto tienes un archivo **php.ini** que controla aspectos de inicio de php, así como la activación de determinados módulos. Hemos configurado un archivo php.ini en tú proyecto para que puedas cambiarlo sin tener que reconstruir el contendor de Docker cada vez que cambies algo.



Estructura de carpetas de proyecto

La estructura del proyecto **tiene las siguientes partes importantes:**


P_ACCESOBBDD
dataMongoDB
dataMYSQL
mongo-init
init-mongo.js
mysql-init
init.sql
vendor
composer.json
docker-compose.yml
Dockerfile
php.ini
search.html
searchMongo.php
searchMYSQL.php

Acceso a datos desde PHP

Nuestro laboratorio requiere que accedamos a los datos guardados en MYSQL y MongoDB. A continuación, vamos a ver **la anatomía que tiene el código de conexión de acceso para ambos sistemas gestores de datos desde PHP.**

Acceso a MongoDB parámetros	Acceso a MYSQL parámetros

`$servername = "mongodb://mongo:27017";`
`$dbname = "rpg_game";`
	`$servername = "mysql";`
	`$username = "root";`
	`$password = "1234";`
	`$dbname = "rpg_game";`

ACCESO A LA BASE DE DATOS

MYSQL

```
$conn = new PDO("mysql:host=$servername;dbname=$dbname", $username, $password);
```

MONGODB

```
$client = new MongoDB\Client($servername);

$db = $client->$dbname;
```

ACCESO A LOS DATOS (CONSULTA)

Obtener todas la misiones de un personaje

ACCESO MONGODB

```
// Buscar el personaje por nombre

$charactersCollection = $db->characters;

$character = $charactersCollection->findOne(['name' => $characterName]);

    if ($character) {
        echo "Misiones de " . $characterName . ":\n";
        foreach ($character['quests'] as $quest) {
```

```
            echo "- " . $quest['title'] . (isset($quest['is_completed']) &&
$quest['is_completed'] ? " (Completada)" : " (No completada)") . "\n";
        }
    }
```

ACCESO MYSQL

```
$stmt = $conn->prepare("
        SELECT q.title, q.description, q.reward, q.is_completed
        FROM quests q
        JOIN character_quests cq ON q.id = cq.quest_id
        JOIN characters c ON cq.character_id = c.id
        WHERE c.name = :characterName
    ");
    $stmt->bindParam(':characterName', $characterName);
    $stmt->execute();
    // Obtener los resultados
    $missions = $stmt->fetchAll(PDO::FETCH_ASSOC);
```

RECORRIDO POR EL CONJUNTO DE RESULTADOS

RECORRIDO MONGODB

```
foreach ($character['quests'] as $quest) {
  echo "- " . $quest['title'] . (isset($quest['is_completed']) &&
$quest['is_completed'] ? " (Completada)" : " (No completada)") . "\n";
        }
```

RECORRIDO MYSQL

```
foreach ($missions as $mission) {
            echo "<li>";
            echo "<strong>Título:</strong> " . htmlspecialchars($mission['title'])
. "<br>";
            echo "<strong>Descripción:</strong> " .
htmlspecialchars($mission['description']) . "<br>";
            echo "<strong>Recompensa:</strong> " .
htmlspecialchars($mission['reward']) . "<br>";
            echo "<strong>Completada:</strong> " . ($mission['is_completed'] ? 'Sí'
: 'No') . "<br>";
            echo "</li><br>";
        }
```

Interfaz HTML de búsqueda y de acceso a datos

Vamos en este punto a desarrollar la interfaz HTML de acceso a datos.

```
<!DOCTYPE html>
<html lang="es">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Búsqueda de Personajes</title>
</head>
<body>
    <form action="searchMYSQL.php" method="get">
        <label>Nombre personaje Odisea:</label>
        <input type="text" name="character"><br>

        <input type="submit" value="Buscar misiones y desafios BBDD MYSQL">
    </form>
 </body>
</html>
```

Módulo php de acceso y búsqueda de datos :searchMYSQL.php

Este script básicamente se conecta a la base de datos de MYSQL y lanza una consulta buscando las misiones de Ulises.

```
<?php
$servername = "mysql";  // Nombre del servicio definido en docker-compose.yml
$username = "root";      // Nombre de usuario
$password = "1234";      // Contraseña
$dbname = "rpg_game";    // Nombre de la base de datos
```

```
try {
    // Crear conexión
    $conn = new PDO("mysql:host=$servername;dbname=$dbname", $username, $password);

    // Establecer el modo de error de PDO a excepción
    $conn->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);

    // Nombre del personaje a buscar
    $characterName = 'Ulises';

    // Preparar y ejecutar la consulta
    $stmt = $conn->prepare("
        SELECT q.title, q.description, q.reward, q.is_completed
        FROM quests q
        JOIN character_quests cq ON q.id = cq.quest_id
        JOIN characters c ON cq.character_id = c.id
        WHERE c.name = :characterName
    ");
    $stmt->bindParam(':characterName', $characterName);
    $stmt->execute();

    // Obtener los resultados
    $missions = $stmt->fetchAll(PDO::FETCH_ASSOC);

    // Mostrar el nombre del personaje
    echo "<h1>Misiones de $characterName</h1>";

    // Comprobar si hay misiones
```

```
    if (count($missions) > 0) {
        echo "<ul>";
        foreach ($missions as $mission) {
            echo "<li>";
            echo "<strong>Título:</strong> " .
htmlspecialchars($mission['title']) . "<br>";
            echo "<strong>Descripción:</strong> " .
htmlspecialchars($mission['description']) . "<br>";
            echo "<strong>Recompensa:</strong> " .
htmlspecialchars($mission['reward']) . "<br>";
            echo "<strong>Completada:</strong> " .
($mission['is_completed'] ? 'Sí' : 'No') . "<br>";
            echo "</li><br>";
        }
        echo "</ul>";
    } else {
        echo "No hay misiones asignadas a este personaje.";
    }

    // Cerrar la conexión
    $conn = null;
} catch (PDOException $e) {
    echo "Error de conexión: " . $e->getMessage();
}
```

Módulo php de acceso y búsqueda de datos searchMONGODB.php

Este script básicamente se conecta a la base de datos de MONGODB y lanza una consulta buscando las misiones de Ulises.

```
<?php
require 'vendor/autoload.php'; // Asegúrate de tener el autoload de Composer
```

```php
$servername = "mongodb://mongo:27017"; // Nombre del servicio MongoDB en docker-compose
$dbname = "rpg_game"; // Nombre de la base de datos
$characterName = "Ulises"; // Nombre del personaje a buscar

try {
    // Crear conexión a MongoDB
    $client = new MongoDB\Client($servername);

    // Seleccionar la base de datos y la colección
    $db = $client->$dbname;
    $charactersCollection = $db->characters;

    // Buscar el personaje por nombre
    $character = $charactersCollection->findOne(['name' => $characterName]);

    if ($character) {
        echo "Misiones de " . $characterName . ":\n";
        foreach ($character['quests'] as $quest) {
            echo "- " . $quest['title'] . (isset($quest['is_completed']) && $quest['is_completed'] ? " (Completada)" : " (No completada)") . "\n";
        }
    } else {
        echo "Personaje no encontrado.";
    }
} catch (MongoDB\Driver\Exception\Exception $e) {
    echo "Error de conexión: " . $e->getMessage();
}
```

Cuestiones

1. Desarrolla la actividad y prueba a acceder mediante la interfaz de acceso HTML a las bases de datos
2. El código de acceso tiene “hardcoded” el nombre de Ulises. Cambia los módulos de acceso para que busquen la información de misiones de cualquier personaje de la Odisea
3. La interfaz accede a la base de datos MYSQL, ¿qué tendrías que para desde la interfaz se accediera a la base de datos mongodb?
4. Estudia el código, la estructura de los módulos y los contenedores docker .Podrías dibujar un esquema de como se comunican los módulos entre si.