GUIÓN DE LA ACTIVIDAD DE REPASO

Título actividad

AR-Modelado con Doctrine

Objetivos

- Repasar y asimilar los conceptos de modelado de clases y tablas en Doctrine

Recursos generales

Presentaciones y videos del Tema 4.

Ejercicio 1. Sistema de Biblioteca

Crea un modelo para un sistema de biblioteca con las siguientes entidades y relaciones:

1. **Entidad Libro:**
    - Campos:
        - id (entero, clave primaria, auto-incremental)
        - titulo (cadena de texto)
        - isbn (cadena de texto única)
        - publicado (fecha)
2. **Entidad Autor:**
    - Campos:
        - id (entero)
        - nombre (cadena de texto)
        - nacionalidad (cadena de texto)
3. **Relaciones: (bidireccional)**
    - Un **Autor puede escribir varios Libros**.
    - Cada **Libro tiene exactamente un Autor.**

Tareas:

- Modelar las entidades en Doctrine.
- Crear las relaciones entre las tablas, asegurando que incluyan claves foráneas.

Ejercicio 1. Sistema de Biblioteca - SOLUCIÓN

Entidad Book.php

```
namespace App\Entity;

use …

#[Entity(repositoryClass: BookRepository::class)]
#[Table(name: 'libros')]
class Book
{
    #[Id]
    #[GeneratedValue]
    #[Column(name: 'id', type: 'integer')]
    private int $id;

    #[Column(name: 'titulo', type: 'string', length: 255)]
    private string $title;

    #[Column(name: 'isbn', type: 'string', length: 13, unique: true)]
    private string $isbn;

    #[Column(name: 'publicado', type: 'date')]
    private \DateTime $publishedAt;

    #[ManyToOne(targetEntity: Author::class, inversedBy: 'books')]
    #[JoinColumn(name: 'autor', referencedColumnName: 'id', nullable: false)]
    private Author $author;

}
```

Entidad Author.php

```
namespace App\Entity;

use …

#[Entity(repositoryClass: AuthorRepository::class)]
#[Table(name: 'autores')]
class Author
{
    #[Id]
    #[GeneratedValue]
    #[Column(name: 'id', type: 'integer')]
    private int $id;

    #[Column(name: 'nombre', type: 'string', length: 255)]
    private string $name;

    #[Column(name: 'nacionalidad', type: 'string', length: 255)]
    private string $nationality;

    #[OneToMany(targetEntity: Book::class, mappedBy: 'author')]
    private Collection $books;

    public function __construct()
    {
        $this->books = new ArrayCollection();
    }
}
```

Ejercicio 2. Sistema de Reserva de Habitaciones

Modela un sistema de reservas donde:

1. **Entidad Habitación:**
    - Campos:
        - id (entero)
        - numero (entero único)
        - precio (decimal)
2. **Entidad Reserva:**
    - Campos:
        - id (entero)
        - fecha_inicio (fecha)
        - fecha_fin (fecha)
        - estado (ENUM con valores: "pendiente", "confirmada", "cancelada")
3. **Relaciones: (bidireccional)**
    - Cada **Habitacion puede tener muchas Reservas**
    - Cada **Reserva pertenece a una Habitacion**

Tareas:

- Implementar entidades en Doctrine con los tipos de datos correspondientes.
- Usar un campo ENUM para el estado.

Ejercicio 2. Sistema de Reserva de Habitaciones - SOLUCIÓN

Entidad Room.php

```
namespace App\Entity;

use …

#[Entity(repositoryClass: RoomRepository::class)]
#[Table(name: 'habitaciones')]
class Room
{
    #[Id]
    #[GeneratedValue]
    #[Column(name: 'id', type: 'integer')]
    private int $id;

    #[Column(name: 'numero', type: 'integer', unique: true)]
    private int $number;

    #[Column(name: 'precio', type: 'decimal', precision: 10, scale: 2)]
    private float $price;

    #[OneToMany(targetEntity: Booking::class, mappedBy: 'room')]
    private Collection $bookings;

    public function __construct()
    {
        $this->bookings = new ArrayCollection();
    }
}
```

Entidad Booking.php

```
namespace App\Entity;

use …

#[Entity(repositoryClass: BookingRepository::class)]
#[Table(name: 'reservas')]
class Booking
{
    #[Id]
    #[GeneratedValue]
    #[Column(name: 'id', type: 'integer')]
    private int $id;

    #[Column(name: 'fecha_inicio', type: 'date')]
    private \DateTime $startDate;

    #[Column(name: 'fecha_fin', type: 'date')]
    private \DateTime $endDate;

    #[Column(name: 'estado', type: 'string', columnDefinition:
"ENUM('pendiente','confirmada','cancelada')")]
    private string $status;

    #[ManyToOne(targetEntity: Room::class, inversedBy: 'bookings')]
    #[JoinColumn(name: 'habitacion', referencedColumnName: 'id', nullable:
false)]
    private Room $room;
}
```

Ejercicio 3. Sistema de Cursos

Crea un modelo para un sistema de cursos online:

1. **Entidad Curso:**
    - Campos:
        - id (entero)
        - nombre (cadena de texto)
        - descripcion (texto largo)
        - precio (decimal)
2. **Entidad Estudiante:**
    - Campos:
        - id (entero)
        - nombre (cadena de texto)
        - email (cadena única)
3. **Relaciones: (bidireccional)**
    - Cada **Curso puede tener muchos Estudiantes y viceversa.**

Tareas:

- Implementar una relación ManyToMany con una tabla intermedia usando Doctrine.

Ejercicio 3. Sistema de Cursos - SOLUCIÓN

Entidad Course.php

```
namespace App\Entity;

use …

#[Entity(repositoryClass: CourseRepository::class)]
#[Table(name: 'cursos')]
class Course
{
    #[Id]
    #[GeneratedValue]
    #[Column(name: 'id', type: 'integer')]
    private int $id;

    #[Column(name: 'nombre', type: 'string', length: 255)]
    private string $name;

    #[Column(name: 'descripcion', type: 'text')]
    private string $description;

    #[Column(name: 'precio', type: 'decimal', precision: 10, scale: 2)]
    private float $price;

    #[ManyToMany(targetEntity: Student::class, mappedBy: 'courses')]
    private Collection $students;

    public function __construct()
    {
        $this->students = new ArrayCollection();
    }
}
```

Entidad Student.php

```
namespace App\Entity;

use …

#[Entity(repositoryClass: StudentRepository::class)]
#[Table(name: 'estudiantes')]
class Student
{
    #[Id]
    #[GeneratedValue]
    #[Column(name: 'id', type: 'integer')]
    private int $id;

    #[Column(name: 'nombre', type: 'string', length: 255)]
    private string $name;

    #[Column(name: 'email', type: 'string', length: 255, unique: true)]
    private string $email;

    #[ManyToMany(targetEntity: Course::class, inversedBy: 'students')]
    #[JoinTable(name: 'cursos_estudiantes')]
    private Collection $courses;

    public function __construct()
    {
        $this->courses = new ArrayCollection();
    }
}
```

Ejercicio 4. Sistema de Perfiles de Usuarios

Crea un modelo donde cada usuario tiene un perfil asociado:

1. **Entidad Usuario:**
    - Campos:
        - id (entero)
        - username (cadena única)
        - password (cadena de texto)
2. **Entidad Perfil:**
    - Campos:
        - id (entero)
        - biografia (texto)
        - foto (cadena, ruta al archivo)
3. **Relaciones: (unidireccional)**
    - Cada **Usuario tiene un Perfil asociado.**

Objetivo:

- Crear la relación OneToOne en Doctrine.
- Usar la clave foránea en la tabla de Perfil.

Ejercicio 4. Sistema de Perfiles de Usuarios - SOLUCIÓN

Entidad User.php

```
namespace App\Entity;

use App\Repository\UserRepository;

#[Entity(repositoryClass: UserRepository::class)]
#[Table(name: 'usuarios')]
class User
{
    #[Id]
    #[GeneratedValue]
    #[Column(name: 'id', type: 'integer')]
    private int $id;

    #[Column(name: 'username', type: 'string', length: 255, unique: true)]
    private string $username;

    #[Column(name: 'password', type: 'string', length: 255)]
    private string $password;

    #[OneToOne(targetEntity: Profile::class)]
    #[JoinColumn(name: 'perfil', referencedColumnName: 'id', nullable: false)]
    private Profile $profile;
}
```

Entidad Profile.php

```
namespace App\Entity;

use …

#[Entity(repositoryClass: ProfileRepository::class)]
#[Table(name: 'perfiles')]
class Profile
{
    #[Id]
    #[GeneratedValue]
    #[Column(name: 'id', type: 'integer')]
    private int $id;

    #[Column(name: 'biografia', type: 'text')]
    private string $biography;

    #[Column(name: 'foto', type: 'string', length: 255)]
    private string $photo;
}
```