```
<?php
// ReservaDAO.php - Implementación del DAO para la entidad Reserva
require_once dirname(__DIR__, 2) . '/config/Database.php';

class ReservaDAO {
private $db;

public function __construct() {
$this->db = Database::getInstance()->getConnection();
}

public function crearReserva($datos) {
// Generar un localizador único para cada reserva
if (!isset($datos['localizador'])) {
$datos['localizador'] = uniqid('RES-', true);
}

// Asignar la fecha de la reserva y la fecha de modificación
if (!isset($datos['fecha_reserva'])) {
$datos['fecha_reserva'] = date('Y-m-d H:i:s');
}

if (!isset($datos['fecha_modificacion'])) {
$datos['fecha_modificacion'] = date('Y-m-d H:i:s');
}

// Verificar los campos obligatorios según el tipo de reserva
if ($datos['id_tipo_reserva'] == 1 || $datos['id_tipo_reserva'] == 3) { // Aeropuerto a Hotel o Ida y Vuelta
if (empty($datos['fecha_entrada'])) {
throw new Exception('El campo "fecha_entrada" es obligatorio para el trayecto Aeropuerto a Hotel.');
}
if (empty($datos['hora_entrada'])) {
```

```
throw new Exception('El campo "hora_entrada" es obligatorio para el trayecto Aeropuerto a Hotel.');

}

if (empty($datos['numero_vuelo'])) {

throw new Exception('El campo "numero_vuelo" es obligatorio para el trayecto Aeropuerto a Hotel.');

}

if (empty($datos['origen_vuelo_entrada'])) {

throw new Exception('El campo "origen_vuelo_entrada" es obligatorio para el trayecto Aeropuerto a Hotel.');

}

}

if ($datos['id_tipo_reserva'] == 2 || $datos['id_tipo_reserva'] == 3) { // Hotel a Aeropuerto o Ida y Vuelta

if (empty($datos['fecha_vuelo_salida'])) {

throw new Exception('El campo "fecha_vuelo_salida" es obligatorio para el trayecto Hotel a Aeropuerto.');

}

if (empty($datos['hora_vuelo_salida'])) {

throw new Exception('El campo "hora_vuelo_salida" es obligatorio para el trayecto Hotel a Aeropuerto.');

}

if (empty($datos['numero_vuelo'])) {

throw new Exception('El campo "numero_vuelo" es obligatorio para el trayecto Hotel a Aeropuerto.');

}

if (empty($datos['hora_recogida'])) {

throw new Exception('El campo "hora_recogida" es obligatorio para el trayecto Hotel a Aeropuerto.');

}

}
```

```
// Preparar la consulta
$sql = "INSERT INTO transfer_reservas (
localizador, id_hotel, id_tipo_reserva, email_cliente,
fecha_reserva, fecha_modificacion, id_destino, fecha_entrada, hora_entrada,
numero_vuelo, origen_vuelo_entrada, fecha_vuelo_salida, hora_vuelo_salida,
hora_recogida, num_viajeros, id_vehiculo
) VALUES (
:localizador, :id_hotel, :id_tipo_reserva, :email_cliente,
:fecha_reserva, :fecha_modificacion, :id_destino, :fecha_entrada, :hora_entrada,
:numero_vuelo, :origen_vuelo_entrada, :fecha_vuelo_salida, :hora_vuelo_salida,
:hora_recogida, :num_viajeros, :id_vehiculo
)";

$stmt = $this->db->prepare($sql);

// Asignar valores a los parámetros, si no están definidos se asigna null
$stmt->bindValue(':localizador', $datos['localizador']);
$stmt->bindValue(':id_hotel', $datos['id_hotel']);
$stmt->bindValue(':id_tipo_reserva', $datos['id_tipo_reserva']);
$stmt->bindValue(':email_cliente', $datos['email_cliente']);
$stmt->bindValue(':fecha_reserva', $datos['fecha_reserva']);
$stmt->bindValue(':fecha_modificacion', $datos['fecha_modificacion']);
$stmt->bindValue(':id_destino', !empty($datos['id_destino']) ? $datos['id_destino'] : null,
PDO::PARAM_INT);

// Solo asignar valores si están definidos según el tipo de reserva
$stmt->bindValue(':fecha_entrada', ($datos['id_tipo_reserva'] == 1 || $datos['id_tipo_reserva'] ==
3) && !empty($datos['fecha_entrada']) ? $datos['fecha_entrada'] : null, PDO::PARAM_STR);
$stmt->bindValue(':hora_entrada', ($datos['id_tipo_reserva'] == 1 || $datos['id_tipo_reserva'] == 3)
&& !empty($datos['hora_entrada']) ? $datos['hora_entrada'] : null, PDO::PARAM_STR);
$stmt->bindValue(':numero_vuelo', !empty($datos['numero_vuelo']) ? $datos['numero_vuelo'] :
null, PDO::PARAM_STR);
$stmt->bindValue(':origen_vuelo_entrada', ($datos['id_tipo_reserva'] == 1 ||
```

```
$datos['id_tipo_reserva'] == 3) && !empty($datos['origen_vuelo_entrada']) ?
$datos['origen_vuelo_entrada'] : null, PDO::PARAM_STR);

$stmt->bindValue(':fecha_vuelo_salida', ($datos['id_tipo_reserva'] == 2 || $datos['id_tipo_reserva']
== 3) && !empty($datos['fecha_vuelo_salida']) ? $datos['fecha_vuelo_salida'] : null,
PDO::PARAM_STR);

$stmt->bindValue(':hora_vuelo_salida', ($datos['id_tipo_reserva'] == 2 || $datos['id_tipo_reserva']
== 3) && !empty($datos['hora_vuelo_salida']) ? $datos['hora_vuelo_salida'] : null,
PDO::PARAM_STR);

$stmt->bindValue(':hora_recogida', ($datos['id_tipo_reserva'] == 2 || $datos['id_tipo_reserva'] ==
3) && !empty($datos['hora_recogida']) ? $datos['hora_recogida'] : null, PDO::PARAM_STR);

$stmt->bindValue(':num_viajeros', $datos['num_viajeros'], PDO::PARAM_INT);

$stmt->bindValue(':id_vehiculo', !empty($datos['id_vehiculo']) ? $datos['id_vehiculo'] : null,
PDO::PARAM_INT);

return $stmt->execute();

}

public function obtenerReservaPorId($id) {

$query = "SELECT * FROM transfer_reservas WHERE id_reserva = :id";

$stmt = $this->db->prepare($query);

$stmt->bindParam(':id', $id, PDO::PARAM_INT);

$stmt->execute();

return $stmt->fetch(PDO::FETCH_ASSOC);

}

public function modificarReserva($id, $datos) {

$query = "UPDATE transfer_reservas

SET fecha_entrada = :fecha_entrada,

hora_entrada = :hora_entrada,

origen_vuelo_entrada = :origen_vuelo_entrada,

fecha_vuelo_salida = :fecha_vuelo_salida,

hora_vuelo_salida = :hora_vuelo_salida,

hora_recogida = :hora_recogida,
```

```
num_viajeros = :num_viajeros,
id_vehiculo = :id_vehiculo,
fecha_modificacion = NOW()
WHERE id_reserva = :id";

$stmt = $this->db->prepare($query);
$stmt->bindParam(':id', $id, PDO::PARAM_INT);
$stmt->bindParam(':fecha_entrada', $datos['fecha_entrada']);
$stmt->bindParam(':hora_entrada', $datos['hora_entrada']);
$stmt->bindParam(':origen_vuelo_entrada', $datos['origen_vuelo_entrada']);
$stmt->bindParam(':fecha_vuelo_salida', $datos['fecha_vuelo_salida']);
$stmt->bindParam(':hora_vuelo_salida', $datos['hora_vuelo_salida']);
$stmt->bindParam(':hora_recogida', $datos['hora_recogida']);
$stmt->bindParam(':num_viajeros', $datos['num_viajeros'], PDO::PARAM_INT);
$stmt->bindParam(':id_vehiculo', $datos['id_vehiculo'], PDO::PARAM_INT);
return $stmt->execute();
}

public function eliminarReserva($id) {
// Verificar que el ID sea válido
if (empty($id) || !is_numeric($id)) {
throw new InvalidArgumentException("ID inválido proporcionado para eliminar la reserva.");
}

// Preparar la consulta SQL
$sql = "DELETE FROM transfer_reservas WHERE id_reserva = :id";
$stmt = $this->db->prepare($sql);

// Ejecutar y verificar resultados
if ($stmt->execute(['id' => $id])) {
if ($stmt->rowCount() > 0) {
// Confirmar que se eliminó un registro
```

```
return true;
} else {
// El registro no existe
throw new Exception("No se encontró ninguna reserva con ID: $id.");
}
} else {
// Depurar errores de SQL
$errorInfo = $stmt->errorInfo();
throw new Exception("Error al ejecutar la consulta: " . implode(', ', $errorInfo));
}
}

public function obtenerTodasLasReservas() {
$sql = "SELECT * FROM transfer_reservas";
$stmt = $this->db->query($sql);
return $stmt->fetchAll(PDO::FETCH_ASSOC) ?: [];
}

}
?>
```