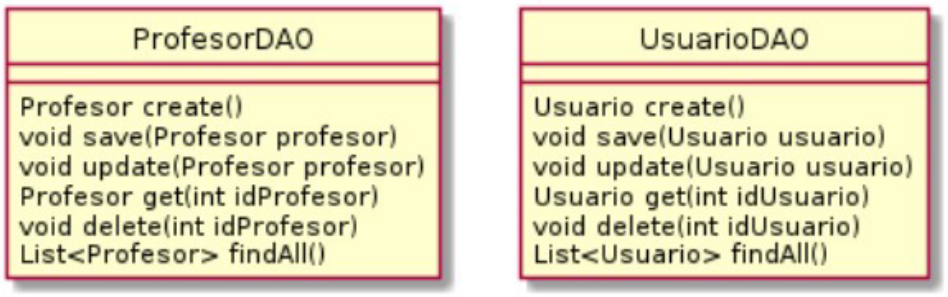
PHP DAO

El patrón DAO permite ser más organizado con las consultas sql. Según wikipedia:

En software de computadores, un objeto de acceso a datos (en inglés, data access object, abreviado DAO) es un componente de software que suministra una interfaz común entre la aplicación y uno o más dispositivos de almacenamiento de datos, tales como una Base de datos o un archivo.

El modelo DAO podría incluso tomar por cada clase que necesitamos crear una clase equivalente DAO. Por ejemplo, supongamos que nuestra aplicación usará dos clases que queremos persistir. Una se llama Profesor y la otra Usuario entonces el patrón DAO nos podría dar:



Si nos fijamos, entonces con la línea:

```
$profesorDAO.update( $profesor )
```

estaríamos en ese método update() tocando la base de datos para actualizar la información del objeto: profesor que tenemos en memoria

Se propone pues, crear un objeto DAO por cada una de las tablas/objetos necesarias. En nuestra aplicación precisaríamos como mínimo:

AlumnoDAO, AsignaturaDAO, MatriculaDAO

Si queremos que nuestras clases DAO sigan el mismo patrón podemos apoyarnos en un interfaz:

```
interface DAO{
        public function findAll();
        function save($dao);
        function findById($id);
        function update($dao);
        function delete($id);

}
```

Veamos como quedaría el prototipo de PersonaDAO entonces (únicamente vemos el código de dos de los métodos)

```
class PersonaDAO implements DAO{
    private $myPDO;
    public function __construct($pdo){
        $this->myPDO = $pdo;

    }

    public function findAll(){
        $stmt = $this->myPDO->prepare("SELECT * FROM
".PersonaContract::TABLE_NAME);

        $stmt->setFetchMode(PDO::FETCH_ASSOC);
        $stmt->execute();
        $personas = [];

        while ($row = $stmt->fetch()){
            $p = new Persona();
            $p->setId($row["id"])
            ->setNombre($row["nombre"])
            ->setEdad($row["edad"]);

            $personas[] = $p;

        }

        return $personas;
    }
```

```
    public function save( $p){
        $tablename = PersonaContract::TABLE_NAME;
        $colid = PersonaContract::COL_ID;
        $coledad = PersonaContract::COL_EDAD;
        $colnombre = PersonaContract::COL_NOMBRE;

        $sql = "INSERT INTO $tablename ($colid, $colnombre, $coledad)
VALUES(:id,:nombre, :edad)";

        $this->myPDO->beginTransaction();
        $stmt = $this->myPDO->prepare($sql);
        $stmt->execute(
            [
            ':id' => $p->id,
            ':nombre' => $p->nombre,
            ":edad" => $p->edad
            ]
        );
        //$this->myPDO->rollback();
        $this->myPDO->commit();
        $stmt = null;

    }
```

Observar que tiene soporte para preparedstatement y podemos establecer los parámetros con etiquetas con dos puntos: `VALUES(:id,:nombre, :edad)`

También vemos que hay soporte para transacciones: `$pdo→beginTransaction()`, `$pdo→rollback()`, `$pdo→ commit()`

Para crear un nuevo objeto haríamos desde el controller:

(si es php sin framework)

```
            $dsn = "mysql:host=localhost;dbname=gente";
            $user = "root";
            $password = "1q2w3e4";
            new PDO($dsn, $user, $password);

            $personaDAO =new PersonaDAO($pdo);
```

(si es con laravel)

```
$pdo = DB::connection()->getPdo();
$personaDAO =new PersonaDAO($pdo);
```

Para que funcione lo anterior con laravel, tenemos que especificar en el fichero .env los parámetros de la base de datos (el fichero está en la carpeta raíz del proyecto)

```
DB_CONNECTION=mysql
DB_HOST=127.0.0.1
DB_PORT=3306
DB_DATABASE=gente
DB_USERNAME=root
DB_PASSWORD=1q2w3e4r
```

Las clases contract son las habituales:

```
class PersonaContract{
        public const TABLE_NAME="personas";
        public const COL_ID="id";
        public const COL_NOMBRE="nombre";
        public const COL_EDAD="edad";
}
```