

# Databases

Data persistence



# Què veurem?

- Repàs de MySQL
- Connexió a BD amb PHP

# Introducció

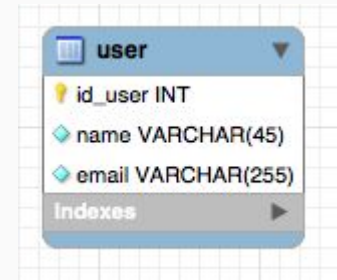
- Una base de dades està formada per taules
- Cada taula recull informació d'una entitat del nostre sistema
- Cada taula es compon per columnes i files

# Exemple d'une classe en PHP

```
/**
 * User
 */
class User
{
    private $id;
    private $name;
    private $email;

    function __construct($id, $name, $email)
    {
        $this->id = $id;
        $this->name = $name;
        $this->email = $email;
    }

    /**
     * Getters and setters
     * ...
     */
}
```



# SQL

- És el llenguatge més estàndard per treballar amb bases de dades
- Permet consultar, inserir, esborrar i actualitzar informació d'una BD
- **SQL** s'utilitza en bases de dades relacionals
- Existeixen bases de dades NoSQL com per exemple **MongoDB**

# Queries

Una query és una petició a una base de dades:

```
SELECT * FROM user;
```

# Creació d'una base de dades

- Per crear una nova base de dades podem fer servir la comanda:

**CREATE DATABASE db\_name;**

- Un cop creada, hem d'indicar la BD que volem fer servir:

**USE db\_name;**

# Creació d'una taula

Un cop seleccionada la base de dades, **pw** en aquest cas, podem afegir una nova taula seguint el següent exemple:

```
CREATE TABLE pw.`user` (  
  id INT NOT NULL AUTO_INCREMENT,  
  name varchar(100) NOT NULL,  
  email varchar(100) NOT NULL,  
  created_at DATETIME NOT NULL,  
  updated_at DATETIME NOT NULL,  
  CONSTRAINT user_PK PRIMARY KEY (id)  
)  
ENGINE=InnoDB  
DEFAULT CHARSET=utf8  
COLLATE=utf8_general_ci;
```



# Insert

- Partint de l'exemple anterior, podem afegir nous registres amb la comanda INSERT INTO.

```
INSERT INTO table(`column_1`, `column_2` ...) VALUES ("value_1",  
"value_2" ...)
```

# Delete

- Per tal d'eliminar un o més registres fem servir la comanda DELETE.

**DELETE FROM table WHERE [condition]**

- En cas de no especificar cap condició, afectarà a tots els registres de la taula.

# Update

- Per actualitzar un o més registres de la taula podem fer servir la comanda UPDATE.

**UPDATE table SET column="new value" WHERE [condition]**

# Select

- Per recuperar informació fem servir la comanda SELECT.

**SELECT \* FROM table**

**SELECT \* FROM table WHERE [condition]**

**SELECT DISTINCT column FROM table**

# PHP and MySQL


# Introducció

- A PHP disposem de diferents opcions per connectar-nos a una base de dades:
  - [PDO](#)
  - [mysqli](#)

# PDO

- Extensió de PHP
- Disposa de múltiples [drivers](#)
- En el nostre cas farem servir MySQL

# PDO



```
$connection = new PDO('mysql:host=localhost;dbname=test', 'user', 'password');
```



# PDO

Un cop creada la connexió tenim diverses opcions per fer queries a la base de dades. La més simple és fent servir la funció **query**.

# PDO



```
$connection = new PDO('mysql:host=localhost;dbname=test', 'user', 'password');  
$rows = $connection->query('SELECT * FROM table_name');  
foreach ($rows as $row) {  
    echo $row['field_name'];  
}
```

# PDO

El resultat de la query és un objecte de tipus [PDOStatement](#)

```
/var/www/projectesweb/php/database/index.php:5:  
object(PDOStatement)[2]  
  public 'queryString' => string 'SELECT * FROM user' (length=18)
```

# PDO

Podem especificar com volem rebre les dades per cada **query** que executem.



```
$connection = new PDO('mysql:host=localhost;dbname=test', 'user', 'password');  
$statement = $connection->query('SELECT * FROM table_name');  
$results = $statement->fetchAll(PDO::FETCH_ASSOC);
```

# PDO

Podem configurar com volem que **PDO** recorri les files obtingudes d'una cerca.

```
<?php
$db = new PDO('mysql:host=localhost;dbname=test', 'homestead', 'secret');
$db->setAttribute(PDO::ATTR_DEFAULT_FETCH_MODE, PDO::FETCH_ASSOC);
```

# SQL Injection

Si permetem que els usuaris puguin enviar paràmetres directament en les nostres queries, estem exposant la nostra aplicació, com per exemple a [SQL injection](#).

```
<?php
$db = new PDO('mysql:host=localhost;dbname=test', 'homestead', 'secret');
$id = " 1 OR 1 = 1;";
$query = "SELECT * FROM user WHERE id = $id";
$statement = $db->query($query);
$results = $statement->fetchAll(PDO::FETCH_ASSOC);
```

# Prepared statements

Guardem les *queries* per a executar-les múltiples vegades canviant únicament els paràmetres que rep.

```
<?php
$db = new PDO('mysql:host=localhost;dbname=test', 'homestead', 'secret');
$email = 'jaume@test.dev';
$name = 'Jaume';
// -----
$stmt = $db->prepare("SELECT * FROM user WHERE name=? AND email=? ");
$stmt->bindParam(1, $name, PDO::PARAM_STR);
$stmt->bindParam(2, $email, PDO::PARAM_STR);
$stmt->execute();
$results = $stmt->fetchAll(PDO::FETCH_ASSOC);
// -----
$stmt = $db->prepare("SELECT * FROM user WHERE name=:name AND email=:email
");
$stmt->bindParam(':name', $email, PDO::PARAM_STR);
$stmt->bindParam(':email', $password, PDO::PARAM_STR);
$stmt->execute();
$results = $stmt->fetchAll(PDO::FETCH_ASSOC);
```

# Filtres

- Disposem de diverses funcions en PHP per filtrar i validar les dades rebudes per part dels usuaris.
- Les més utilitzades són **filter\_var** i **filter\_input**.



# Validacions

Podem comprovar si la informació rebuda compleix certs requeriments:

```
<?php
$validEmail = 'joe@example.com';
$invalidEmail = 'test';

var_dump(filter_var($validEmail, FILTER_VALIDATE_EMAIL)); // Prints
joe@example.com
var_dump(filter_var($invalidEmail, FILTER_VALIDATE_EMAIL)); // Prints false
```

# Sanejar

Podem modifica la informació eliminant els caràcters invàlids:

```
<?php
$email = '(joe@example.com)';
var_dump(filter_var($invalidEmail, FILTER_SANITIZE_EMAIL)); // Prints
joe@example.com
```

# Hands on

- Implementar els [exercicis](#) 10 - 11.

# In the next episode

- Com gestionar la sessió d'un usuari
- Com gestionar les cookies del navegador