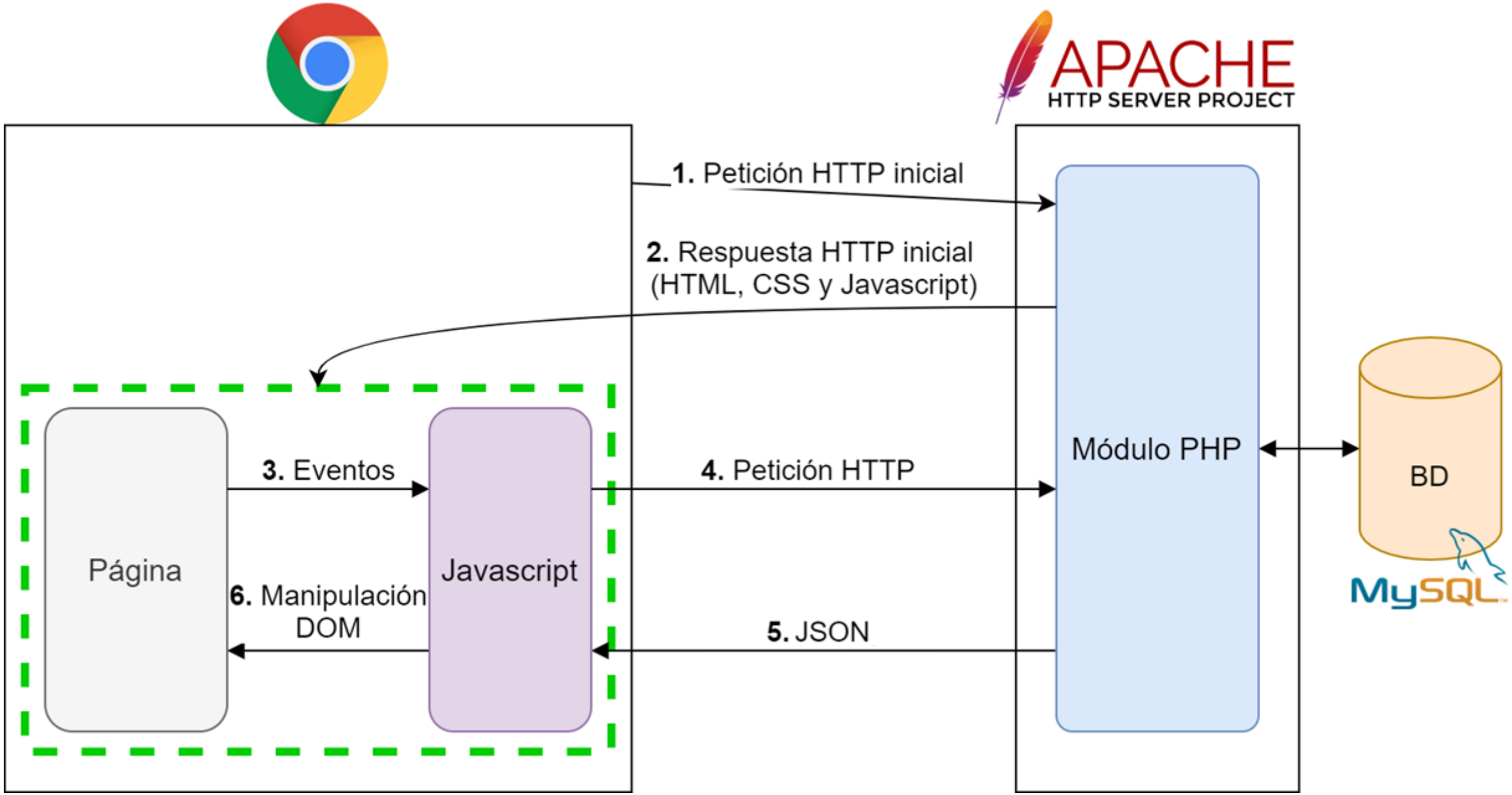
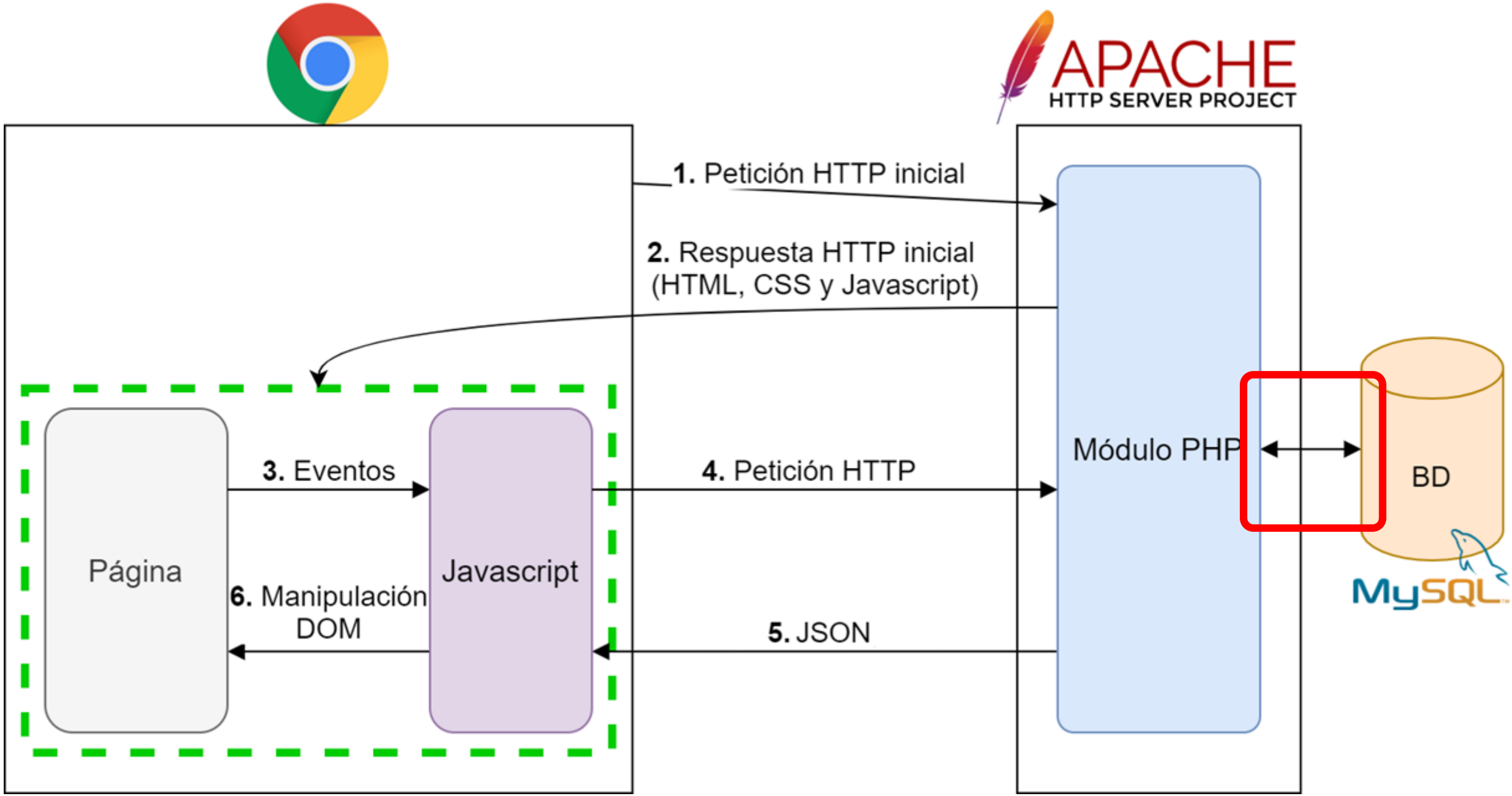
UD7. Mapeo objeto-relacional (ORM)

DESARROLLO WEB EN ENTORNO SERVIDOR


APACHE
HTTP SERVER PROJECT
1. Petición HTTP inicial
2. Respuesta HTTP inicial
(HTML, CSS y Javascript)
Página
3. Eventos
Javascript
4. Petición HTTP
Módulo PHP
BD
MySQL
6. Manipulación
DOM
5. JSON


APACHE
HTTP SERVER PROJECT
1. Petición HTTP inicial
2. Respuesta HTTP inicial
(HTML, CSS y Javascript)
Página
3. Eventos
Javascript
4. Petición HTTP
Módulo PHP
BD
MySQL
6. Manipulación
DOM
5. JSON

Hasta ahora…

- Escribíamos sentencias SQL manualmente:

```
$usuario = "SELECT * FROM restaurantes WHERE cod=3";
$resultado = $bd->query($usuario);
```

- Inconvenientes:

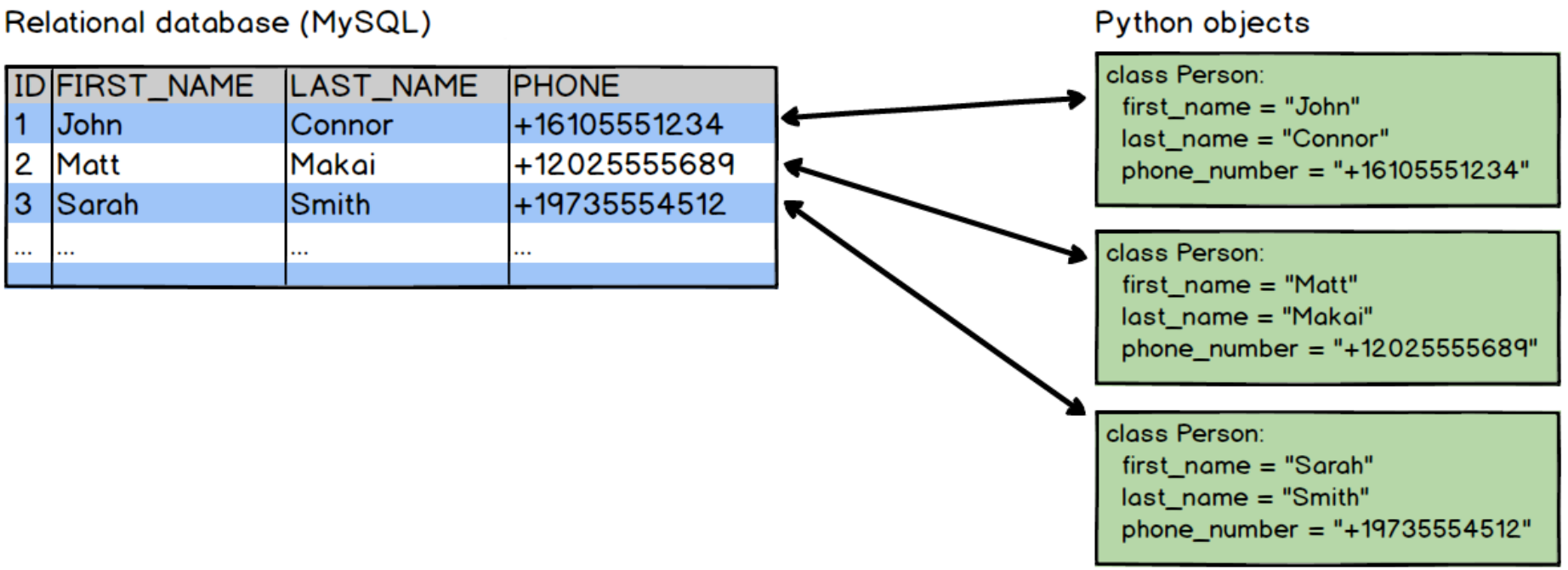
Hasta ahora…

- Escribíamos sentencias SQL manualmente:

```
$usuario = "SELECT * FROM restaurantes WHERE cod=3";
$resultado = $bd->query($usuario);
```

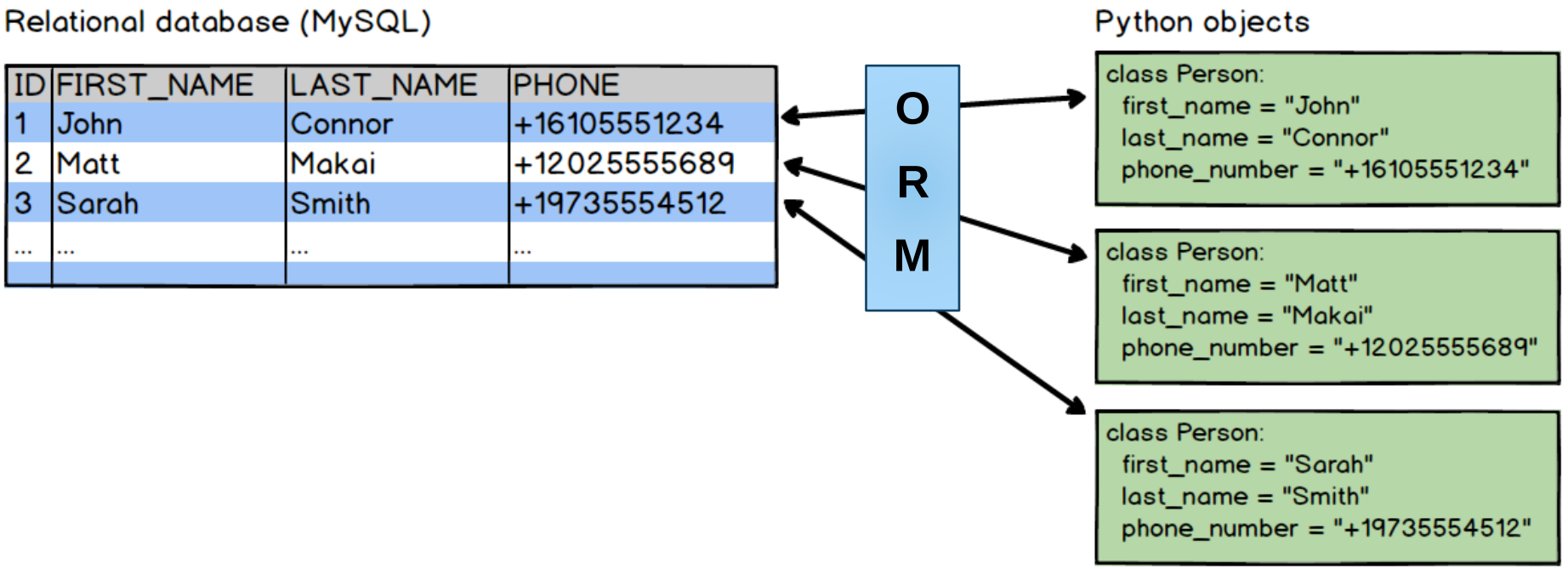
- Inconvenientes:
  - Es fácil cometer errores.
  - Se escribe código repetido en varios puntos de la aplicación.
  - Dependencia de la sintaxis SQL particular de cada BD.

Mapeo objeto-relacional (ORM)



Mapeo objeto-relacional (ORM)

- Técnica para asociar los elementos de la base de datos con los objetos de la aplicación.



Mapeo objeto-relacional (ORM)

- No manipulamos directamente la BD, lo hacemos usando los objetos.
- El ORM se encargará de ejecutar las sentencias SQL necesarias.
- Ejemplo:

```
$person->setPhoneNumber("600112233");
$entityManager->flush();
```

Mapeo objeto-relacional (ORM)

- NO es una técnica exclusiva del desarrollo web.
- El software que implementa la técnica se llama también ORM.
- Algunos:

Doctrine: instalación

Doctrine

- ORM para PHP.
- Basado en Hibernate.
- De código libre.

Instalación de Doctrine

1. Instala Composer (https://getcomposer.org/) indicándole que php.exe está en C:\xampp\php\php.exe
2. Abre cmd.exe
3. Crea el directorio C:\xampp\htdocs\doctrine
4. Entra en el directorio C:\xampp\htdocs\doctrine con `cmd`.
5. Pide a Composer que instale Doctrine:

```
composer require doctrine/orm
composer require symfony/cache
```

- Composer actualizará el fichero C:\xampp\htdocs\doctrine\vendor\autoload.php
- Al incluir el fichero con `include`, ya puedes usar Doctrine

Ejemplos

Utilizan la base de datos “**doctrine**” con las siguientes tablas:

Equipo(**Id**, Nombre, Socios, Fundación, Ciudad)

Jugador(**Id**, Nombre, Apellidos, Edad, **Equipo**)

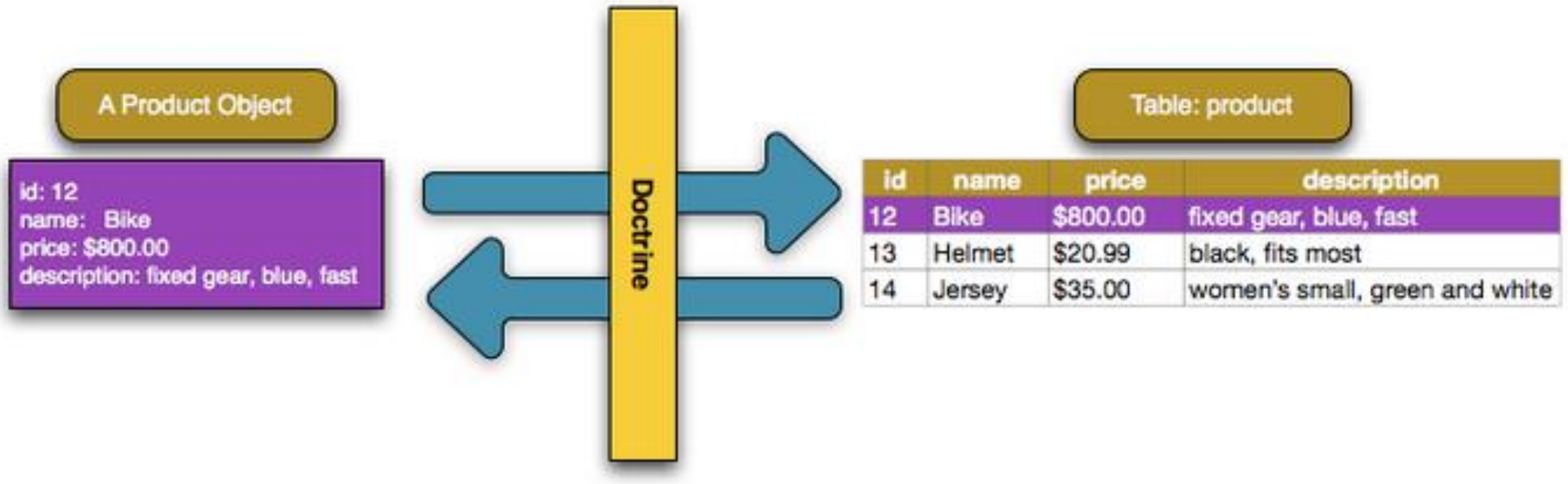
Doctrine: configuración y crear entidades

Configuración

- Doctrine requiere una inicialización.
- La vamos a delegar en un fichero llamado `bootstrap.php`
- Tenemos que incluirlo en todas las páginas que vayan a usar la BD.
- Se utiliza para:
  - Configurar la conexión con la base de datos.
  - Indicar cómo tiene que "asociar" las clases a las tablas de la BD.
  - Resultado: objeto `EntityManager`, interfaz que utilizará nuestra aplicación.

Entidades

Es el nombre que se da a **las clases** que están mapeadas con una tabla de la base de datos.



`src/Equipo.php`

Entidades (II)

¿Cómo escribir una entidad?

1. 1 clase-entidad representa 1 tabla de la BD.
2. 1 atributo de la clase representa 1 columna de la tabla.
3. Los atributos son `protected` o `private`.
4. Se crea un setter y getter para cada uno.
   - …salvo para los campos autogenerados (AUTO_INCREMENT), que no tienen por qué tener setters.

src/Equipo.php

Entidades

¿Y cómo sabe Doctrine qué relación hay entre una clase y una tabla de la BD?

Entidades

¿Y cómo sabe Doctrine qué relación hay entre una clase y una tabla de la BD? Hay que añadir metadatos a la clase:

- XML.
- YAML.
- ~~Añadir anotaciones PHP al fichero de la clase.~~
- Añadir atributos PHP 8 al fichero de la clase.

src/Equipo.php

Atributos (I)

- Sintaxis: #[…]
- NO son comentarios.
- Es una manera de indicar configuraciones.
- Aportan metainformación.
- Afectan al elemento que tienen JUSTO DEBAJO.

```
#[ORM\Entity]
#[ORM\Table(name: 'products')]
class Product {
    …
}
```

src/Equipo.php
ejemplos_basicos.php

Atributos (II)

¿Qué podemos escribir para configurar una entidad en Doctrine?

- `#[ORM\Entity]`: declara una clase como entidad.
- `#[ORM\Table]`: mapea una clase con una tabla.
- `#[ORM\Column]`: mapea un atributo con una columna de la tabla.
  - `type`: atributo que indica el tipo de dato (`string`, `integer`, etc.).
  - `name`: atributo para indicar el nombre de la columna (se omite si coincide).
- `#[ORM\Id]`: indica que un atributo es clave primaria.
- `#[ORM\GeneratedValue]`: indica que un atributo es autoincremental.
- Las relaciones que hay con otras clases (asociaciones).

https://www.doctrine-project.org/projects/doctrine-orm/en/2.17/reference/basic-mapping.html

Doctrine: usar entidades

Uso de las entidades

1. Recuperar un objeto y modificarlo

   `ejemplos_basicos.php`

2. Crear un objeto en la BD:

   `crear_equipo.php`

3. Borrar un objeto de la BD:

   `borrar_equipo.php`

Búsquedas: por clave primaria

Se llama a:

```
$entityManager->find("NombreEntidad", clave);
```

- Devuelve un objeto de la entidad “NombreEntidad”
- Que tenga la clave primaria indicada por “clave”.

findBy.php

Búsquedas: por otro atributo

Para búsquedas más complejas, obligatorio usar `EntityRepository`.

1. Obtener el repositorio de la entidad:

```
$rep = $entityManager->getRepository("Jugador");
```

2. Ejecutar una búsqueda:
   - `$rep->findBy($arr)`. Criterios de búsqueda en base a un array.
   - `$rep->findOneBy($arr)`. Como el anterior, pero sólo devuelve 1 resultado.
   - `$rep->findAll()`. Devuelve todos los objetos.

Búsquedas: por si no queda claro...