ar y PHP

ular es un framework para el desarrollo de
na sola página. Una situación muy común es tener
servidor de internet los datos que se ingresan en la

logías para procesar los datos que envía y recibe
y mediante este acceder a una base de datos

ue debemos desarrollar tanto en el cliente

rtículos, cada artículo almacena el código,
os datos de un artículo almacenados en una base

el proyecto022:

elegar todas las responsabilidades de acceso a
tras clases que colaboran con las componentes.

LI:

Con el comando anterior estamos creando la clase 'ArticulosService'

El código que debemos guardar en el archivo 'articulos.service.ts' es:

```
import { Injectable } from '@angular/core';
import { HttpClient } from '@angular/common/http';

@Injectable({
  providedIn: 'root'
})
export class ArticulosService {

  url='https://ejerciciostutorialesya.com/angular/proyecto016/'; // disponer url de su servidor q
ue tiene las páginas PHP

  constructor(private http: HttpClient) { }

  recuperarTodos() {
    return this.http.get(`${this.url}recuperartodos.php`);
  }

  alta(articulo:any) {
    return this.http.post(`${this.url}alta.php`, JSON.stringify(articulo));
  }

  baja(codigo:number) {
    return this.http.get(`${this.url}baja.php?codigo=${codigo}`);
  }

  seleccionar(codigo:number) {
    return this.http.get(`${this.url}seleccionar.php?codigo=${codigo}`);
  }

  modificacion(articulo:any) {
    return this.http.post(`${this.url}modificacion.php`, JSON.stringify(articulo));
  }
}
```

Presentaremos primero todos los archivos y luego explicaremos como se relacionan.

3. Debemos ahora modificar el archivo 'app.config.ts' para poder utilizar la función provideHttpClient:

```
import { ApplicationConfig, provideZoneChangeDetection } from '@angular/core';
import { provideRouter } from '@angular/router';
import { routes } from './app.routes';
import { provideHttpClient } from '@angular/common/http';
import { withFetch } from '@angular/common/http';

export const appConfig: ApplicationConfig = {
  providers: [provideZoneChangeDetection({ eventCoalescing: true }), provideRouter(routes),
 provideHttpClient(withFetch())]
};
```

4. Nuestra única componente debe implementar el siguiente código en el archivo 'app.component.ts':

```
import { Component } from '@angular/core';
import { RouterOutlet } from '@angular/router';
import { ArticulosService } from './articulos.service';
import { FormsModule } from '@angular/forms';

@Component({
  selector: 'app-root',
  imports: [RouterOutlet, FormsModule],
  templateUrl: './app.component.html',
  styleUrls: ['./app.component.css']
})
export class AppComponent {
  articulos:any;

  art={
    codigo:0,
    descripcion:"",
    precio:0
  }

  constructor(private articulosServicio: ArticulosService) {
    this.recuperarTodos();
  }


  recuperarTodos() {
    this.articulosServicio.recuperarTodos().subscribe((result:any) => this.articulos = result);
  }

  alta() {
    this.articulosServicio.alta(this.art).subscribe((datos:any) => {
      if (datos['resultado']=='OK') {
        alert(datos['mensaje']);
        this.recuperarTodos();
      }
```

```
    });
  }

  baja(codigo:number) {
    this.articulosServicio.baja(codigo).subscribe((datos:any) => {
      if (datos['resultado']=='OK') {
        alert(datos['mensaje']);
        this.recuperarTodos();
      }
    });
  }

  modificacion() {
    this.articulosServicio.modificacion(this.art).subscribe((datos:any) => {
      if (datos['resultado']=='OK') {
        alert(datos['mensaje']);
        this.recuperarTodos();
      }
    });
  }

  seleccionar(codigo:number) {
    this.articulosServicio.seleccionar(codigo).subscribe((result:any) => this.art = result[0]);
  }

}
```

5. Y el archivo 'app.component.html' con:

```
<div>
  <h1>Administración de artículos</h1>
    <table border="1">
      <tr>
        <td>Codigo</td><td>Descripcion</td><td>Precio</td><td>Borrar</td><td>Selecciona
r</td>
      </tr>
      @for(art of articulos;track art.codigo) {
      <tr>
        <td>{{art.codigo}}</td>
        <td>{{art.descripcion}}</td>
        <td>{{art.precio}}</td>
        <td><button (click)="baja(art.codigo)">Borrar?</button></td>
        <td><button (click)="seleccionar(art.codigo)">Seleccionar</button></td>
      </tr>
    }
    </table>
    <div>
      <p>
        descripcion:<input type="text" [(ngModel)]="art.descripcion" />
      </p>
      <p>
        precio:<input type="number" [(ngModel)]="art.precio" />
      </p>
      <p><button (click)="alta()">Agregar</button>
      <button (click)="modificacion()">Modificar</button></p>
    </div>
</div>
<router-outlet />
```

Todos los archivos presentados son los necesarios en Angular 19, veamos ahora que tenemos en PHP y MySQL.

6. Primero debemos crear una base de datos en MySQL llamada 'bd1' y crear la siguiente tabla:

```
CREATE TABLE articulos (
  codigo int AUTO_INCREMENT,
  descripcion varchar(50),
  precio float,
  PRIMARY KEY (codigo)
)
```

7. Tenemos una serie de archivos PHP que reciben datos en formato JSON y retornan también un JSON.

   El archivo 'recuperartodos.php' retorna en formato JSON todos los artículos:

```php
<?php
  header('Access-Control-Allow-Origin: *');
  header("Access-Control-Allow-Headers: Origin, X-Requested-With, Content-Type, Accept")
;

  require("conexion.php");
  $con=retornarConexion();

  $registros=mysqli_query($con,"select codigo, descripcion, precio from articulos");
  $vec=[];
  while ($reg=mysqli_fetch_array($registros))
  {
    $vec[]=$reg;
  }

  $cad=json_encode($vec);
  echo $cad;
  header('Content-Type: application/json');
?>
```

8. El archivo 'alta.php' recibe los datos en formato JSON y los almacena en la tabla:

```php
<?php
  header('Access-Control-Allow-Origin: *');
  header("Access-Control-Allow-Headers: Origin, X-Requested-With, Content-Type, Accept")
;

  $json = file_get_contents('php://input');

  $params = json_decode($json);

  require("conexion.php");
  $con=retornarConexion();


  mysqli_query($con,"insert into articulos(descripcion,precio) values
                ('$params->descripcion',$params->precio)");


  class Result {}

  $response = new Result();
  $response->resultado = 'OK';
  $response->mensaje = 'datos grabados';

  header('Content-Type: application/json');
  echo json_encode($response);
?>
```

9. El archivo 'baja.php':

```php
<?php
  header('Access-Control-Allow-Origin: *');
  header("Access-Control-Allow-Headers: Origin, X-Requested-With, Content-Type, Accept")
;

  require("conexion.php");
  $con=retornarConexion();

  mysqli_query($con,"delete from articulos where codigo=$_GET[codigo]");


  class Result {}

  $response = new Result();
  $response->resultado = 'OK';
  $response->mensaje = 'articulo borrado';

  header('Content-Type: application/json');
  echo json_encode($response);
?>
```

10. El archivo 'modificacion.php':

```php
<?php
  header('Access-Control-Allow-Origin: *');
  header("Access-Control-Allow-Headers: Origin, X-Requested-With, Content-Type, Accept")
;

  $json = file_get_contents('php://input');

  $params = json_decode($json);

  require("conexion.php");
  $con=retornarConexion();


  mysqli_query($con,"update articulos set descripcion='$params->descripcion',
                                      precio=$params->precio
                                      where codigo=$params->codigo");


  class Result {}

  $response = new Result();
  $response->resultado = 'OK';
  $response->mensaje = 'datos modificados';

  header('Content-Type: application/json');
  echo json_encode($response);
?>
```

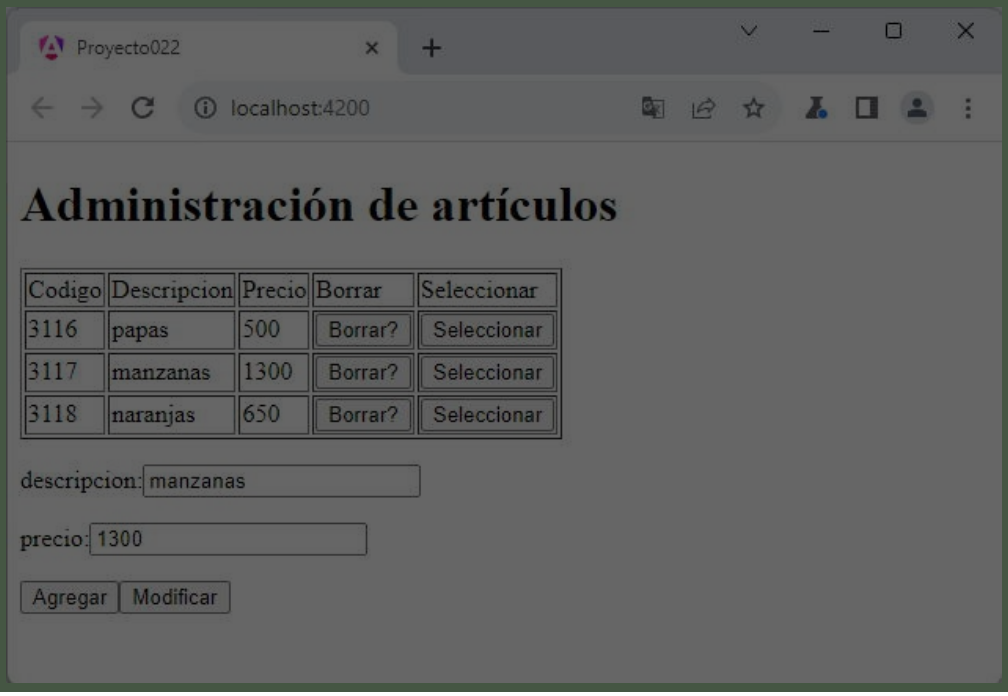
11. El archivo 'seleccionar.php':

```
<?php
  header('Access-Control-Allow-Origin: *');
  header("Access-Control-Allow-Headers: Origin, X-Requested-With, Content-Type, Accept")
;

  require("conexion.php");
  $con=retornarConexion();

  $registros=mysqli_query($con,"select codigo, descripcion, precio from articulos where codi
go=$_GET[codigo]");

  if ($reg=mysqli_fetch_array($registros))
  {
    $vec[]=$reg;
  }

  $cad=json_encode($vec);
  echo $cad;
  header('Content-Type: application/json');
?>
```

12. El archivo 'conexion.php':

```
<?php
function retornarConexion() {
  $con=mysqli_connect("localhost","root","","bd1");
  return $con;
}
?>
```

El resultado de ejecutar esta aplicación en el navegador, teniendo en funcionamiento el servidor con PHP y MySQL es:



Explicación

Ahora veremos como funciona esta aplicación cliente/servidor implementada con Angular en el cliente y PHP en el servidor.

- Recuperación de todos los registros

  Inmediatamente se inicia la aplicación Angular se crea el objeto de la clase 'AppComponent' (nuestra única componente), en esta clase debe llegar al constructor el objeto de la clase 'ArticulosService' y procedemos a llamar al método 'recuperarTodos':

  ```
  constructor(private articulosServicio: ArticulosService) {
    this.recuperarTodos();
  }
  ```

  La clase ArticulosService está creada en el archivo 'articulos.service.ts':

  ```
  export class ArticulosService {
    ...
  }
  ```

  recuperarTodos tiene por objetivo utilizar el 'servicio' que llega al constructor para llamar al método 'recuperarTodos' del servicio propiamente dicho:

  ```
  recuperarTodos() {
    this.articulosServicio.recuperarTodos().subscribe((result:any) => this.articulos = result);
  }
  ```

  Si vemos ahora el método 'recuperarTodos' de la clase 'ArticulosService', es el que tiene la responsabilidad de hacer una petición al servidor:

  ```
  recuperarTodos() {
    return this.http.get(`${this.url}recuperartodos.php`);
  }
  ```

  El método 'recuperarTodos' de la clase 'ArticulosService' retorna un objeto de la clase 'HttpClient'.

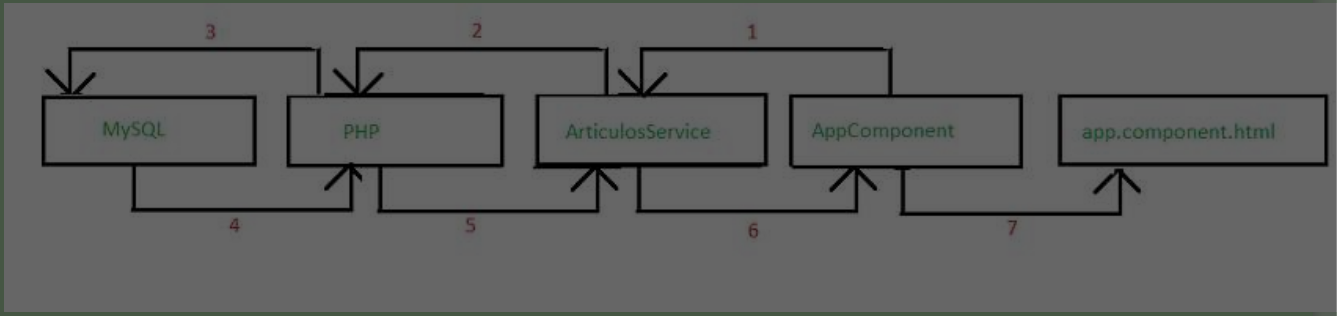
  Ahora debemos entender porque podemos llamar al método 'suscribe':

  ```
  recuperarTodos() {
    this.articulosServicio.recuperarTodos().subscribe((result:any) => this.articulos = result);
  }
  ```

  El método 'suscribe' recibe los resultados y procedemos a asignar a la propiedad 'articulos', con esto, Angular se encarga de actualizar la página con todos los artículos recuperados. El proceso de actualizar la página sucede en el archivo 'app.component.html':

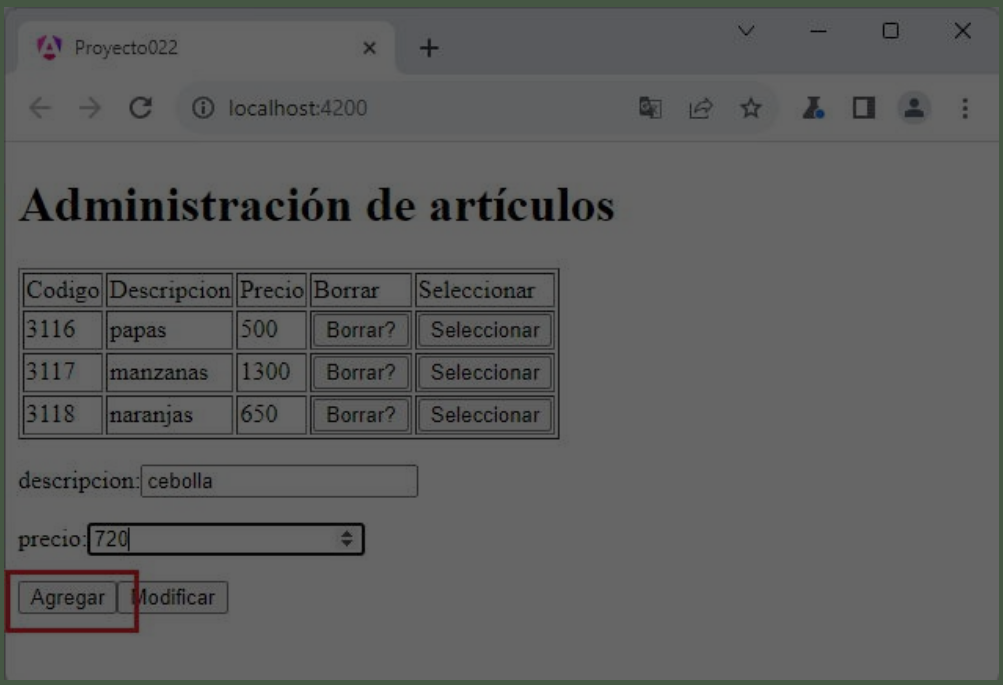
```
<div>
  <h1>Administración de artículos</h1>
    <table border="1">
      <tr>
        <td>Codigo</td><td>Descripcion</td><td>Precio</td><td>Borrar</td><td>Seleccionar</td>
      </tr>
      @for(art of articulos;track art.codigo) {
      <tr>
        <td>{{art.codigo}}</td>
        <td>{{art.descripcion}}</td>
        <td>{{art.precio}}</td>
        <td><button (click)="baja(art.codigo)">Borrar?</button></td>
        <td><button (click)="seleccionar(art.codigo)">Seleccionar</button></td>
      </tr>
    }
    </table>
    <div>
      <p>
        descripcion:<input type="text" [(ngModel)]="art.descripcion" />
      </p>
      <p>
        precio:<input type="number" [(ngModel)]="art.precio" />
      </p>
      <p><button (click)="alta()">Agregar</button>
      <button (click)="modificacion()">Modificar</button></p>
    </div>
</div>
<router-outlet />
```

El flujo de la información lo podemos representar con el siguiente esquema:



- Alta

Veamos ahora los pasos cuando se agrega una fila en la tabla 'articulos'.
Todo comienza cuando el operador presiona el botón de 'Agregar':



La etiqueta button tiene enlazado la llamada al método 'alta':

```
    <p><button (click)="alta()">Agregar</button>
```

El método 'alta' se encuentra codificado en el archivo 'app.component.ts' dentro de la clase 'AppComponent':

```
  alta() {
    this.articulosServicio.alta(this.art).subscribe((datos:any) => {
      if (datos['resultado']=='OK') {
        alert(datos['mensaje']);
        this.recuperarTodos();
      }
    });
  }
```

En este método procedemos a llamar al método 'alta' de la clase 'ArticulosService' y se le pasa como parámetro la propiedad 'art' que almacena la descripción y precio del artículo que el operador acaba de ingresar por teclado.

El método 'alta' de la clase 'ArticulosService' hace la llamada al servidor mediante el objeto 'http' de la clase 'HttpClient'. Se utiliza el método 'post' ya que se enviarán datos al servidor:

```
  alta(articulo:any) {
    return this.http.post(`${this.url}alta.php`, JSON.stringify(articulo));
  }
```

Ahora se ejecuta el programa PHP definido en el archivo 'alta.php' donde procedemos a efectuar el insert en la tabla de MySQL:

```
<?php
  header('Access-Control-Allow-Origin: *');
  header("Access-Control-Allow-Headers: Origin, X-Requested-With, Content-Type, Accept");

  $json = file_get_contents('php://input');

  $params = json_decode($json);

  require("conexion.php");
  $con=retornarConexion();


  mysqli_query($con,"insert into articulos(descripcion,precio) values
                ('$params->descripcion',$params->precio)");


  class Result {}

  $response = new Result();
  $response->resultado = 'OK';
  $response->mensaje = 'datos grabados';

  header('Content-Type: application/json');
  echo json_encode($response);
?>
```

También dentro del programa PHP procedemos a retornar en formato JSON que la operación se efectuó en forma correcta.

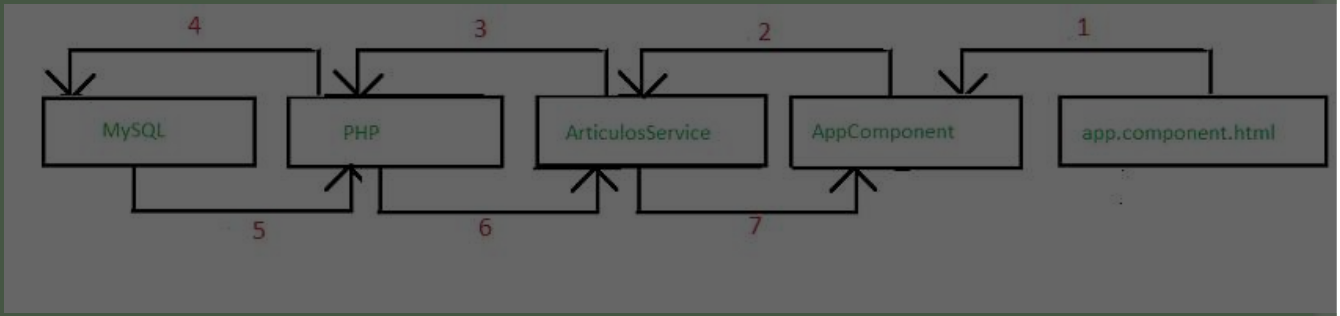
En el método 'alta' de la clase 'AppComponent' que ya vimos, recibe los datos de la respuesta JSon, mostrando un mensaje si la carga se efectuó correctamente:

```
  alta() {
    this.articulosServicio.alta(this.art).subscribe((datos:any) => {
      if (datos['resultado']=='OK') {
        alert(datos['mensaje']);
        this.recuperarTodos();
      }
    });
  }
```

También llamamos al método 'recuperarTodos' con el objetivo que se actualice la pantalla con los datos actuales

de la tabla 'articulos'.

El flujo de la información para efectuar el 'alta' de un articulo en la base de datos es::



La actualización de la página HTML la logramos llamando al método 'recuperarTodos'.

- Baja

El borrado de un artículo se efectúa cuando el operador presiona el botón con la etiqueta 'Borra?':

```
      <td><button (click)="baja(art.codigo)">Borrar?</button></td>
```

Se llama al método 'baja' de la clase 'AppComponent' y se le pasa como parámetro el código de artículo a borrar.

El método baja:

```
baja(codigo:number) {
  this.articulosServicio.baja(codigo).subscribe((datos:any) => {
    if (datos['resultado']=='OK') {
      alert(datos['mensaje']);
      this.recuperarTodos();
    }
  });
}
```

Llmamos al método 'baja' de la clase 'ArticulosService' y le pasamos como parámetro el código de artículo que queremos borrar.

El método 'baja' de la clase 'ArticulosService' procede a llamar al archivo baja.php y le pasa como parámetro el código de artículo que se debe borrar:

```
baja(codigo:number) {
  return this.http.get(`${this.url}baja.php?codigo=${codigo}`);
}
```

Ahora en el servidor se ejecuta la aplicación PHP baja.php:

```
<?php
  header('Access-Control-Allow-Origin: *');
  header("Access-Control-Allow-Headers: Origin, X-Requested-With, Content-Type, Accept");

  require("conexion.php");
  $con=retornarConexion();

  mysqli_query($con,"delete from articulos where codigo=$_GET[codigo]");


  class Result {}

  $response = new Result();
  $response->resultado = 'OK';
  $response->mensaje = 'articulo borrado';

  header('Content-Type: application/json');
  echo json_encode($response);
?>
```

Luego en el método 'baja' de la clase 'AppComponent' podemos mostrar un mensaje si la baja se ejecutó con éxito:

```
  baja(codigo:number) {
    this.articulosServicio.baja(codigo).subscribe((datos:any) => {
      if (datos['resultado']=='OK') {
        alert(datos['mensaje']);
        this.recuperarTodos();
      }
    });
  }
```

Como podemos ver para actualizar la pantalla procedemos a llamar al método 'recuperarTodos'.

- Consulta

  La consulta o selección se dispara cuando el operador presiona el botón que tiene la etiqueta 'Seleccionar' y tiene por objetivo mostrar en los dos controles 'text' la descripción y precio del artículo:

```
        <td><button (click)="seleccionar(art.codigo)">Seleccionar</button></td>
```

  Al presionar el botón se llama el método 'seleccionar' de la clase 'AppComponent':

```
  seleccionar(codigo:number) {
    this.articulosServicio.seleccionar(codigo).subscribe((result:any) => this.art = result[0]);
  }
```

  Ahora llamamos al método 'seleccionar' de la clase 'ArticulosService':

```
  seleccionar(codigo:number) {
    return this.http.get(`${this.url}seleccionar.php?codigo=${codigo}`);
  }
```

  Recuperamos del servidor llamando a la página 'seleccionar.php' los datos del artículo cuyo código pasamos como parámetro :

```
<?php
  header('Access-Control-Allow-Origin: *');
  header("Access-Control-Allow-Headers: Origin, X-Requested-With, Content-Type, Accept");

  require("conexion.php");
  $con=retornarConexion();

  $registros=mysqli_query($con,"select codigo, descripcion, precio from articulos where codigo=$_GET[codigo]");

  if ($reg=mysqli_fetch_array($registros))
  {
    $vec[]=$reg;
  }

  $cad=json_encode($vec);
  echo $cad;
  header('Content-Type: application/json');
?>
```

  La ejecución del programa en PHP procede a recuperar la fila de la tabla que coincide con el código enviado y lo retorna con formato JSON.

  En el método 'seleccionar' de la clase AppComponent al ejecutarse el método subscribe almacena en la propiedad 'art' el resultado devuelto por el servidor (con esta asignación se actualizan los dos controles 'input' del HTML):

```
  seleccionar(codigo:number) {
    this.articulosServicio.seleccionar(codigo).subscribe((result:any) => this.art = result[0]);
  }
```

- Modificación

  El último algoritmo que implementa nuestra aplicación es la modificación de la descripción y precio de un artículo que seleccionemos primeramente.

Cuando presionamos el botón que tiene la etiqueta 'Modificar' se ejecuta el método 'modificación':

```
    <button (click)="modificacion()">Modificar</button></p>
```

El método 'modificación' se implementa en la clase 'AppComponent':

```
  modificacion() {
    this.articulosServicio.modificacion(this.art).subscribe((datos:any) => {
      if (datos['resultado']=='OK') {
        alert(datos['mensaje']);
        this.recuperarTodos();
      }
    });
  }
```

Lo primero que hacemos en este método es llamar al método 'modificación' de la clase 'ArticulosService' y pasar como dato todos los datos del artículo seleccionado y las posibles modificaciones efectuadas.

El método 'modificacion' de la clase 'ArticulosService':

```
  modificacion(articulo:any) {
    return this.http.post(`${this.url}modificacion.php`, JSON.stringify(articulo));
  }
```

Accede al servidor pidiendo el archivo 'modificacion.php' y pasando todos los datos del artículo mediante un 'post'.

El archivo 'modificacion.php' procede a cambiar la descripción y precio del artículo:

```
<?php
  header('Access-Control-Allow-Origin: *');
  header("Access-Control-Allow-Headers: Origin, X-Requested-With, Content-Type, Accept");

  $json = file_get_contents('php://input');

  $params = json_decode($json);

  require("conexion.php");
  $con=retornarConexion();


  mysqli_query($con,"update articulos set descripcion='$params->descripcion',
                                          precio=$params->precio
                                          where codigo=$params->codigo");


  class Result {}

  $response = new Result();
  $response->resultado = 'OK';
  $response->mensaje = 'datos modificados';

  header('Content-Type: application/json');
  echo json_encode($response);
?>
```

En la clase 'AppComponent' podemos comprobar si el resultado fue 'OK' y actualizar nuevamente la página:

```
  modificacion() {
    this.articulosServicio.modificacion(this.art).subscribe((datos:any) => {
      if (datos['resultado']=='OK') {
        alert(datos['mensaje']);
        this.recuperarTodos();
      }
    });
  }
```

Hemos explicado con este problema todos los pasos esenciales para implementar una aplicación en Angular que se comunica con un servidor web con PHP y acceder a una base de datos MySQL.

Puede probar funcionando esta aplicación de Angular desde aquí

Retornar