

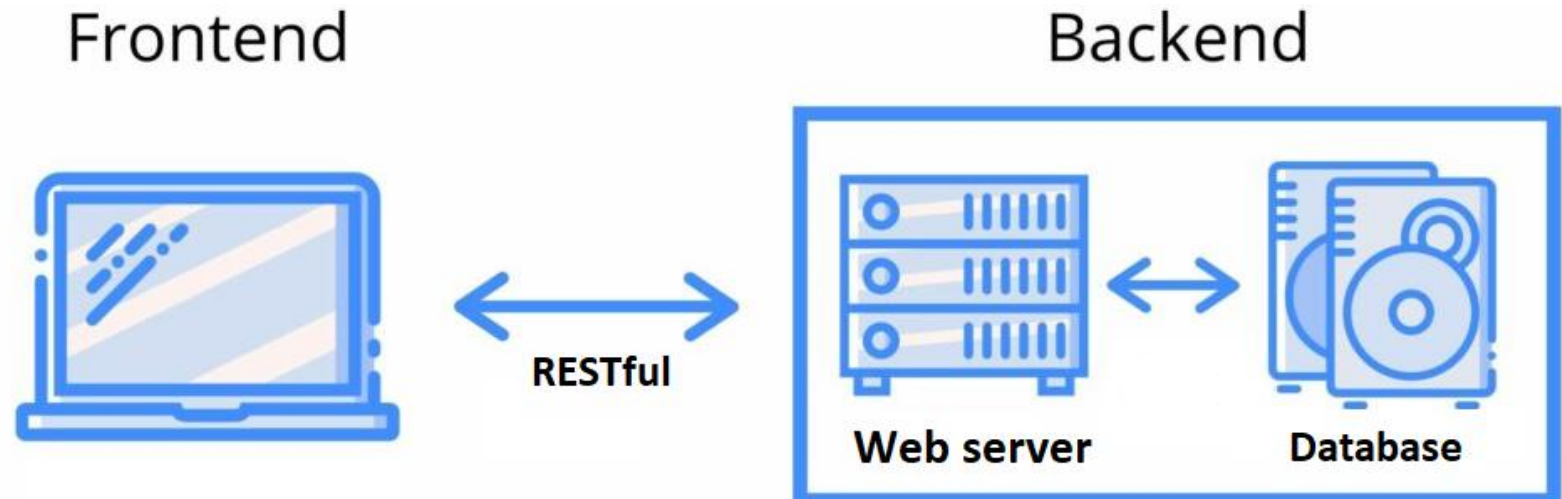
IONIC

consumiendo API

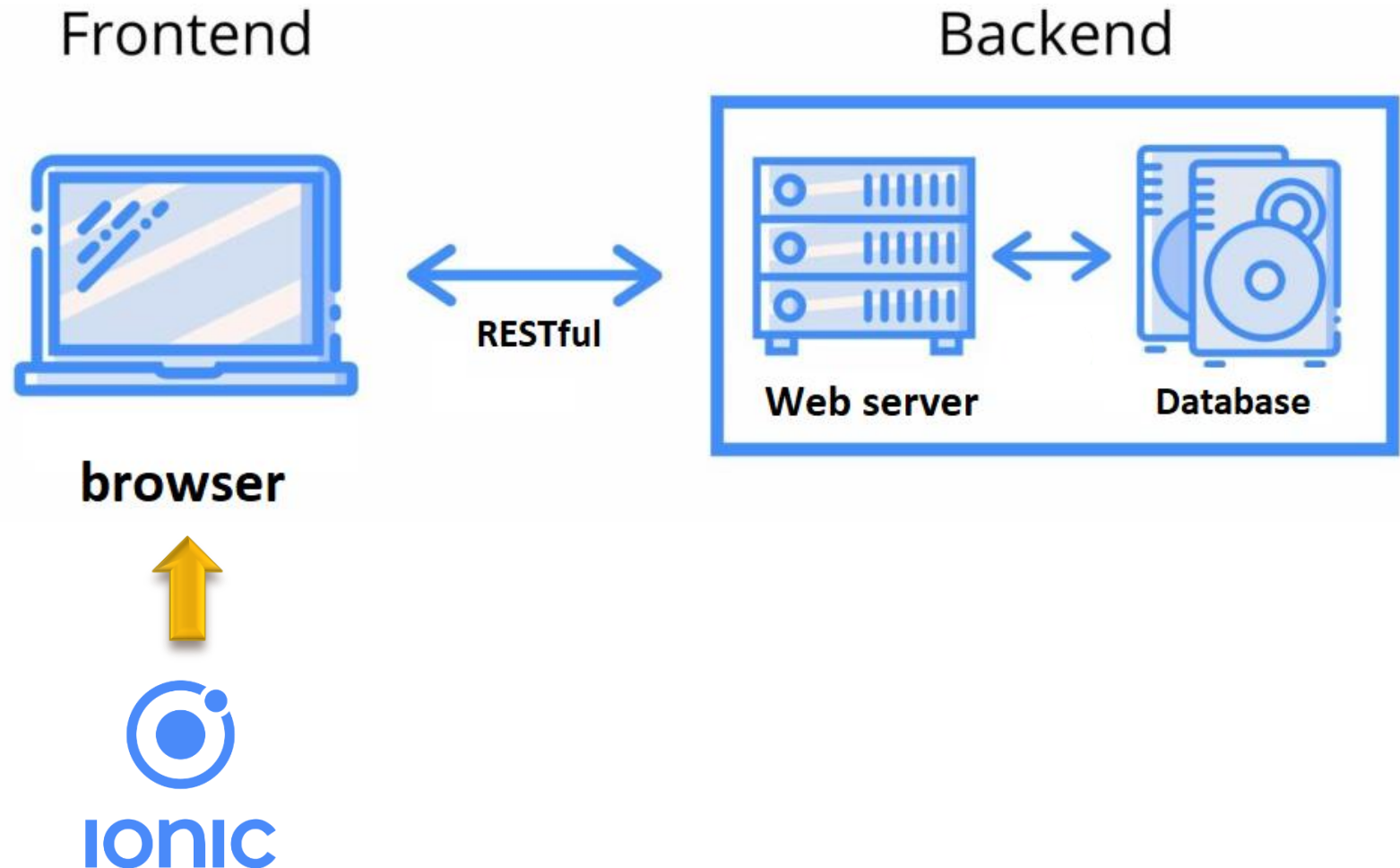
(Usando Angular)

Formularios Reactivos (ReactiveForms)
CRUD accediendo a una API

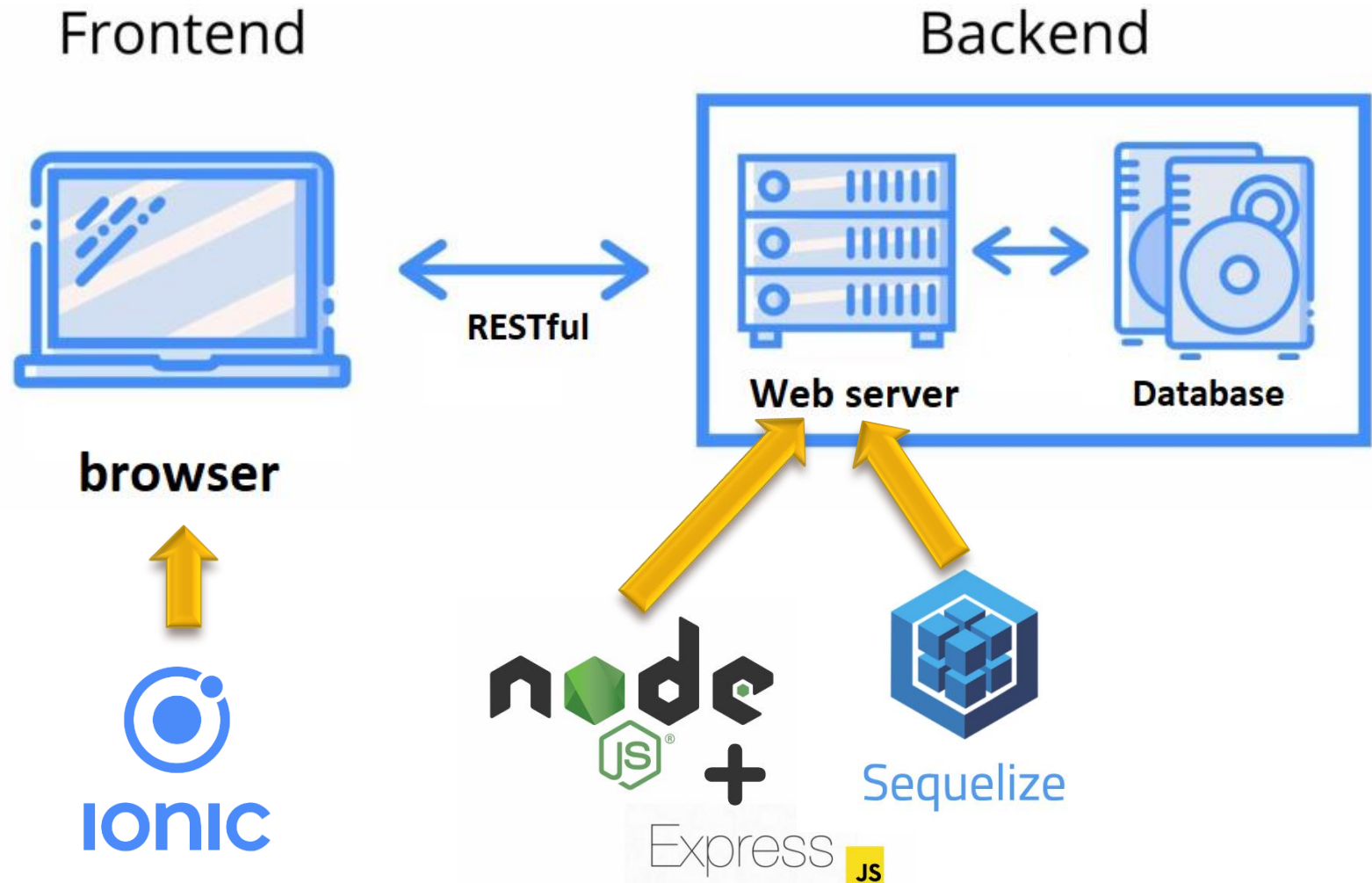
¿Dónde corre Ionic?



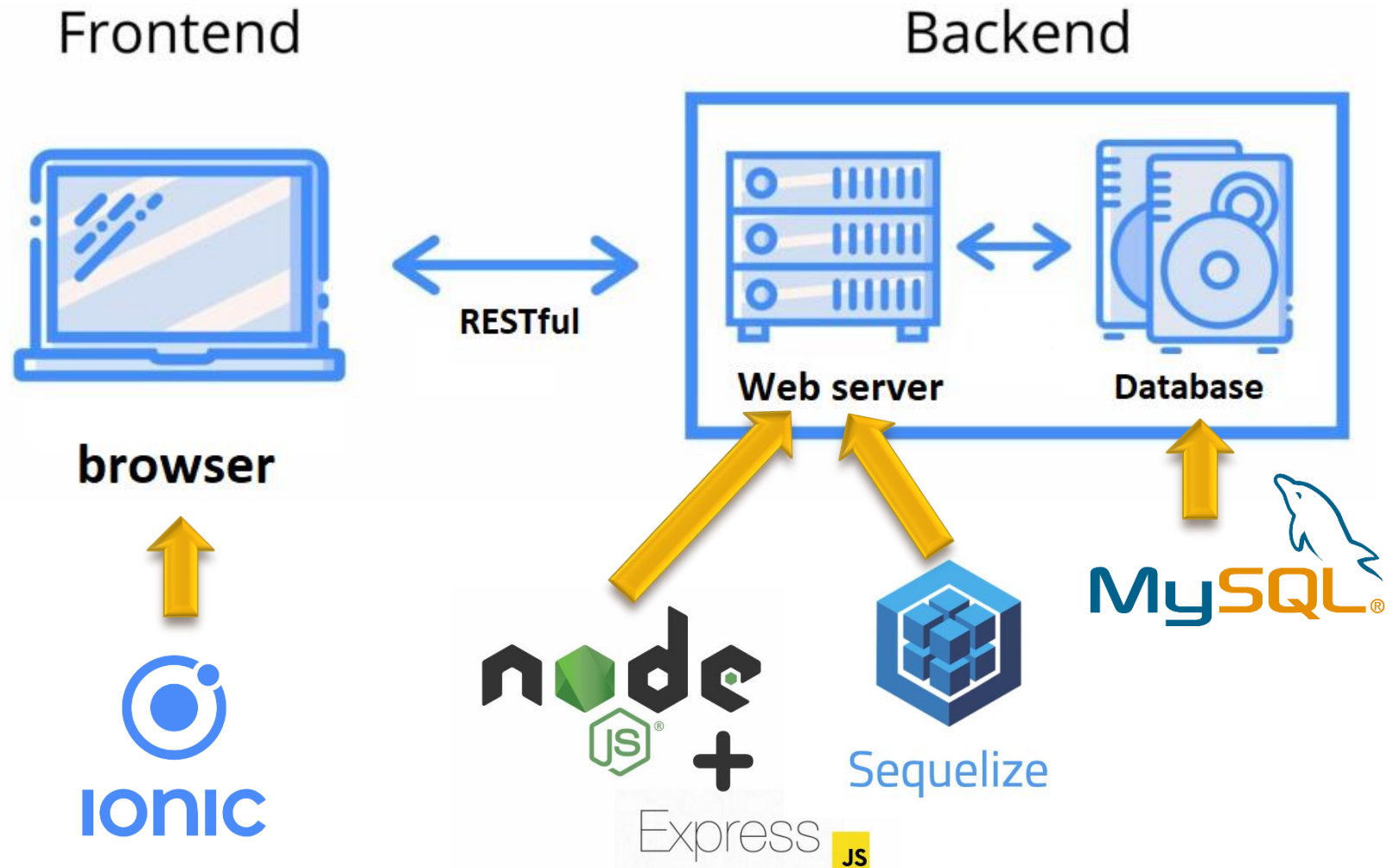
¿Dónde corren Express & Sequelize?



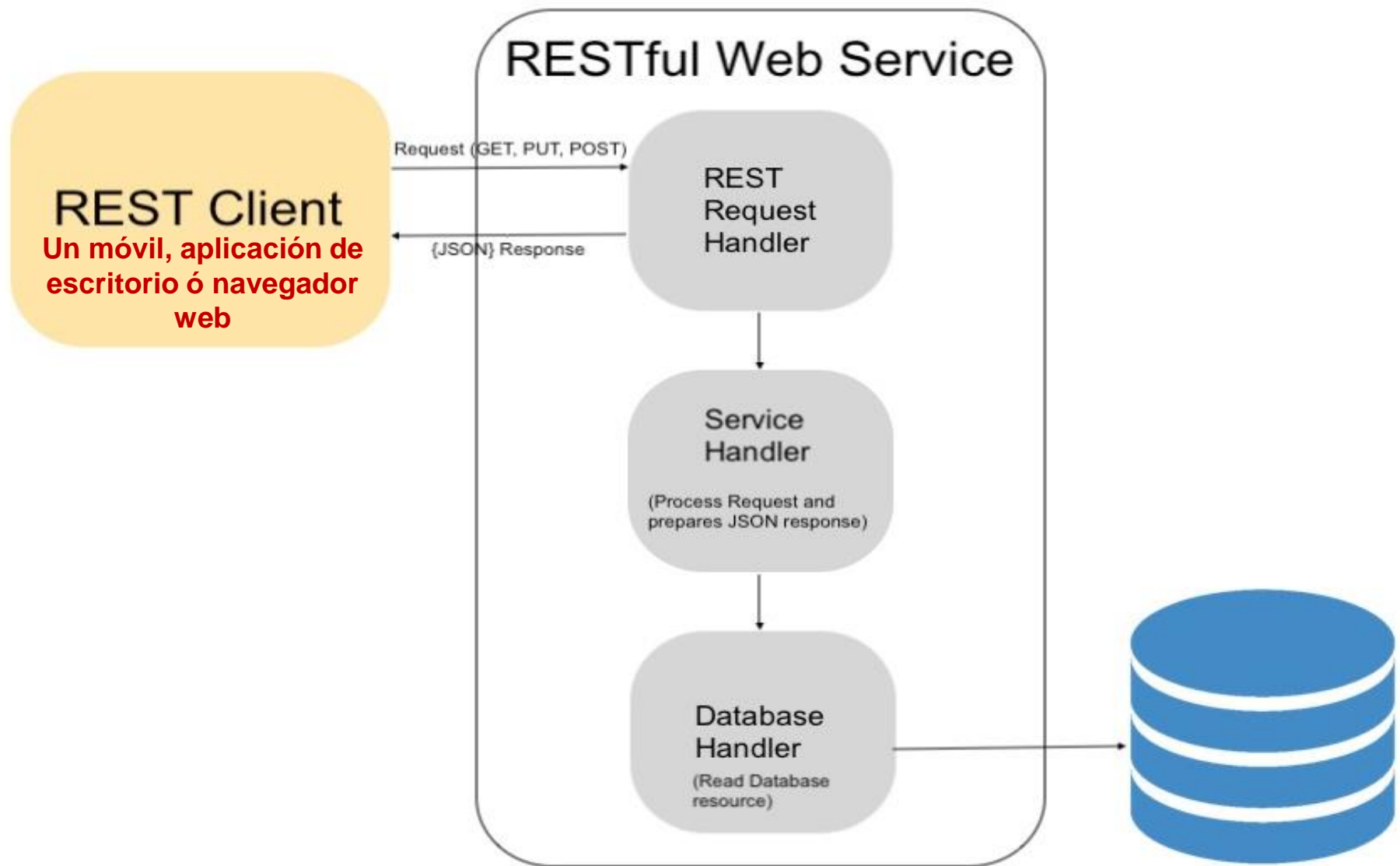
¿Dónde corre mysql?



Visión global



No perdamos nunca la visión global que perseguimos...

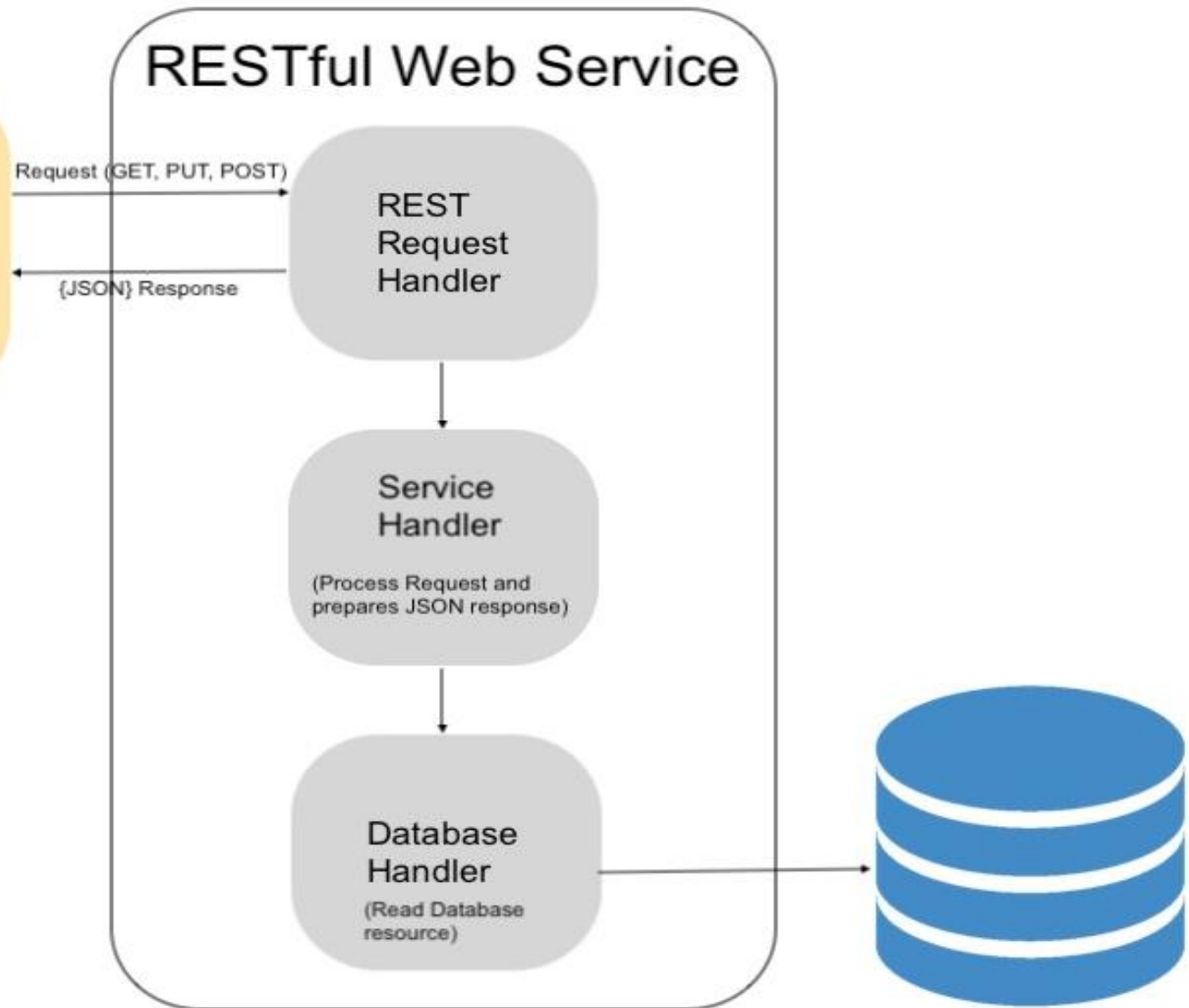


No perdamos nunca la visión global que perseguimos...

IONIC

REST Client

Un móvil, aplicación de escritorio ó navegador web



Al lío... vamos a hacer nuestra App con Ionic...

¿Qué tenemos ahora?

Previamente debes tener instalado:

- **NodeJS**, que es lo que ha hecho que JavaScript pueda ejecutarse fuera del navegador web.
- **npm**, que es el gestor de paquetes de node. (Parecido a apt para Linux)
- Los comandos de abajo permiten ver la versión instalada.

```
$ node --version  
$ npm --version
```


Al lío... vamos a hacer nuestra App con Ionic...

Instalemos ionic...

Instalemos ionic:

- **@ionic/cli**, ionic viene como un paquete más disponible en npm.
- La opción **-g** es para instalarlo globalmente en nuestro equipo.

Con el siguiente comando puedes comprobar la versión de ionic que tienes instalada:

```
$ ionic --version
```

Si aún no tienes ionic instalado ejecuta el siguiente comando:

```
$ npm install -g @ionic/cli
```

Al lío... vamos a hacer nuestra App con Ionic...

Creemos
nuestra primera
app con ionic...

Creando nuestra primera app en ionic:

- **ionic start**, es para crear nuestro proyecto.
- **myApp**, es el nombre que le doy a mi proyecto.
- **blank**, es la plantilla de inicio. Otras opciones son tabs, sidemenu, etc...
- **--capacitor**, es que voy a usar integración con capacitor. La otra opción posible es --cordova
- **--type=angular**, es que voy a trabajar con angular. Otras opciones son react y vue. También puedes usar versiones anteriores: "ionic1" ó "ionic-angular". "ionic start --list" para un listado completo.

```
$ ionic start myApp blank --capacitor --type=angular
```

***Nota:** elige ngModules cuando te salga la opción*


Al lóo... vamos a hacer nuestra App con Ionic...

Creemos nuestra primera app con ionic...

Seguramente te preguntará lo siguiente si quieres crear una cuenta de Ionic...

Para esta práctica no necesitas una cuenta de Ionic...

Para ir al grano en esta práctica responde que No pulsando ENTER.

Join the Ionic Community! 

Connect with millions of developers on the Ionic Forum and get access to live events, news updates, and more.

? Create free Ionic account? (y/N)

Al lío... vamos a hacer nuestra App con Ionic...

Creemos nuestra primera app con ionic...

Si llegas a esta pantalla es que has conseguido crear el esqueleto de un proyecto con Ionic que ya está listo para trabajar...

Your Ionic app is ready! Follow these next steps:

- Go to your new project: `cd .\myApp`
- Run `ionic serve` within the app directory to see your app in the browser
- Run `ionic capacitor add` to add a native iOS or Android project using Capacitor
- Generate your app icon and splash screens using `cordova-res --skip-config --copy`
- Explore the Ionic docs for components, tutorials, and more: <https://ion.link/docs>
- Building an enterprise app? Ionic has Enterprise Support and Features: <https://ion.link/enterprise-edition>

```
tibur@MSI MINGW64 /c/MisCosas/Casa/Ionic
$ █
```

Al lío... vamos a hacer nuestra App con Ionic...

Ejecutamos nuestra primera App con Ionic...

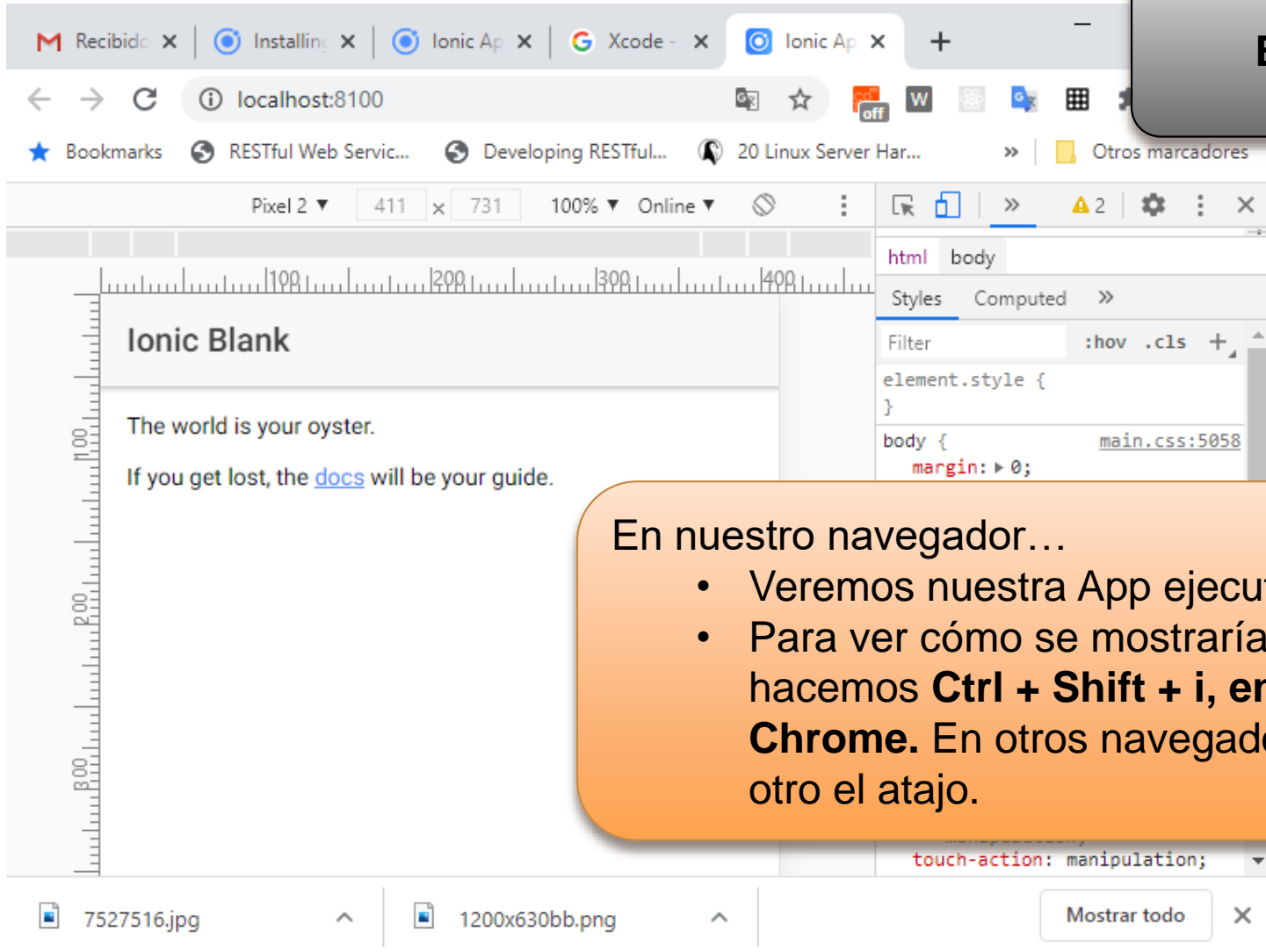
Ejecutamos nuestra primera App con Ionic:

- **cd myApp**, Cambiamos al directorio creado.
- **ionic serve**, Ejecutamos el simulador.

```
$ cd myApp  
$ ionic serve
```

Al lío... vamos a hacer nuestra App con Ionic...

Et voilà...



En nuestro navegador...

- Veremos nuestra App ejecutándose.
- Para ver cómo se mostraría en un móvil hacemos **Ctrl + Shift + i**, en **Google Chrome**. En otros navegadores puede ser otro el atajo.

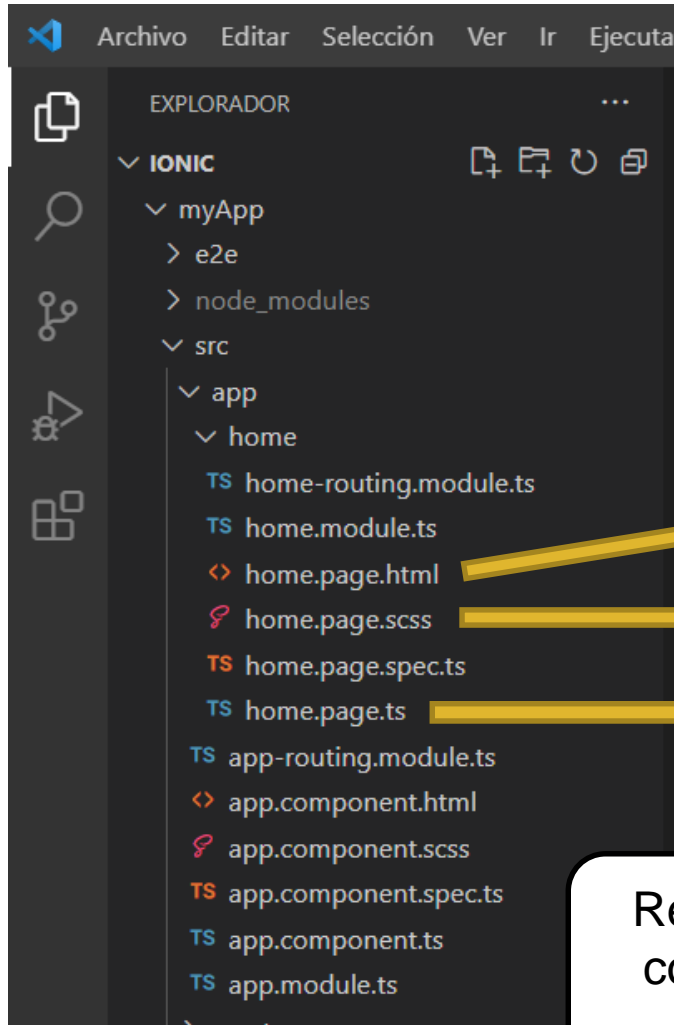
Comencemos a picar
código.

**Todo se reduce a
HTML, CSS y
JavaScript**

Creando una App con Ionic...

Todo es HTML, CSS y JavaScript

Observemos que lo más importante es saber HTML, CSS y JavaScript



HTML

SCSS

TypeScript

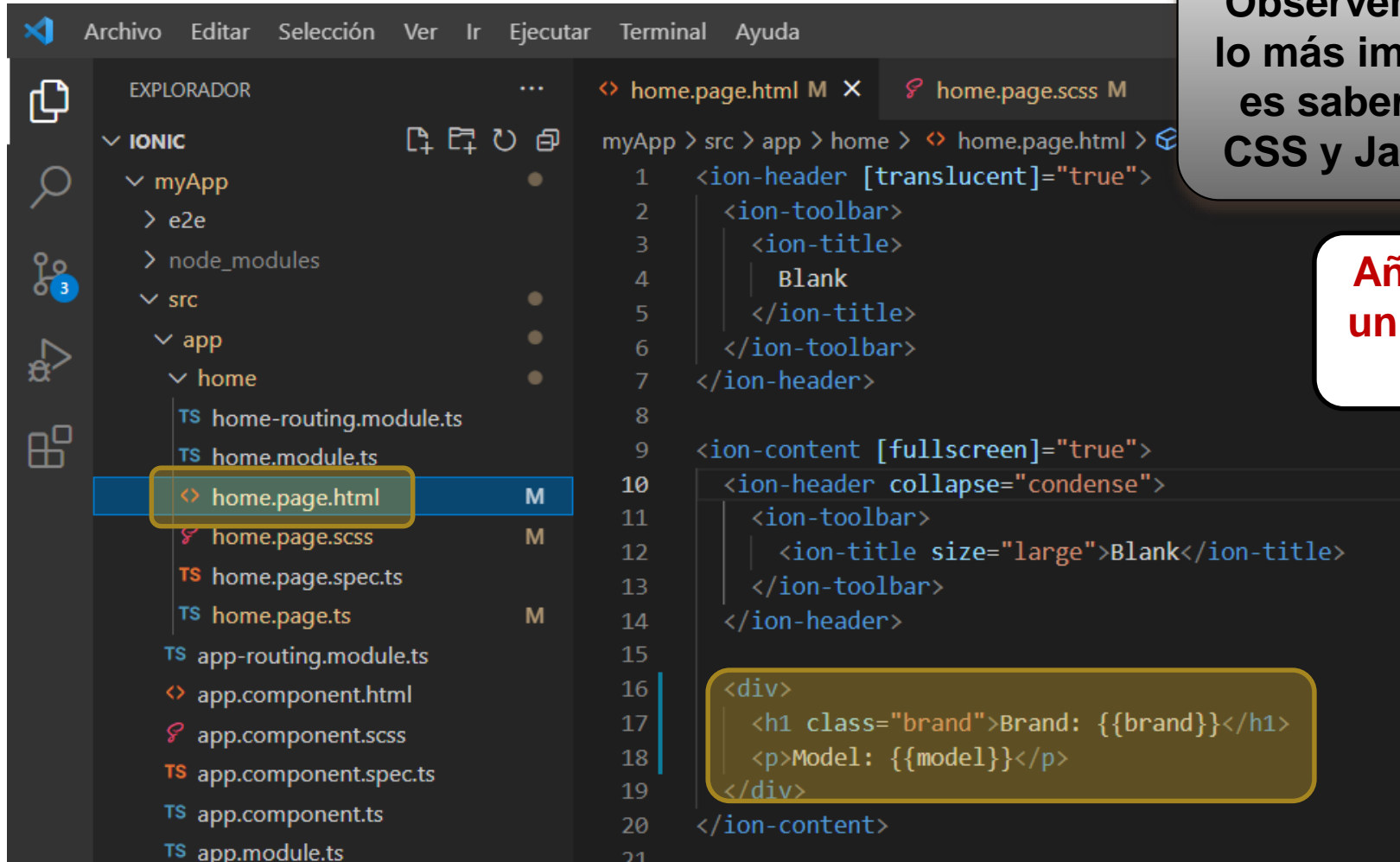
Realmente usamos el framework Angular y trabajamos con SCSS para los estilos y con TypeScript en vez de JavaScript

Creando una App con Ionic...

Todo es HTML, CSS y JavaScript

Observemos que lo más importante es saber HTML, CSS y JavaScript

Añadamos un poco de HTML



The screenshot shows the Visual Studio Code interface with an Ionic project. The Explorer on the left shows the file structure: `myApp` (containing `e2e`, `node_modules`, and `src`), and `src` (containing `app` and `home`). The `home` directory contains `home-routing.module.ts`, `home.module.ts`, `home.page.html` (selected), `home.page.scss`, `home.page.spec.ts`, and `home.page.ts`. The `app` directory contains `app-routing.module.ts`, `app.component.html`, `app.component.scss`, `app.component.spec.ts`, `app.component.ts`, and `app.module.ts`. The main editor shows the content of `home.page.html`, which is an Ionic page template. It features an `<ion-header>` with a translucent toolbar containing the title 'Blank'. The main content area is defined by `<ion-content>` and contains a `<div>` with a `<h1>` and a `<p>` using Angular templating syntax for 'Brand' and 'Model'.

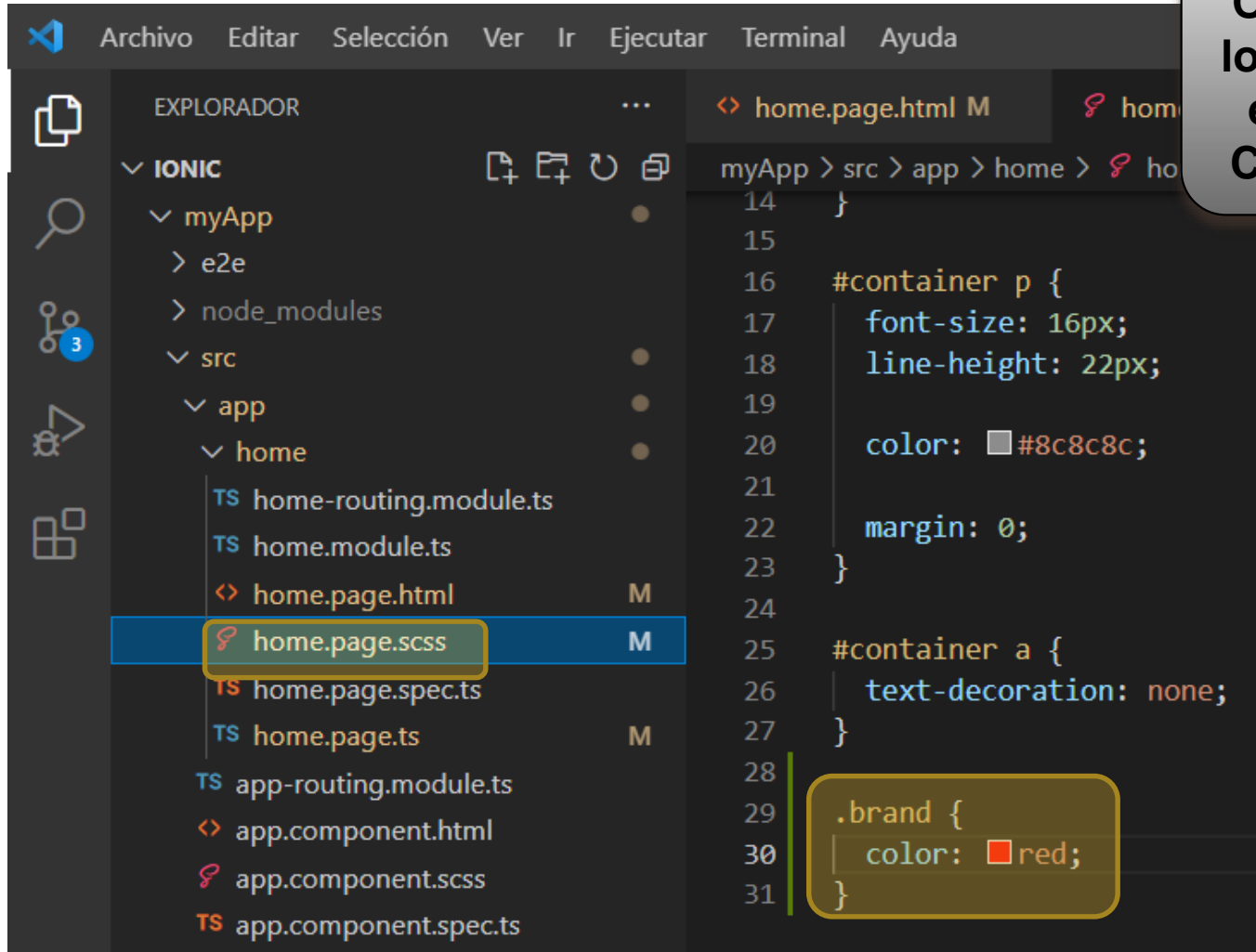
```
1 <ion-header [translucent]="true">
2   <ion-toolbar>
3     <ion-title>
4       Blank
5     </ion-title>
6   </ion-toolbar>
7 </ion-header>
8
9 <ion-content [fullscreen]="true">
10  <ion-header collapse="condense">
11    <ion-toolbar>
12      <ion-title size="large">Blank</ion-title>
13    </ion-toolbar>
14  </ion-header>
15
16  <div>
17    <h1 class="brand">Brand: {{brand}}</h1>
18    <p>Model: {{model}}</p>
19  </div>
20 </ion-content>
21
```

Creando una App con Ionic...

Todo es HTML, CSS y JavaScript

Observemos que lo más importante es saber HTML, CSS y JavaScript

Añadamos un poco de SCSS



The screenshot shows the Visual Studio Code interface with the Explorer sidebar on the left. The project structure is as follows:

- IONIC
 - myApp
 - e2e
 - node_modules
 - src
 - app
 - home
 - home-routing.module.ts
 - home.module.ts
 - home.page.html
 - home.page.scss
 - home.page.spec.ts
 - home.page.ts

The file `home.page.scss` is selected and highlighted. The main editor displays the content of `home.page.html`, which includes SCSS code for styling a container and a brand element.

```
14 }
15
16 #container p {
17   font-size: 16px;
18   line-height: 22px;
19
20   color: #8c8c8c;
21
22   margin: 0;
23 }
24
25 #container a {
26   text-decoration: none;
27 }
28
29 .brand {
30   color: red;
31 }
```

Creando una App con Ionic...

Todo es HTML, CSS y JavaScript

Observemos que lo más importante es saber HTML, CSS y JavaScript

Añadamos un poco de TypeScript

The screenshot shows the Visual Studio Code interface with the Explorer sidebar on the left. The project structure is as follows:

- EXPLORADOR
 - IONIC
 - myApp
 - e2e
 - node_modules
 - src
 - app
 - home
 - home-routing.module.ts
 - home.module.ts
 - home.page.html
 - home.page.scss
 - home.page.spec.ts
 - home.page.ts**
 - app-routing.module.ts
 - app.component.html

The main editor displays the `home.page.ts` file with the following code:

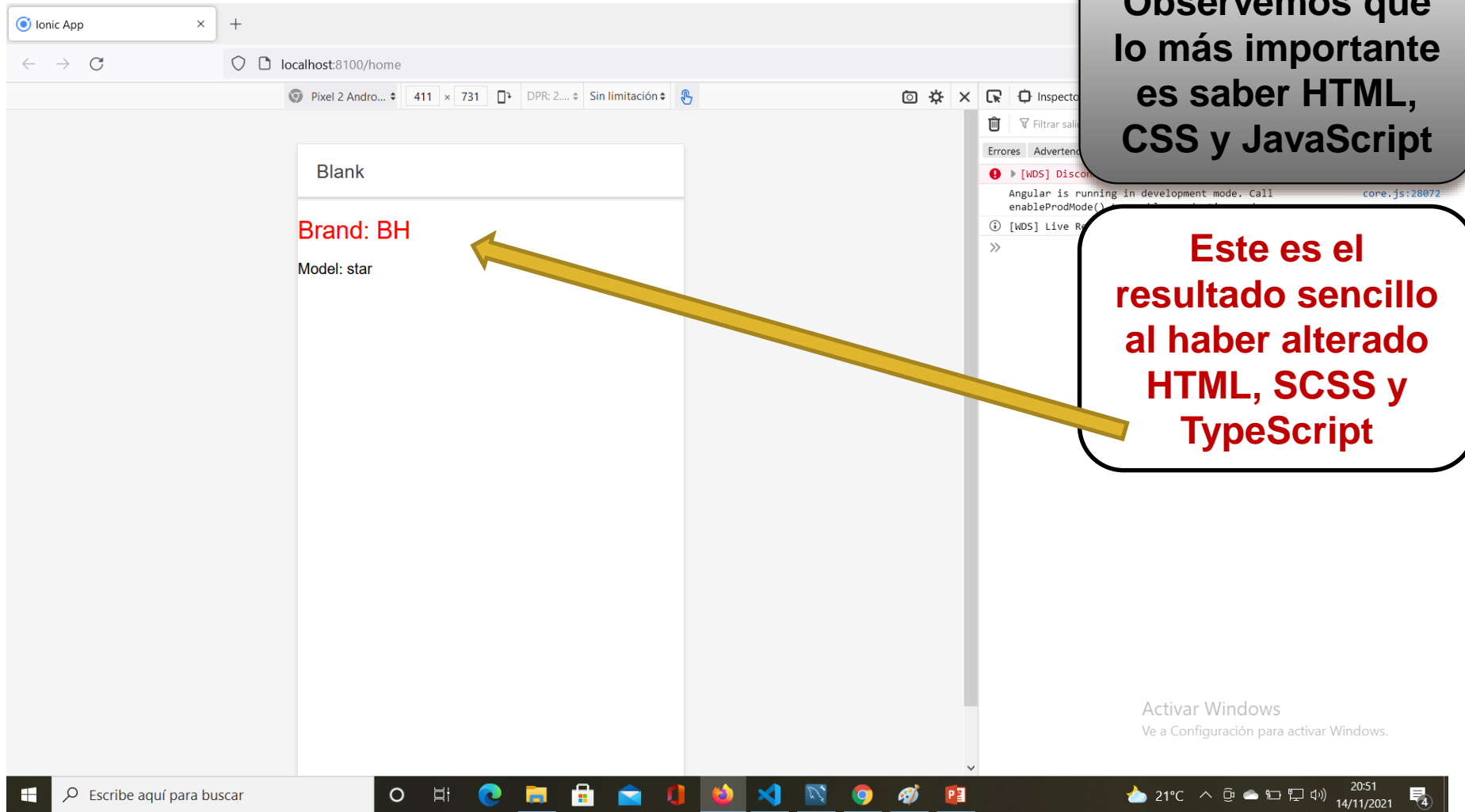
```
1 import { Component } from '@angular/core';
2
3 @Component({
4   selector: 'app-home',
5   templateUrl: 'home.page.html',
6   styleUrls: ['home.page.scss'],
7 })
8 export class HomePage {
9
10   brand: string = "BH";
11   model: string = "star";
12
13   constructor() {}
14
15 }
16
```

Creando una App con Ionic...

Todo es HTML, CSS y JavaScript

Observemos que lo más importante es saber HTML, CSS y JavaScript

Este es el resultado sencillo al haber alterado HTML, SCSS y TypeScript



Creemos ahora una nueva página llamada my-bicycles que muestre un listado a partir de un Array de objetos JSON

Creando una App con Ionic...

Listado a partir de un Array de objetos JSON

Creemos una página llamada my-bicycles

```
tibur@DESKTOP-02362TM MINGW64 /c/MisCosas/Casa/Ionic/myApp
$ ionic generate page my-bicycles
> ng.cmd generate page my-bicycles --project=app
CREATE src/app/my-bicycles/my-bicycles-routing.module.ts (364 bytes)
CREATE src/app/my-bicycles/my-bicycles.module.ts (502 bytes)
CREATE src/app/my-bicycles/my-bicycles.page.html (130 bytes)
CREATE src/app/my-bicycles/my-bicycles.page.spec.ts (690 bytes)
CREATE src/app/my-bicycles/my-bicycles.page.ts (275 bytes)
CREATE src/app/my-bicycles/my-bicycles.page.scss (0 bytes)
UPDATE src/app/app-routing.module.ts (630 bytes)
[OK] Generated page!

tibur@DESKTOP-02362TM MINGW64 /c/MisCosas/Casa/Ionic/myApp (master)
$
```

Observa que se usa la notación “hyphen case”, también conocida como “kebab case”

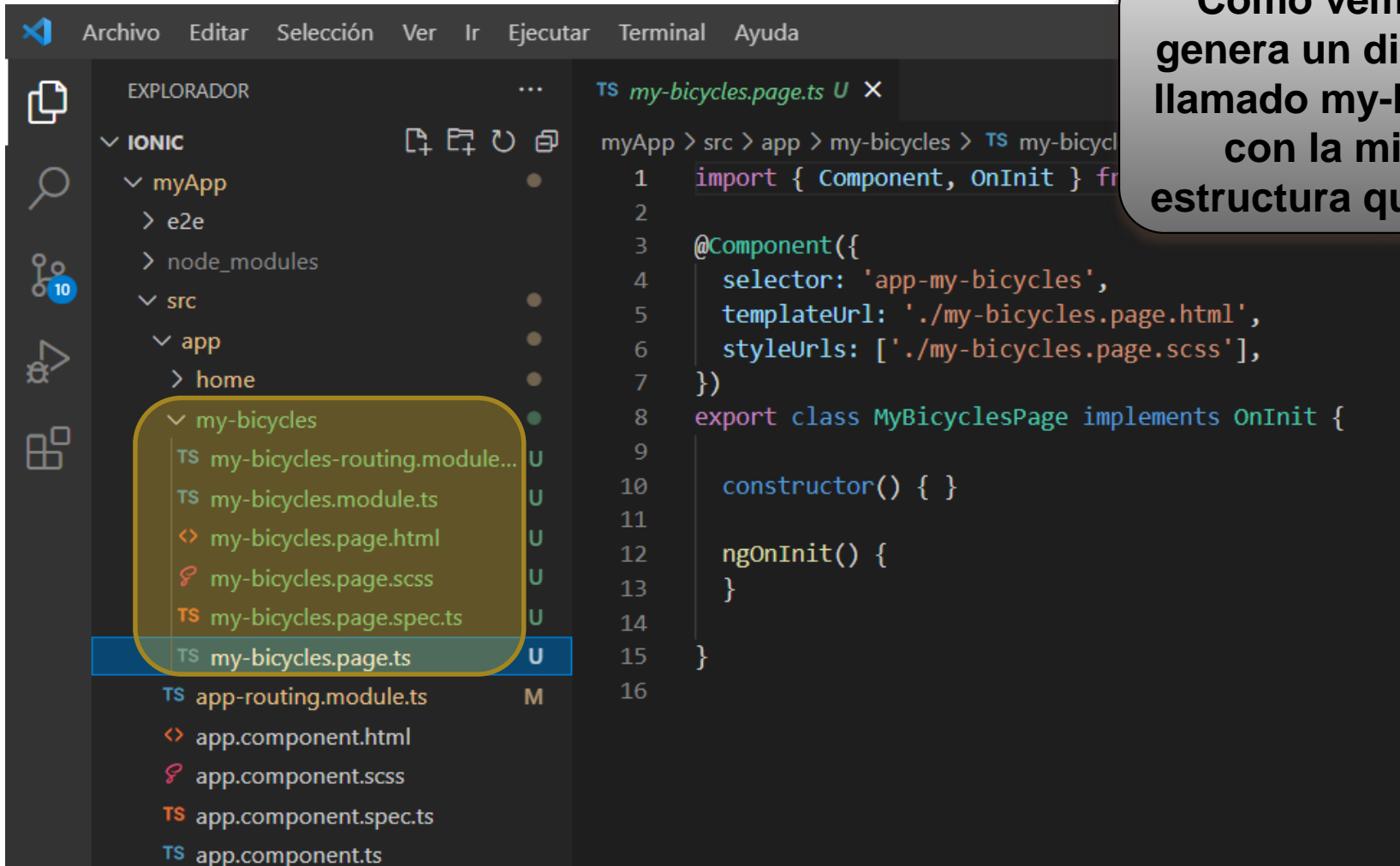
Repaso de notaciones:

<https://www.freecodecamp.org/news/snake-case-vs-camel-case-vs-pascal-case-vs-kebab-case-whats-the-difference/>

Creando una App con Ionic...

Listado a partir de un Array de objetos JSON

Como vemos se genera un directorio llamado my-bicycles con la misma estructura que home



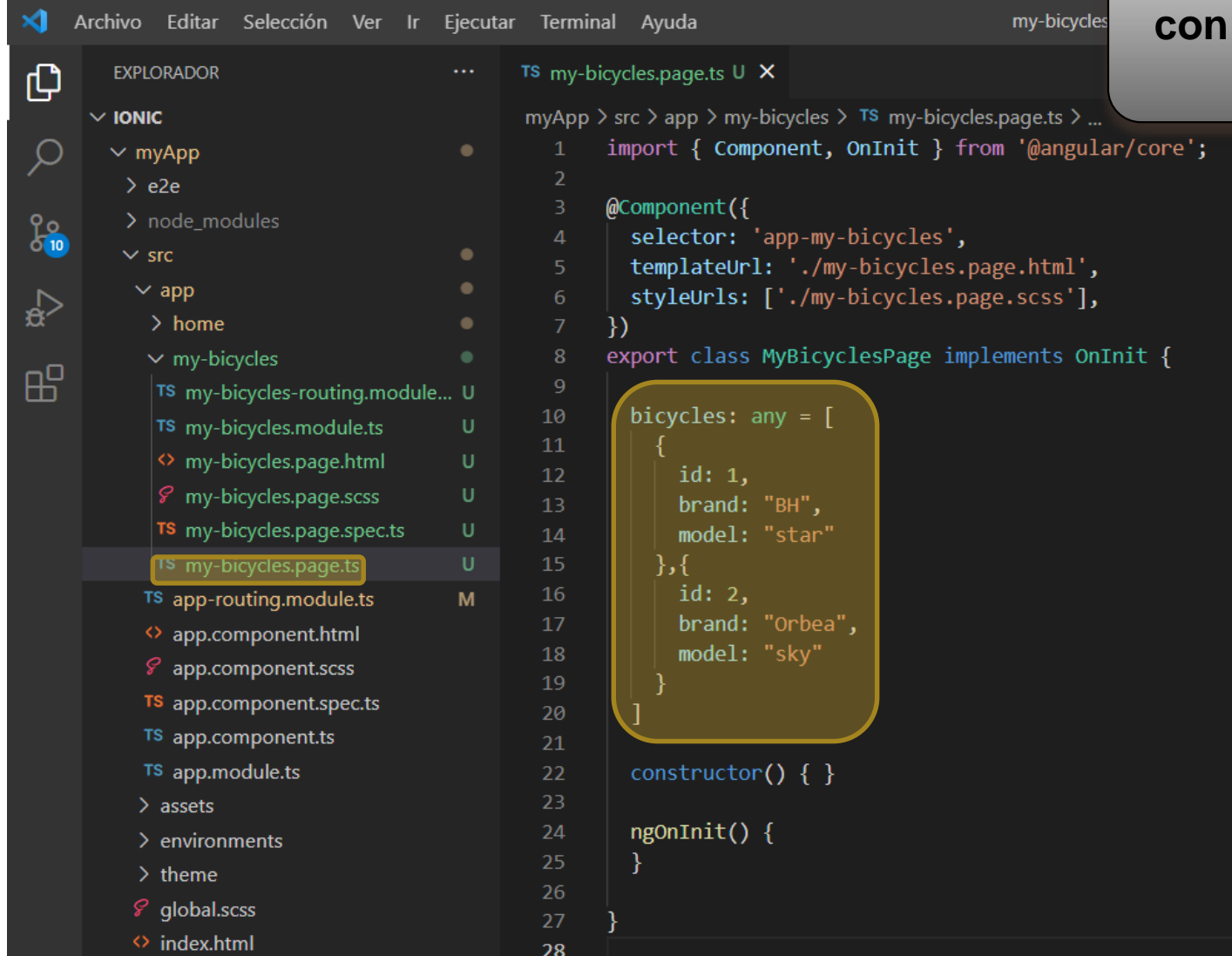
The screenshot shows the Visual Studio Code interface. On the left, the 'EXPLORADOR' (Explorer) sidebar displays the project structure. The 'my-bicycles' directory is highlighted with a yellow box, showing files: `my-bicycles-routing.module.ts`, `my-bicycles.module.ts`, `my-bicycles.page.html`, `my-bicycles.page.scss`, `my-bicycles.page.spec.ts`, and `my-bicycles.page.ts`. The `my-bicycles.page.ts` file is selected and highlighted in blue. The main editor area shows the content of `my-bicycles.page.ts`, which includes imports for `Component` and `OnInit`, a `@Component` decorator with configuration for the page, and a `MyBicyclesPage` class implementing `OnInit`.

```
1 import { Component, OnInit } from '@angular/core';
2
3 @Component({
4   selector: 'app-my-bicycles',
5   templateUrl: './my-bicycles.page.html',
6   styleUrls: ['./my-bicycles.page.scss'],
7 })
8 export class MyBicyclesPage implements OnInit {
9
10   constructor() { }
11
12   ngOnInit() {
13   }
14
15 }
```

Creando una App con Ionic...

Listado a partir de un Array de objetos JSON

Añadamos un Array con información de bicicletas



The screenshot shows the Visual Studio Code editor with an Ionic project. The Explorer on the left shows the project structure, with the file `my-bicycles.page.ts` selected. The main editor displays the code for `my-bicycles.page.ts`, which includes an Angular component definition and a TypeScript array of bicycle objects. A yellow box highlights the `bicycles` array definition.

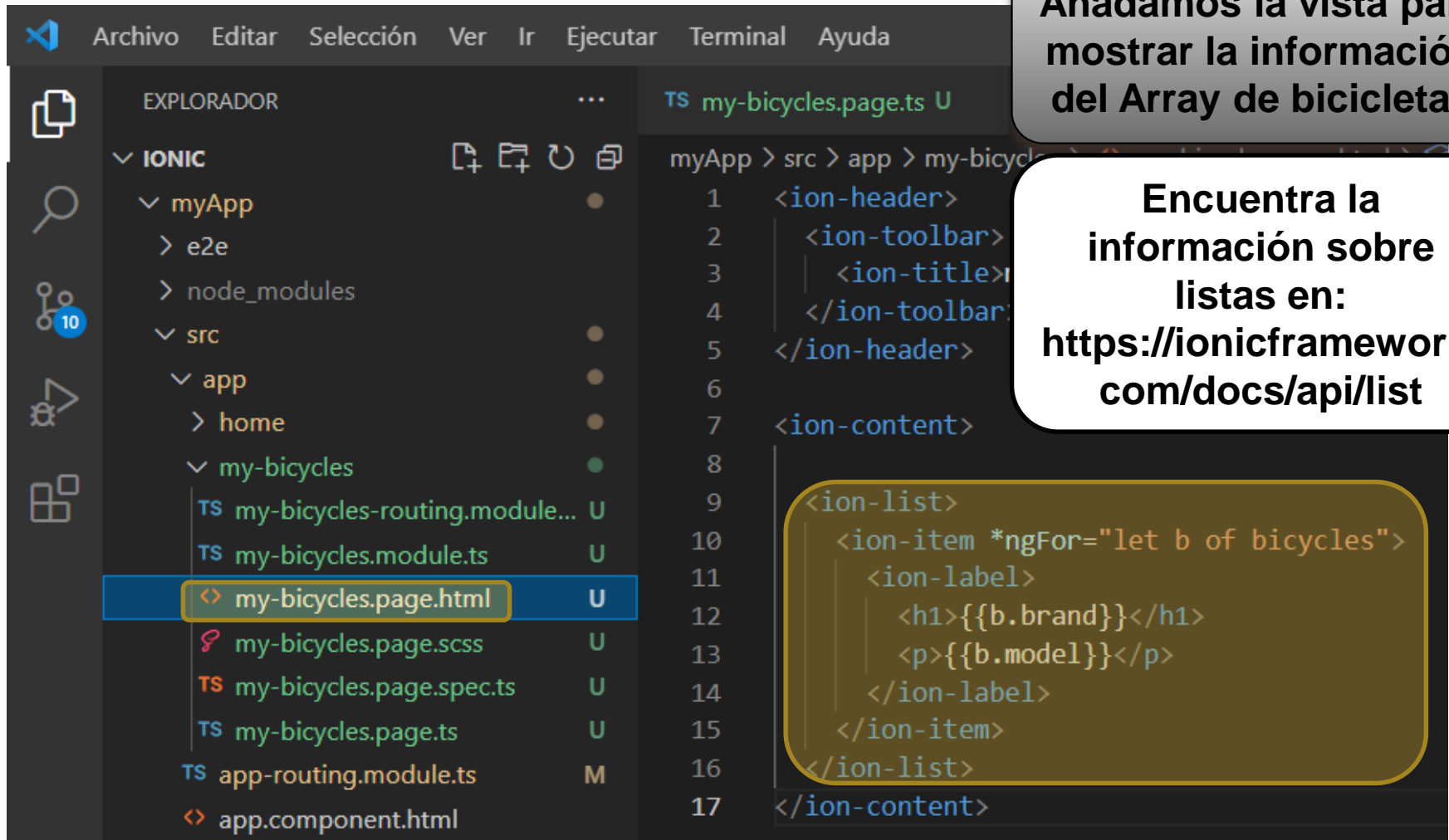
```
myApp > src > app > my-bicycles > TS my-bicycles.page.ts > ...
1  import { Component, OnInit } from '@angular/core';
2
3  @Component({
4    selector: 'app-my-bicycles',
5    templateUrl: './my-bicycles.page.html',
6    styleUrls: ['./my-bicycles.page.scss'],
7  })
8  export class MyBicyclesPage implements OnInit {
9
10     bicycles: any = [
11       {
12         id: 1,
13         brand: "BH",
14         model: "star"
15       }, {
16         id: 2,
17         brand: "Orbea",
18         model: "sky"
19       }
20     ]
21
22     constructor() { }
23
24     ngOnInit() {
25     }
26
27   }
28
```


Creando una App con Ionic...

Listado a partir de un Array de objetos JSON

Añadamos la vista para mostrar la información del Array de bicicletas

Encuentra la información sobre listas en:
<https://ionicframework.com/docs/api/list>

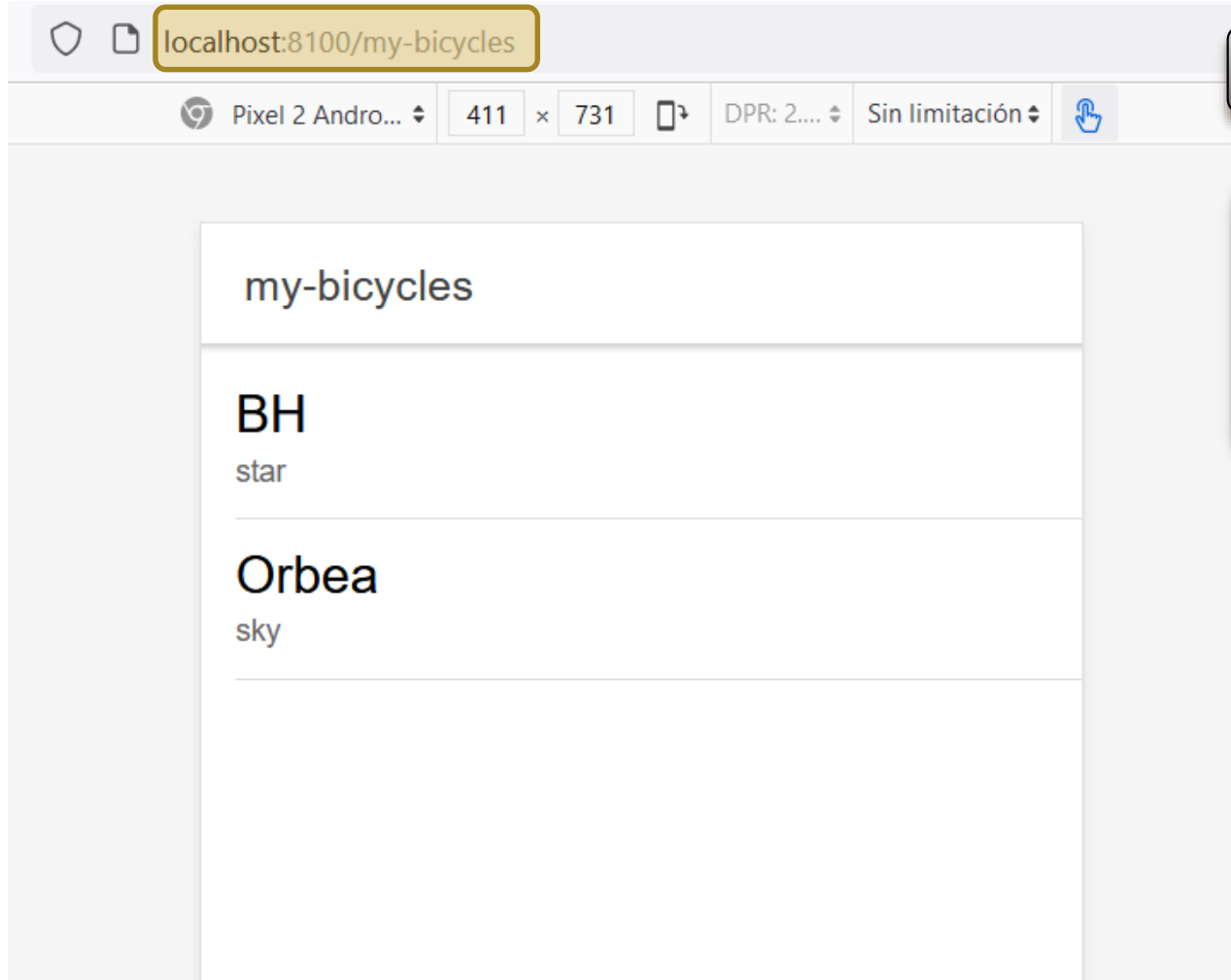


The screenshot shows the Visual Studio Code interface. On the left, the Explorer sidebar displays the project structure under the 'myApp' folder, with 'my-bicycles' selected. The file 'my-bicycles.page.html' is highlighted. The main editor shows the content of 'my-bicycles.page.html' with the following code:

```
1 <ion-header>
2   <ion-toolbar>
3     <ion-title>
4   </ion-toolbar>
5 </ion-header>
6
7 <ion-content>
8
9   <ion-list>
10     <ion-item *ngFor="let b of bicycles">
11       <ion-label>
12         <h1>{{b.brand}}</h1>
13         <p>{{b.model}}</p>
14       </ion-label>
15     </ion-item>
16   </ion-list>
17 </ion-content>
```

Creando una App con Ionic...

Listado a partir de un Array de objetos JSON



Y este es el resultado

Para verlo cambia la URL de acceso a la página:
<http://localhost:8100/my-bicycles>

Encuentra la información sobre listas en:
<https://ionicframework.com/docs/api/list>

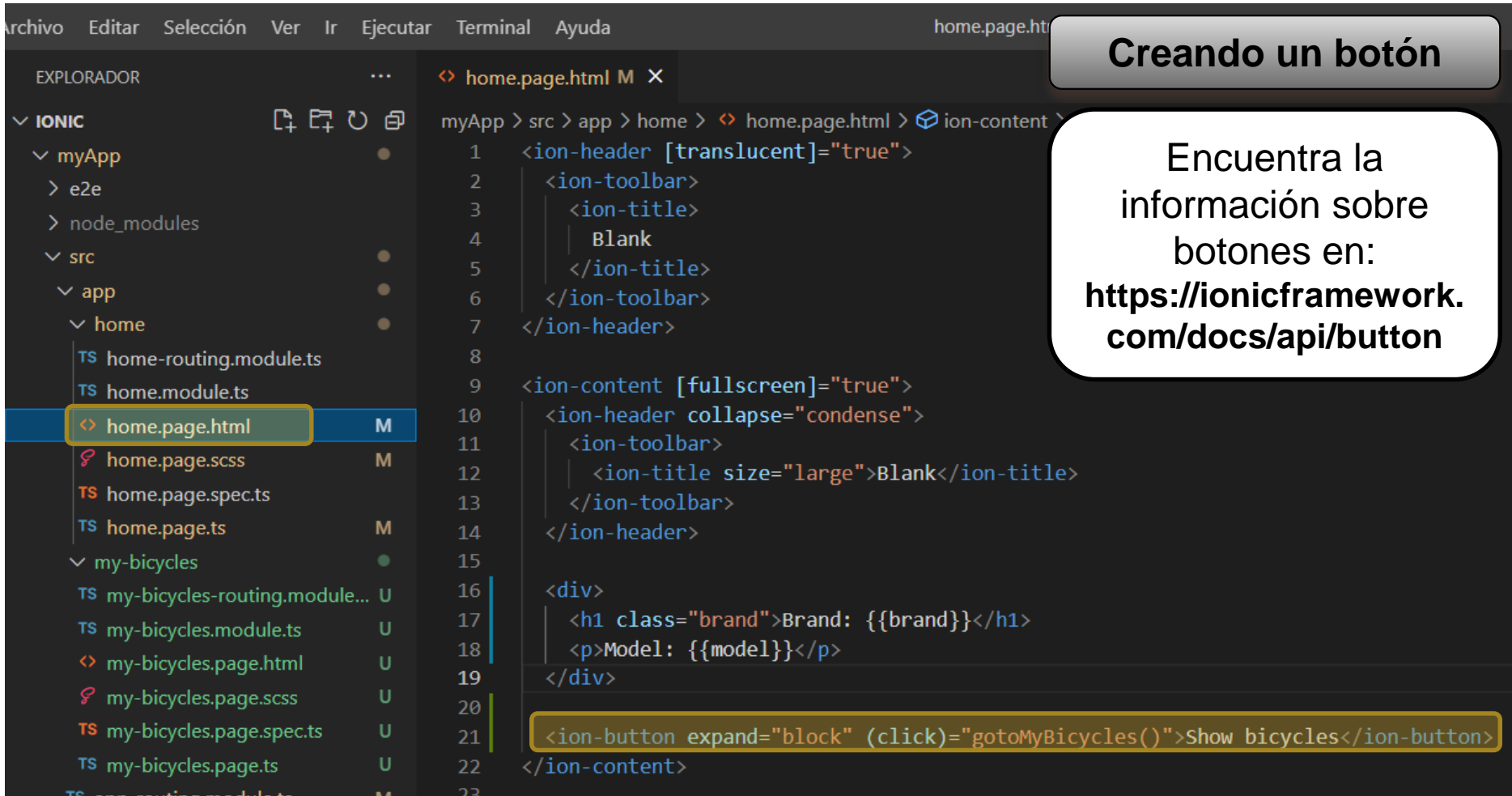
En vez de cambiar la URL a mano vamos a pasar de una página a otra a través de un botón.

Creando una App con Ionic...

Pasar de una página a otra

Creando un botón

Encuentra la información sobre botones en:
<https://ionicframework.com/docs/api/button>



```
myApp > src > app > home > <> home.page.html > ion-content >
1 <ion-header [translucent]="true">
2   <ion-toolbar>
3     <ion-title>
4       Blank
5     </ion-title>
6   </ion-toolbar>
7 </ion-header>
8
9 <ion-content [fullscreen]="true">
10  <ion-header collapse="condense">
11    <ion-toolbar>
12      <ion-title size="large">Blank</ion-title>
13    </ion-toolbar>
14  </ion-header>
15
16  <div>
17    <h1 class="brand">Brand: {{brand}}</h1>
18    <p>Model: {{model}}</p>
19  </div>
20
21  <ion-button expand="block" (click)="gotoMyBicycles()">Show bicycles</ion-button>
22 </ion-content>
23
```

Creando una App con Ionic...

Pasar de una página a otra

The screenshot shows an IDE with the following structure:

- EXPLORADOR
 - IONIC
 - myApp
 - e2e
 - node_modules
 - src
 - app
 - home
 - home-routing.module.ts
 - home.module.ts
 - home.page.html
 - home.page.scss
 - home.page.spec.ts
 - home.page.ts**
 - my-bicycles
 - my-bicycles-routing.module...
 - my-bicycles.module.ts
 - my-bicycles.page.html
 - my-bicycles.page.scss
 - my-bicycles.page.spec.ts

The active file is `home.page.ts`, which contains the following code:

```
1 import { Component } from '@angular/core';
2 import { Router } from '@angular/router';
3
4 @Component({
5   selector: 'app-home',
6   templateUrl: 'home.page.html',
7   styleUrls: ['home.page.scss'],
8 })
9 export class HomePage {
10
11   brand: string = "BH";
12   model: string = "star";
13
14   constructor(private router: Router) { }
15
16   gotoMyBicycles() {
17     this.router.navigateByUrl("/my-bicycles");
18   }
19
20 }
21
```

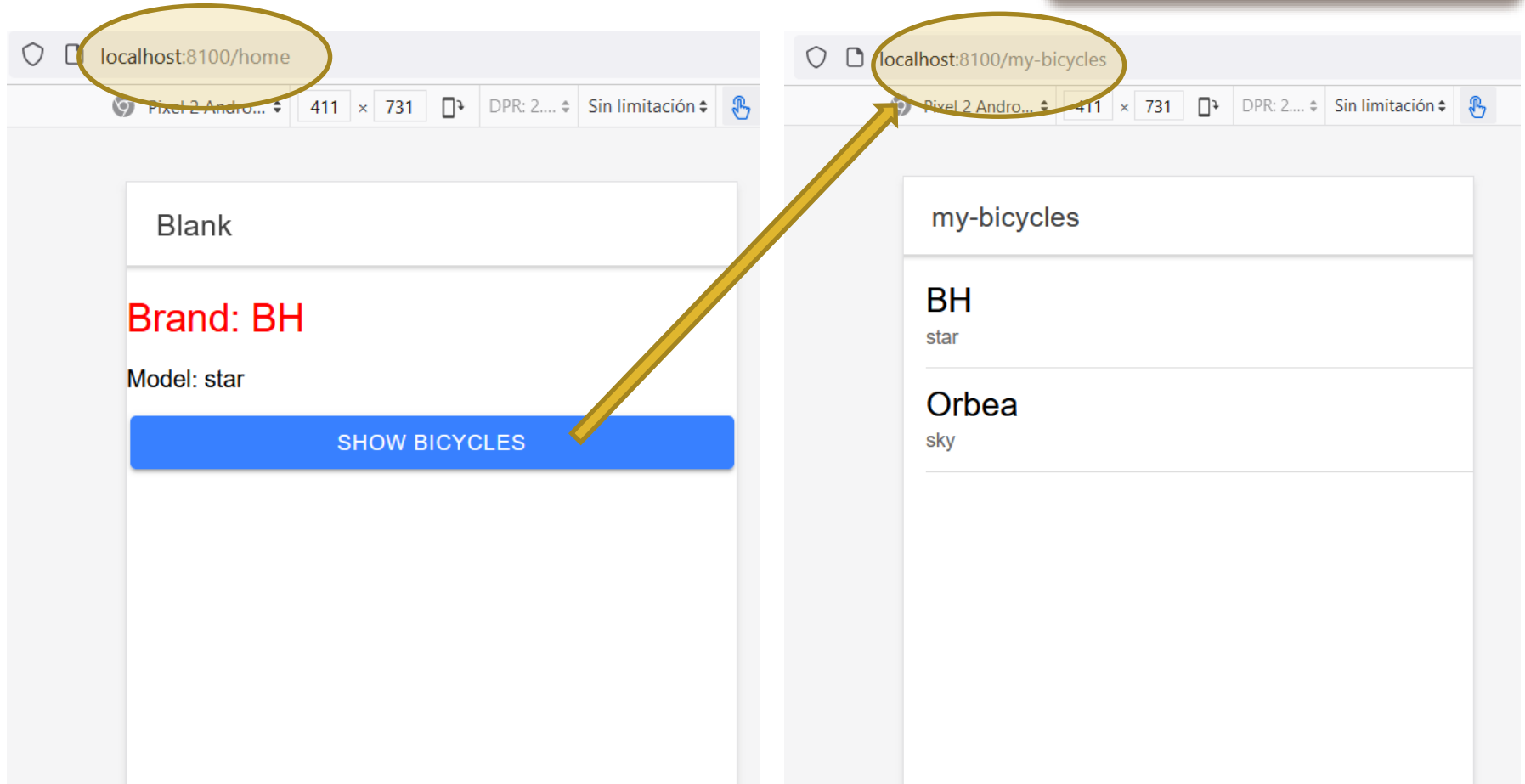
Creando un enlace

Encuentra la información sobre navegación en:
<https://ionicframework.com/docs/angular/navigation>

Creando una App con Ionic...

Pasar de una página a otra

Probando el botón



Entendiendo el enrutamiento en Ionic

Creando una App con Ionic...

Entendiendo el enrutamiento en Ionic

e.page.html M TS app-routing.module.ts M X

> src > app > TS app-routing.module.ts > ...

```
import { NgModule } from '@angular/core';
import { PreloadAllModules, RouterModule, Routes } from '@angular/router';

const routes: Routes = [
  {
    path: 'home',
    loadChildren: () => import('./home/home.module').then( m => m.HomePageModule)
  },
  {
    path: '',
    redirectTo: 'home',
    pathMatch: 'full'
  },
  {
    path: 'my-bicycles',
    loadChildren: () => import('./my-bicycles/my-bicycles.module').then( m => m.MyBicyclesPageModule)
  },
];

@NgModule({
  imports: [
    RouterModule.forRoot(routes, { preloadingStrategy: PreloadAllModules })
  ],
  exports: [RouterModule]
})
export class AppRoutingModule { }
```

Observemos el fichero
app-routing.modules.ts

Al crear una página la
ruta correspondiente
se crea en este fichero.
De momento tenemos
2 páginas: “home” y
“my-bicycles”

Creando un servicio para
consumir una API RESTful

Sustituir nuestro Array de
objetos JSON “hardcoded”
por datos obtenidos de la API

Creando una App con Ionic...

Creando un servicio para consumir una API

The screenshot shows an IDE with a file explorer on the left, a code editor in the center, and a terminal at the bottom.

EXPLORADOR (File Explorer):

- IONIC
 - myApp
 - e2e
 - node_modules
 - src
 - app
 - home
 - home-routing.module.ts
 - home.module.ts
 - home.page.html
 - home.page.scss
 - home.page.spec.ts
 - home.page.ts
 - my-bicycles
 - services
 - bicycle.service.spec.ts
 - bicycle.service.ts**
 - app-routing.module.ts
 - app.component.html
 - app.component.scss
 - app.component.spec.ts
 - app.component.ts
 - app.module.ts

TS bicycle.service.ts

```
1 import { Injectable } from '@angular/core';
2
3 @Injectable({
4   providedIn: 'root'
5 })
6 export class BicycleService {
7
8   constructor() { }
9
10 }
```

TERMINAL

```
tibur@DESKTOP-02362TM MINGW64 /c/MisCosas/Casa/Ionic
$ cd myApp/

tibur@DESKTOP-02362TM MINGW64 /c/MisCosas/Casa/Ionic/myApp (master)
$ ionic generate service services/bicycle
> ng.cmd generate service services/bicycle --project=app
CREATE src/app/services/bicycle.service.spec.ts (362 bytes)
CREATE src/app/services/bicycle.service.ts (136 bytes)
[OK] Generated service!

tibur@DESKTOP-02362TM MINGW64 /c/MisCosas/Casa/Ionic/myApp (master)
$
```

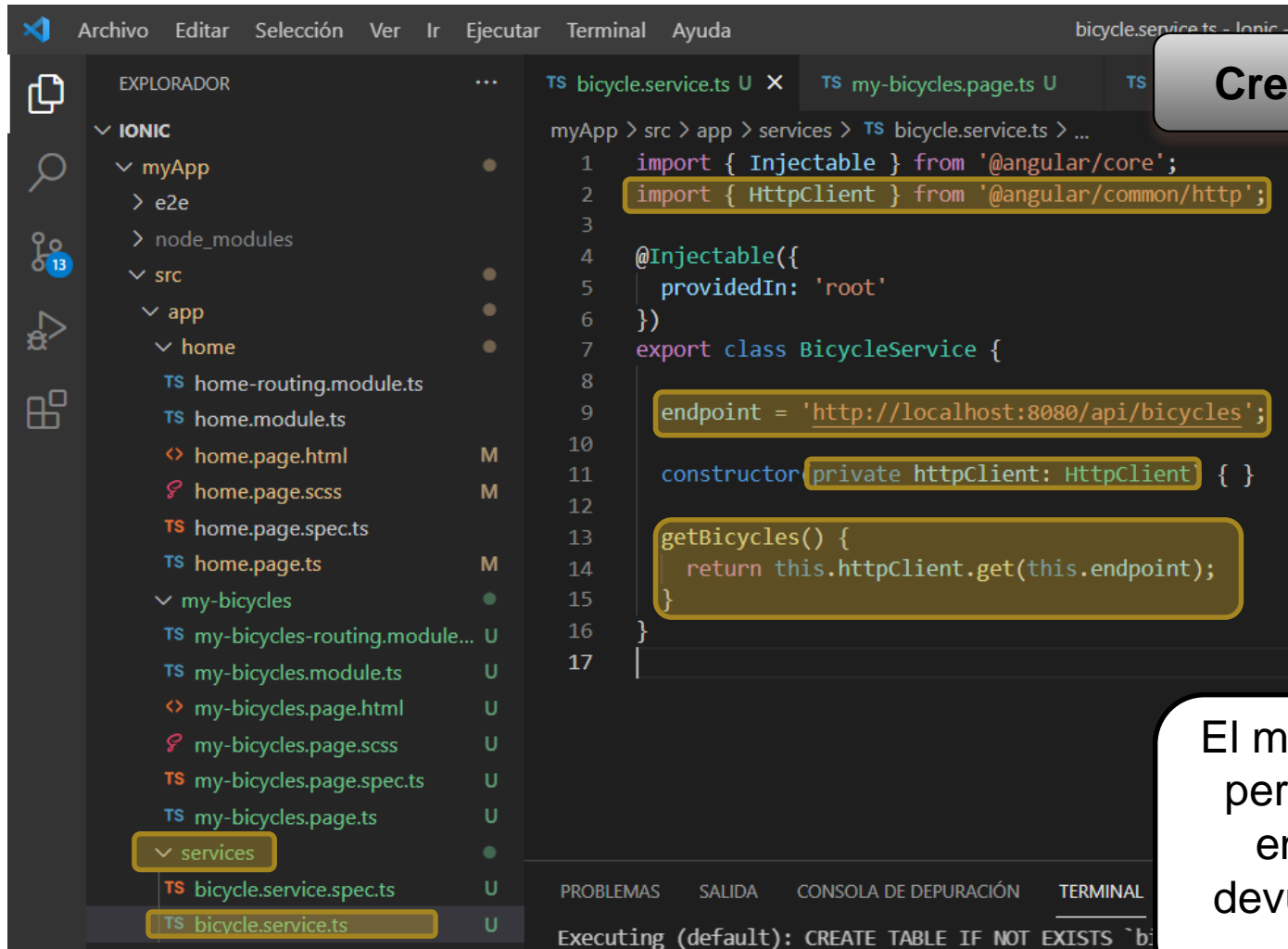
Creamos un servicio

Creamos el servicio
BicycleService al
ejecutar el comando:

**ionic generate service
services/bicycle**

Creando una App con Ionic...

Creando un servicio para consumir una API



The screenshot shows the Visual Studio Code interface with the Explorer panel on the left and the Source Editor in the center. The Explorer panel shows the project structure with the following folders and files:

- myApp
 - e2e
 - node_modules
 - src
 - app
 - home
 - home-routing.module.ts
 - home.module.ts
 - home.page.html
 - home.page.scss
 - home.page.spec.ts
 - home.page.ts
 - my-bicycles
 - my-bicycles-routing.module.ts
 - my-bicycles.module.ts
 - my-bicycles.page.html
 - my-bicycles.page.scss
 - my-bicycles.page.spec.ts
 - my-bicycles.page.ts
 - services
 - bicycle.service.spec.ts
 - bicycle.service.ts

The Source Editor shows the code for `bicycle.service.ts`:

```
1 import { Injectable } from '@angular/core';
2 import { HttpClient } from '@angular/common/http';
3
4 @Injectable({
5   providedIn: 'root'
6 })
7 export class BicycleService {
8
9   endpoint = 'http://localhost:8080/api/bicycles';
10
11   constructor(private httpClient: HttpClient) { }
12
13   getBicycles() {
14     return this.httpClient.get(this.endpoint);
15   }
16 }
17
```

Creamos un servicio

El método `getBicycles()` permite acceder a un end-point que nos devuelve un Array con las bicicletas

Creando una App con Ionic...

Creando un servicio para consumir una API

Usamos el servicio

```
myApp > src > app > my-bicycles > TS my-bicycles.page.ts > myBicyclesPage

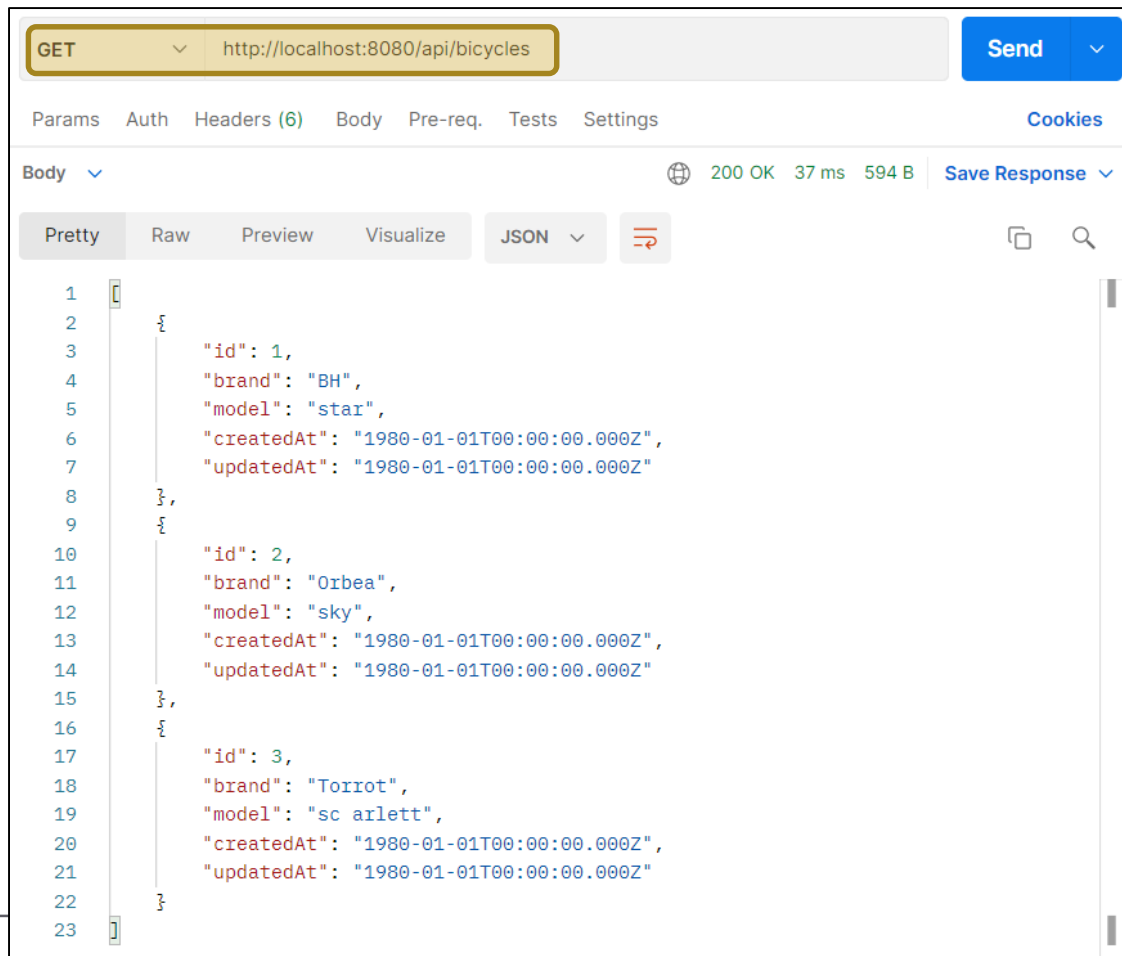
1  import { Component, OnInit } from '@angular/core';
2  import { BicycleService } from '../services/bicycle.service';
3
4  @Component({
5    selector: 'app-my-bicycles',
6    templateUrl: './my-bicycles.page.html',
7    styleUrls: ['./my-bicycles.page.scss'],
8  })
9  export class MyBicyclesPage implements OnInit {
10
11    bicycles: any = [];
12
13    constructor(private bicycleService: BicycleService) { }
14
15    ngOnInit() {
16      this.getAllBicycles();
17    }
18
19    getAllBicycles() {
20      this.bicycleService.getBicycles().subscribe(response => {
21        this.bicycles = response;
22      });
23    }
24  }
```

Transformamos nuestra anterior versión en la que usábamos datos hardcoded en **my-bicycles.page.ts** para obtener ahora los datos del servicio

Creando una App con Ionic...

Creando un servicio para consumir una API

Arranca la API



GET http://localhost:8080/api/bicycles Send

Params Auth Headers (6) Body Pre-req. Tests Settings Cookies

Body 200 OK 37 ms 594 B Save Response

Pretty Raw Preview Visualize JSON

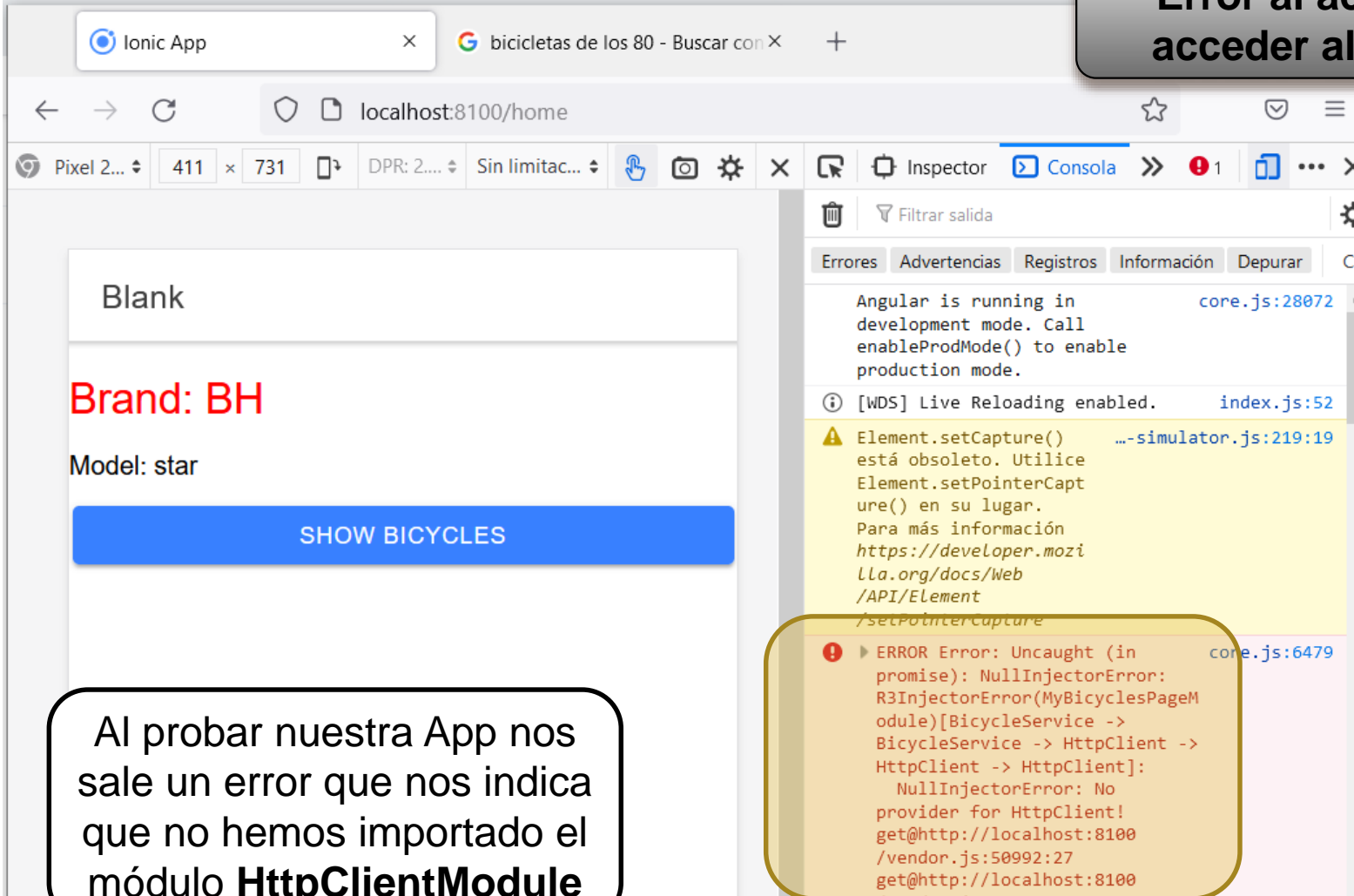
```
1 {
2   {
3     "id": 1,
4     "brand": "BH",
5     "model": "star",
6     "createdAt": "1980-01-01T00:00:00.000Z",
7     "updatedAt": "1980-01-01T00:00:00.000Z"
8   },
9   {
10    "id": 2,
11    "brand": "Orbea",
12    "model": "sky",
13    "createdAt": "1980-01-01T00:00:00.000Z",
14    "updatedAt": "1980-01-01T00:00:00.000Z"
15  },
16  {
17    "id": 3,
18    "brand": "Torrot",
19    "model": "sc arlett",
20    "createdAt": "1980-01-01T00:00:00.000Z",
21    "updatedAt": "1980-01-01T00:00:00.000Z"
22  }
23 }
```

Arranca la API
que habíamos
creado en una
práctica anterior
y asegúrate que
tiene datos que
mostrar usando
POSTMAN

Creando una App con Ionic...

Creando un servicio para consumir una API

Error al acceder al
acceder al servicio



Blank

Brand: BH

Model: star

SHOW BICYCLES

Al probar nuestra App nos sale un error que nos indica que no hemos importado el módulo **HttpClientModule**

Angular is running in development mode. Call enableProdMode() to enable production mode. core.js:28072

[WDS] Live Reloading enabled. index.js:52

Element.setCapture() está obsoleto. Utilice Element.setPointerCapture() en su lugar. Para más información https://developer.mozilla.org/docs/Web/API/Element/setPointerCapture ...-simulator.js:219:19

ERROR Error: Uncaught (in promise): NullInjectorError: R3InjectorError(MyBicyclesPageModule)[BicycleService -> BicycleService -> HttpClient -> HttpClient]: NullInjectorError: No provider for HttpClient! get@http://localhost:8100/vendor.js:50992:27 get@http://localhost:8100

Creando una App con Ionic...

Creando un servicio para consumir una API

Importamos el
módulo

En **app.module.ts** es
donde debemos incluir
el proveedor que
usamos. En este caso
el proveedor
provideHttpClient()

```
TS my-bicycles.page.ts U TS app.module.ts M X {} tsconfig.app.json TS m
end > src > app > TS app.module.ts > AppModule

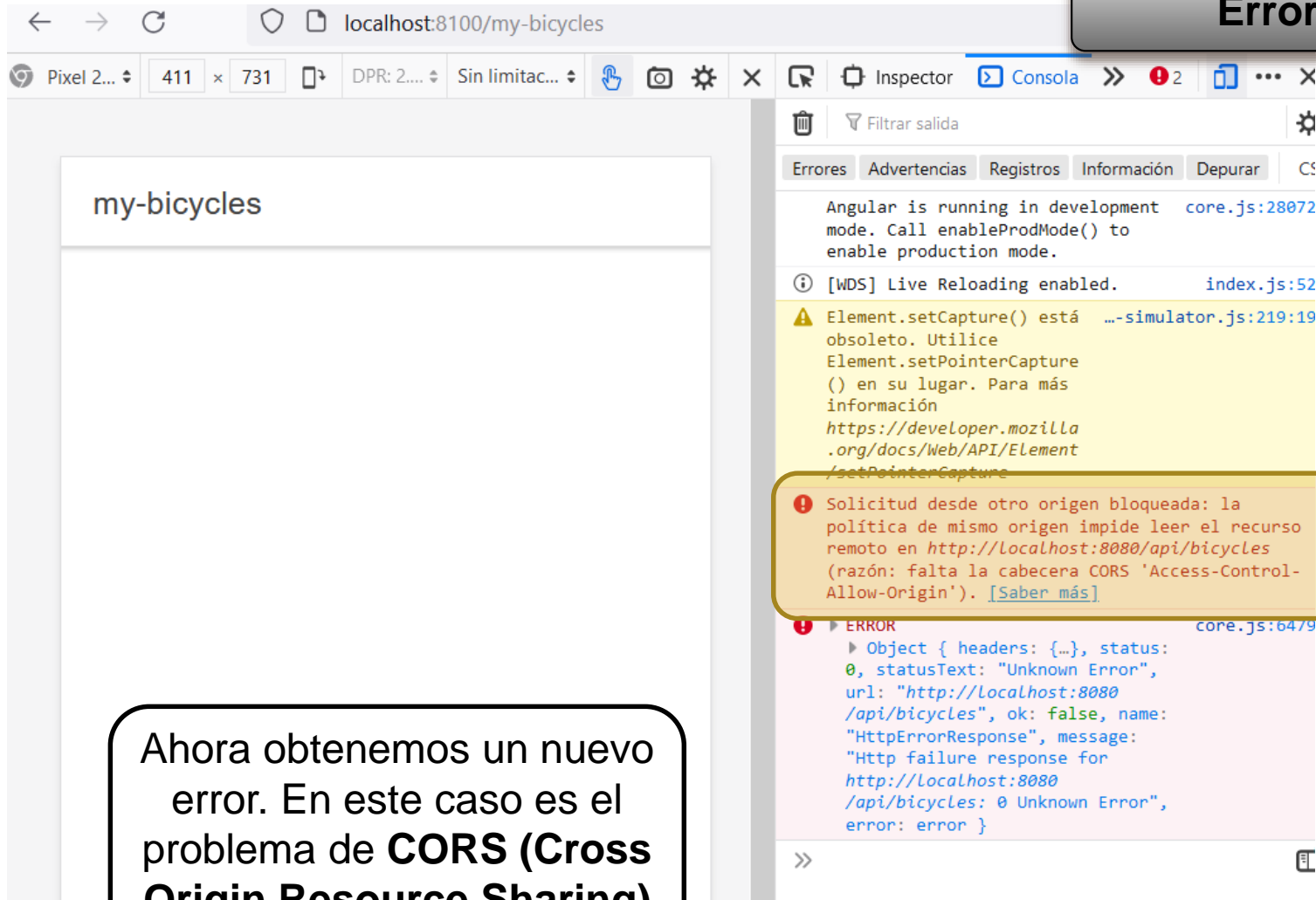
import { AppComponent } from './app.component';
import { AppRoutingModuleModule } from './app-routing.module';
import { provideHttpClient } from '@angular/common/http';

You, 1 minute ago | 1 author (You)
@NgModule({
  declarations: [AppComponent],
  imports: [BrowserModule, IonicModule.forRoot(), AppRoutingModuleModule],
  providers: [{ provide: RouteReuseStrategy, useClass: IonicRouteStrategy }, provideHttpClient()],
  bootstrap: [AppComponent],
})
export class AppModule {}
```

Creando una App con Ionic...

Creando un servicio para consumir una API

Error CORS



The screenshot shows a web browser at `localhost:8100/my-bicycles`. The page content is a simple box labeled "my-bicycles". The browser's developer console is open, showing several messages. A yellow warning message states: "Element.setCapture() está obsoleto. Utilice Element.setPointerCapture() en su lugar. Para más información https://developer.mozilla.org/docs/Web/API/Element/setPointerCapture". Below this, a red error message is highlighted with a yellow box: "Solicitud desde otro origen bloqueada: la política de mismo origen impide leer el recurso remoto en http://localhost:8080/api/bicycles (razón: falta la cabecera CORS 'Access-Control-Allow-Origin')." Below the error message, the console shows an "ERROR" object: `{ headers: {...}, status: 0, statusText: "Unknown Error", url: "http://localhost:8080/api/bicycles", ok: false, name: "HttpErrorResponse", message: "Http failure response for http://localhost:8080/api/bicycles: 0 Unknown Error", error: error }`.

my-bicycles

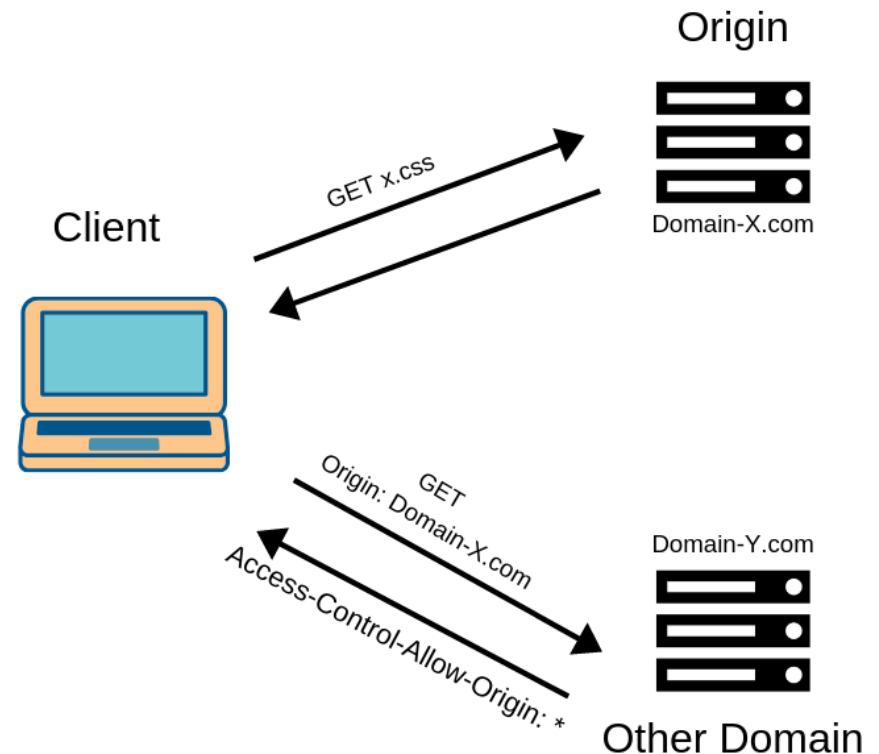
Ahora obtenemos un nuevo error. En este caso es el problema de **CORS (Cross Origin Resource Sharing)**

Creando una App con Ionic...

Creando un servicio para consumir una API

Error CORS

El Intercambio de Recursos de Origen Cruzado (**CORS**) es un mecanismo que utiliza cabeceras HTTP adicionales para permitir que un user agent (navegador, móvil, etc...) obtenga permiso para acceder a recursos seleccionados desde un servidor, en un origen distinto (dominio) al que pertenece.



Creando una App con Ionic...

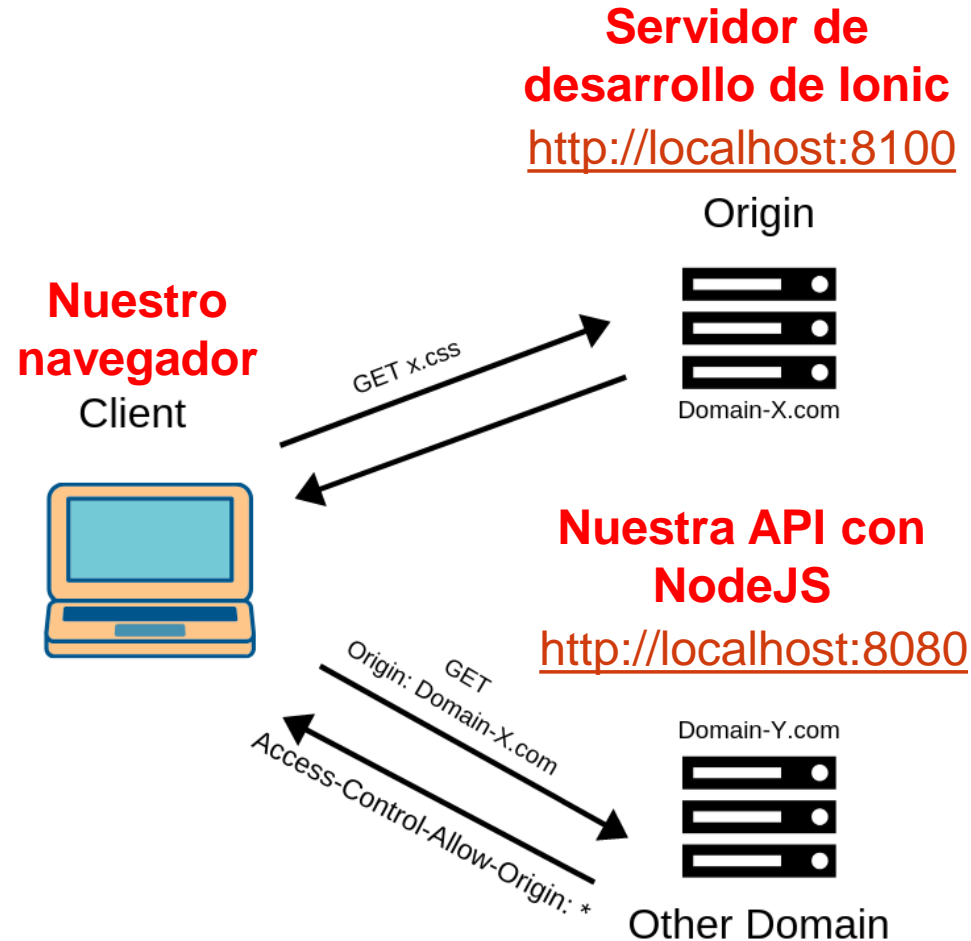
Creando un servicio para consumir una API

En nuestro caso...

Cuando arrancamos Ionic en realidad estamos arrancando un servidor de desarrollo que aloja nuestra App en

<http://localhost:8100>

Y claro nuestra API está en <http://localhost:8080> por lo que estamos accediendo a recursos de dominios cruzados



Creando una App con Ionic...

Creando un servicio para consumir una API

Devolviendo el
permiso CORS
desde la API

Para ello instalamos
el paquete **cors**, y
editamos **index.js**
para incluir el permiso
para la URL del
dominio de origen de
nuestro servidor de
desarrollo de Ionic

BACKEND

config

JS db.config.js

controllers

JS bicycle.controller.js

models

JS bicycle.model.js

JS index.js

node_modules

routes

JS bicycle.routes.js

JS index.js

{ } package-lock.json

{ } package.json

JS index.js > ...

```
1  const express = require("express");
2  const cors = require("cors");
3
4  const app = express();
5
6  var corsOptions = {
7    | origin: "http://localhost:8100"
8  };
9
10 app.use(cors(corsOptions));
11
12 // parse requests of content-type - app
13 app.use(express.json());
14
15 // parse requests of content-type - app
16 app.use(express.urlencoded({ extended:
17
18 const db = require("../models");
```

PROBLEMAS SALIDA CONSOLA DE DEPURACIÓN TERMINAL

PS C:\MisCosas\Casa\Bicycles\backend> npm install cors

added 2 packages, and audited 84 packages in 1s

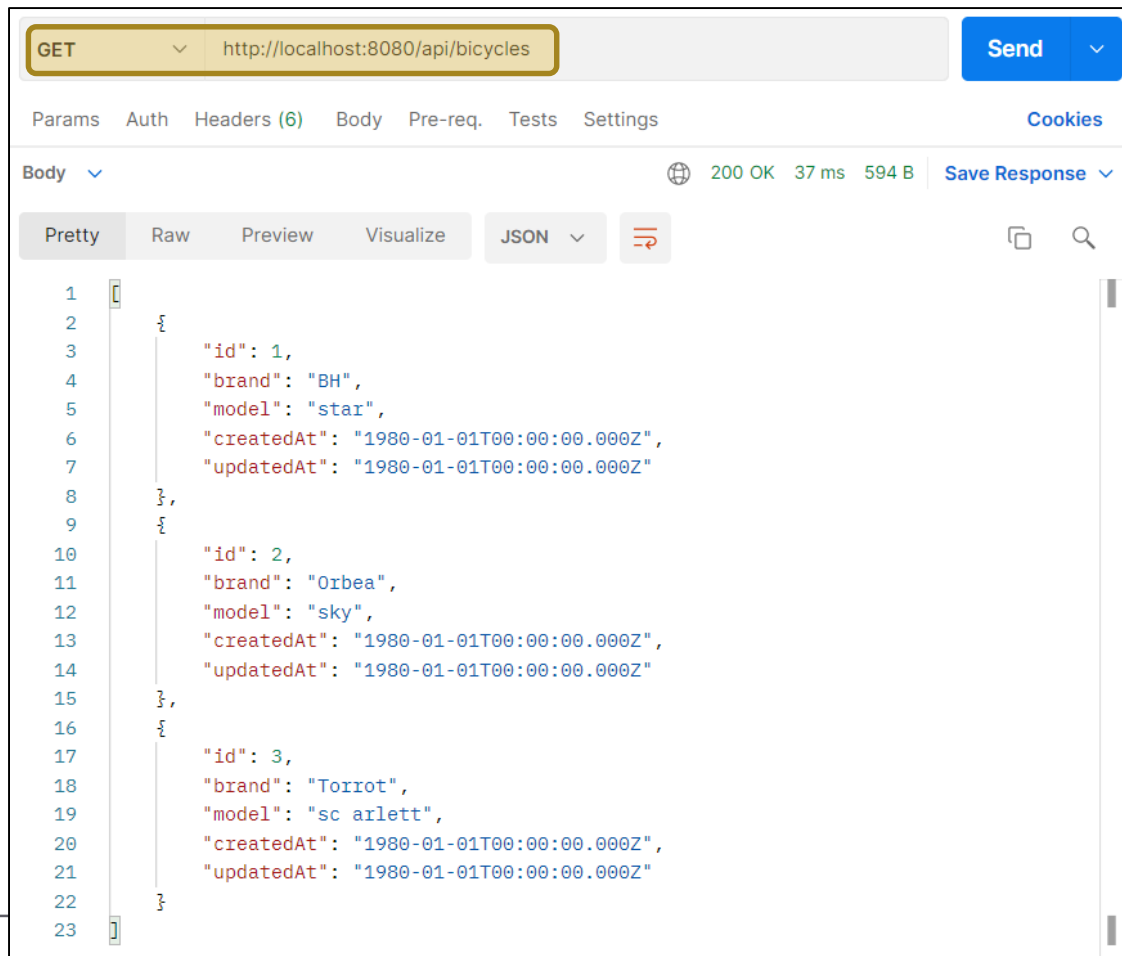
found 0 vulnerabilities

PS C:\MisCosas\Casa\Bicycles\backend> |

Creando una App con Ionic...

Creando un servicio para consumir una API

```
tibur@DESKTOP-02362TM MINGW64 /c/MisCosas/Casa/Bicycles/backend  
$ node index.js
```



```
GET http://localhost:8080/api/bicycles Send  
Params Auth Headers (6) Body Pre-req. Tests Settings Cookies  
Body 200 OK 37 ms 594 B Save Response  
Pretty Raw Preview Visualize JSON  
1 {  
2   {  
3     "id": 1,  
4     "brand": "BH",  
5     "model": "star",  
6     "createdAt": "1980-01-01T00:00:00.000Z",  
7     "updatedAt": "1980-01-01T00:00:00.000Z"  
8   },  
9   {  
10    "id": 2,  
11    "brand": "Orbea",  
12    "model": "sky",  
13    "createdAt": "1980-01-01T00:00:00.000Z",  
14    "updatedAt": "1980-01-01T00:00:00.000Z"  
15  },  
16  {  
17    "id": 3,  
18    "brand": "Torrot",  
19    "model": "sc arlett",  
20    "createdAt": "1980-01-01T00:00:00.000Z",  
21    "updatedAt": "1980-01-01T00:00:00.000Z"  
22  }  
23 }
```

Rearranca tu API

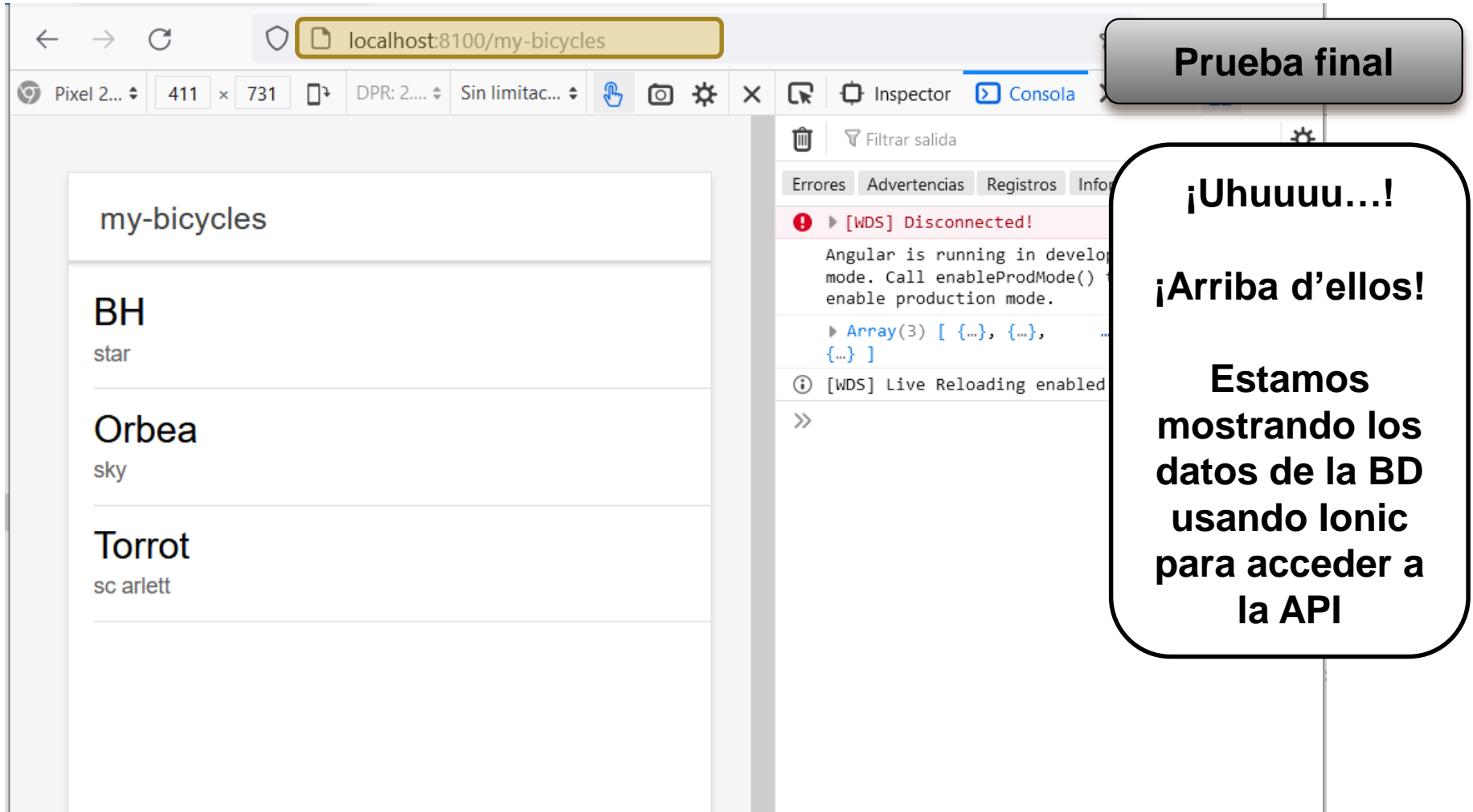
A continuación
añade algunos
registros a tu
BD y
comprueba que
hay datos

Si se te han
borrado los datos
es porque en tu
API debes quitar la
opción **force: true**

¿Recuerdas de la
anterior práctica?

Creando una App con Ionic...

Creando un servicio para consumir una API



The screenshot shows a web browser at `localhost:8100/my-bicycles`. The app displays a list of bicycles with the following details:

my-bicycles
BH star
Orbea sky
Torrot sc arlett

On the right, the browser's developer console is open, showing the following messages:

- [WDS] Disconnected!
- Angular is running in development mode. Call `enableProdMode()` to enable production mode.
- Array(3) [{...}, {...}, {...}]
- [WDS] Live Reloading enabled

A callout box on the right contains the following text:

Prueba final

¡Uhuuuu...!

¡Arriba d'ellos!

Estamos mostrando los datos de la BD usando Ionic para acceder a la API

En los siguientes pasos
debes crear el POST, PUT y
DELETE para completar el
CRUD

Pero sólo te indicaré los pasos más generales. Es importante que desarrolles la capacidad de buscar soluciones por ti misma. Tu éxito como desarrolladora depende de ello...

Creando una App con Ionic...

Creando un servicio para consumir una API

Los pasos para crear un formulario con ReactiveForms son:

- **Paso 1:** Importar **ReactiveFormsModule** en **app.module.ts**
- **Paso 2:** Importar **ReactiveFormsModule** y **FormsModule** en todos los módulos en los que se crea un formulario.
- **Paso 3:** Configurar el formulario con las validaciones
- **Paso 4:** Crear el código HTML.
- **Paso 5:** Hacer la llamada a la API desde el servicio

Consejos para
hacer el POST

Para crear el
POST necesitas
crear una
ventana con un
formulario

Puedes usar
ReactiveForms

```
@NgModule({  
  declarations: [AppComponent],  
  imports: [  
    BrowserModule,  
    IonicModule.forRoot(),  
    AppRoutingModule,  
    ReactiveFormsModule // <-- Asegúrate de incluir ReactiveFormsModule aquí  
  ],  
  providers: [{ provide: RouteReuseStrategy, useClass: IonicRouteStrategy }],  
  bootstrap: [AppComponent],  
})
```

Creando una App con Ionic...

Creando un servicio para consumir una API

Los pasos para crear un formulario con ReactiveForms son:

- **Paso 1:** Importar **ReactiveFormsModule** en **app.module.ts**
- **Paso 2:** Importar **ReactiveFormsModule** y **FormsModule** en todos los módulos en los que se crea un formulario.
- **Paso 3:** Configurar el formulario con las validaciones
- **Paso 4:** Crear el código HTML.
- **Paso 5:** Hacer la llamada a la API desde el servicio

```
@NgModule({
  imports: [
    CommonModule,
    FormsModule,
    ReactiveFormsModule, // <-- Aquí está ReactiveFormsModule
    IonicModule,
    RegistrationPageRoutingModule
  ],
  declarations: [RegistrationPage]
})
```

Consejos para
hacer el POST

Para crear el
POST necesitas
crear una
ventana con un
formulario

Puedes usar
ReactiveForms

Creando una App con Ionic...

Creando un servicio para consumir una API

```
export class BicycleFormPage implements OnInit {  
  
  bicycleForm: FormGroup;  
  
  constructor(public FormBuilder: FormBuilder,  
    private bicycleService: BicycleService,  
    private route: Router) {  
    this.bicycleForm = this.formBuilder.group({  
      brand: ['', Validators.compose([Validators.required])],  
      model: ['', Validators.compose([Validators.required])]  
    })  
  }  
  
  ngOnInit() { }  
  
  createBicycle() {  
    if (this.bicycleForm.valid) {  
      console.log('Formulario válido:', this.bicycleForm.value);  
      this.bicycleService.create(this.bicycleForm.value).subscribe(response => {  
        this.route.navigateByUrl("/my-bicycles");  
      })  
    } else {  
      console.log('Formulario no válido');  
    }  
  }  
  
  getFormControl(field: string) {  
    return this.bicycleForm.get(field);  
  }  
}
```

**Consejos para
hacer el POST**

**Para crear el
POST necesitas
crear una
ventana con un
formulario**

**Puedes usar
ReactiveForms**

**- Paso 3: Configurar el formulario con las
validaciones**

Creando una App con Ionic...

Creando un servicio para consumir una API

- Paso 4: Crear el código HTML

Consejos para
hacer el POST

```
<form [formGroup]="bicycleForm" (ngSubmit)="createBicycle()">

  <ion-item>
    <ion-label position="floating">Brand</ion-label>
    <ion-input formControlName="brand" type="text"></ion-input>
  </ion-item>
  <ion-note color="danger" *ngIf="getFormControl('brand')?.hasError('required') && getFormControl('brand')?.touched">
    El brand es obligatorio.
  </ion-note>

  <ion-item>
    <ion-label position="floating">Model</ion-label>
    <ion-input formControlName="model" type="text"></ion-input>
  </ion-item>
  <ion-note color="danger" *ngIf="getFormControl('model')?.hasError('required') && getFormControl('model')?.touched">
    El model es obligatorio.
  </ion-note>

  <!-- Botón de Enviar -->
  <ion-button expand="full" type="submit" [disabled]="!bicycleForm.valid">Crear</ion-button>

</form>
```

Creando una App con Ionic...

Creando un servicio para consumir una API

Los pasos para crear un formulario con ReactiveForms son:

- **Paso 1:** Importar **ReactiveFormsModule** en **app.module.ts**
- **Paso 2:** Importar **ReactiveFormsModule** y **FormsModule** en todos los módulos en los que se crea un formulario.
- **Paso 3:** Configurar el formulario con las validaciones
- **Paso 4:** Crear el código HTML.
- **Paso 5:** Hacer la llamada a la API desde el servicio

Consejos para
hacer el POST

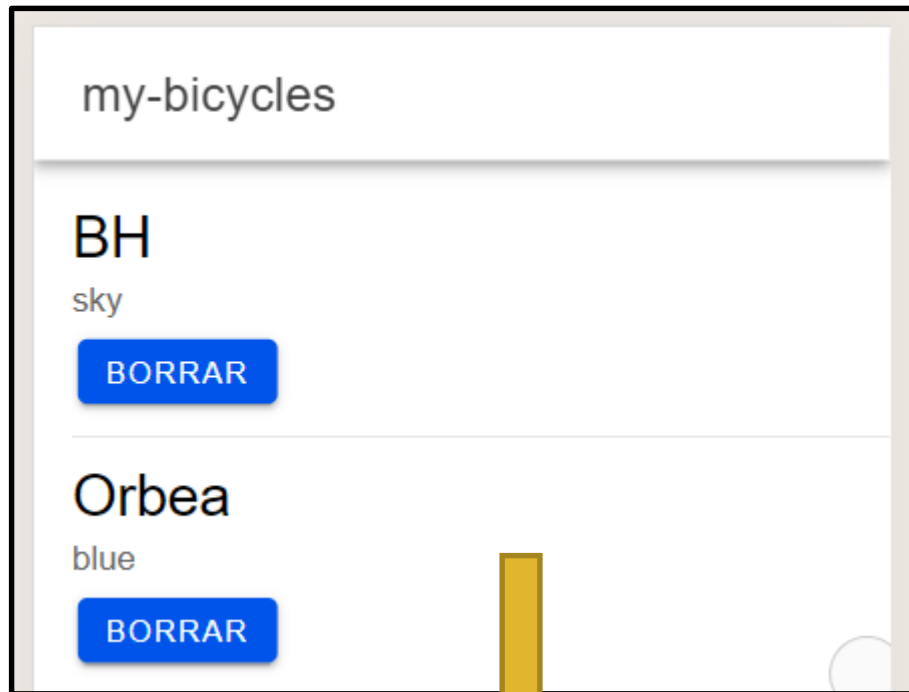
Para crear el
POST necesitas
crear una
ventana con un
formulario

Puedes usar
ReactiveForms

```
create(bicycle: any){  
  const headers = new HttpHeaders({  
    'Content-Type': 'application/x-www-form-urlencoded'  
  });  
  
  const body = new URLSearchParams();  
  body.append("brand", bicycle.brand);  
  body.append("model", bicycle.model);  
  
  return this.httpClient.post(this.serverUrl, body.toString(), { headers });  
}
```

Creando una App con Ionic...

Creando un servicio para consumir una API



**Consejos para
hacer el DELETE**


**Para hacer el
DELETE puedes
crear un botón
en el listado de
bicicletas**

Esto es lo que pretendemos conseguir

Creando una App con Ionic...

Creando un servicio para consumir una API

En el HTML debes incluir el botón y llamar a la función cuando se haga clic sobre dicho botón pasando el id de la bicicleta a borrar



Consejos para hacer el DELETE

Para hacer el DELETE puedes crear un botón en el listado de bicicletas

```
<ion-list>
  <ion-item *ngFor="let b of bicycles">
    <ion-label>
      <h1>{{b.brand}}</h1>
      <p>{{b.model}}</p>
      <ion-button (click)="deleteBicycle(b.id)">borrar</ion-button>
    </ion-label>
  </ion-item>
</ion-list>
```

Creando una App con Ionic...

Creando un servicio para consumir una API

**Consejos para
hacer el DELETE**

**Para hacer el
DELETE puedes
crear un botón
en el listado de
bicicletas**

**En el código de la clase haz la
llamada al servicio.**

```
deleteBicycle(id: any){  
  this.bicycleService.delete(id).subscribe(response => {  
    |  
  })  
}
```


Creando una App con Ionic...

Creando un servicio para consumir una API

Consejos para
hacer el DELETE

Para hacer el
DELETE puedes
crear un botón
en el listado de
bicicletas

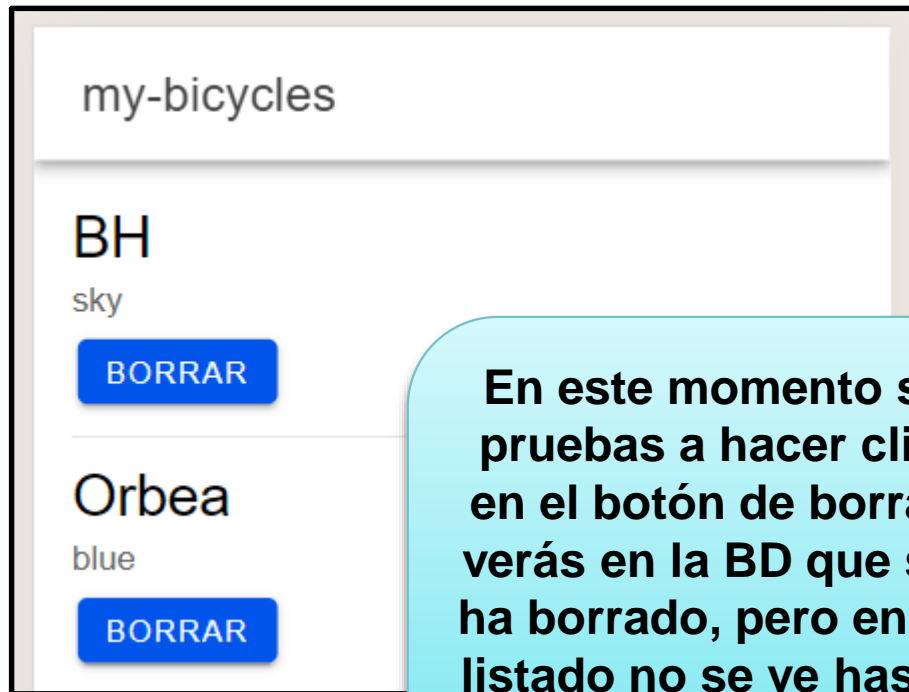
La llamada a la API en el
servicio debe incluir el id de la
bicicleta a borrar



```
delete(id: any){  
  return this.httpClient.delete(`${this.serverUrl}/${id}`);  
}
```

Creando una App con Ionic...

Creando un servicio para consumir una API



En este momento si pruebas a hacer clic en el botón de borrar verás en la BD que se ha borrado, pero en el listado no se ve hasta que no recargas la página.

Consejos para
hacer el DELETE

Para hacer el
DELETE puedes
crear un botón
en el listado de
bicicletas

¿Por qué?

Creando una App con Ionic...

Creando un servicio para consumir una API

Consejos para
hacer el DELETE

Cuando la llamada asíncrona se completa puedes volver a recargar el listado de todas las bicicletas, y verás que ahora sí se ve el resultado

Para hacer el DELETE puedes crear un botón en el listado de bicicletas

```
deleteBicycle(id: any){  
  this.bicycleService.delete(id).subscribe(response => {  
    this.getAllBicycles();  
  })  
}
```

Sigue aprendiendo...

Ya sólo te falta el UPDATE... Es más de lo mismo... seguro que lo consigues autónomamente...

¡Cuidado con las versiones!

A fecha de la creación de este tutorial deberías trabajar con la versión 8 de Ionic.

Los tutoriales incluso de la versión de Ionic 5 te pueden aún servir en su mayoría.

Pero los anteriores ya tienen bastantes diferencias.

Conclusiones

¿Qué hemos aprendido?

- Simplemente hemos consumido una API para mostrar los datos obtenidos mediante el método GET en Ionic.
- Para ello hemos tenido que aprender en Ionic como crear una página, el enrutamiento para pasar de página a página, hemos creado un servicio, hemos importado un módulo, y seguro que alguna cosilla más que se me escapa...
- También hemos aprendido lo que es CORS.

Próximos pasos...

- Haz que el resultado sea atractivo visualmente. Para ello estudia los “UI Components” de Ionic. Mira el enlace: <https://ionicframework.com/docs/components>
-