

IONIC

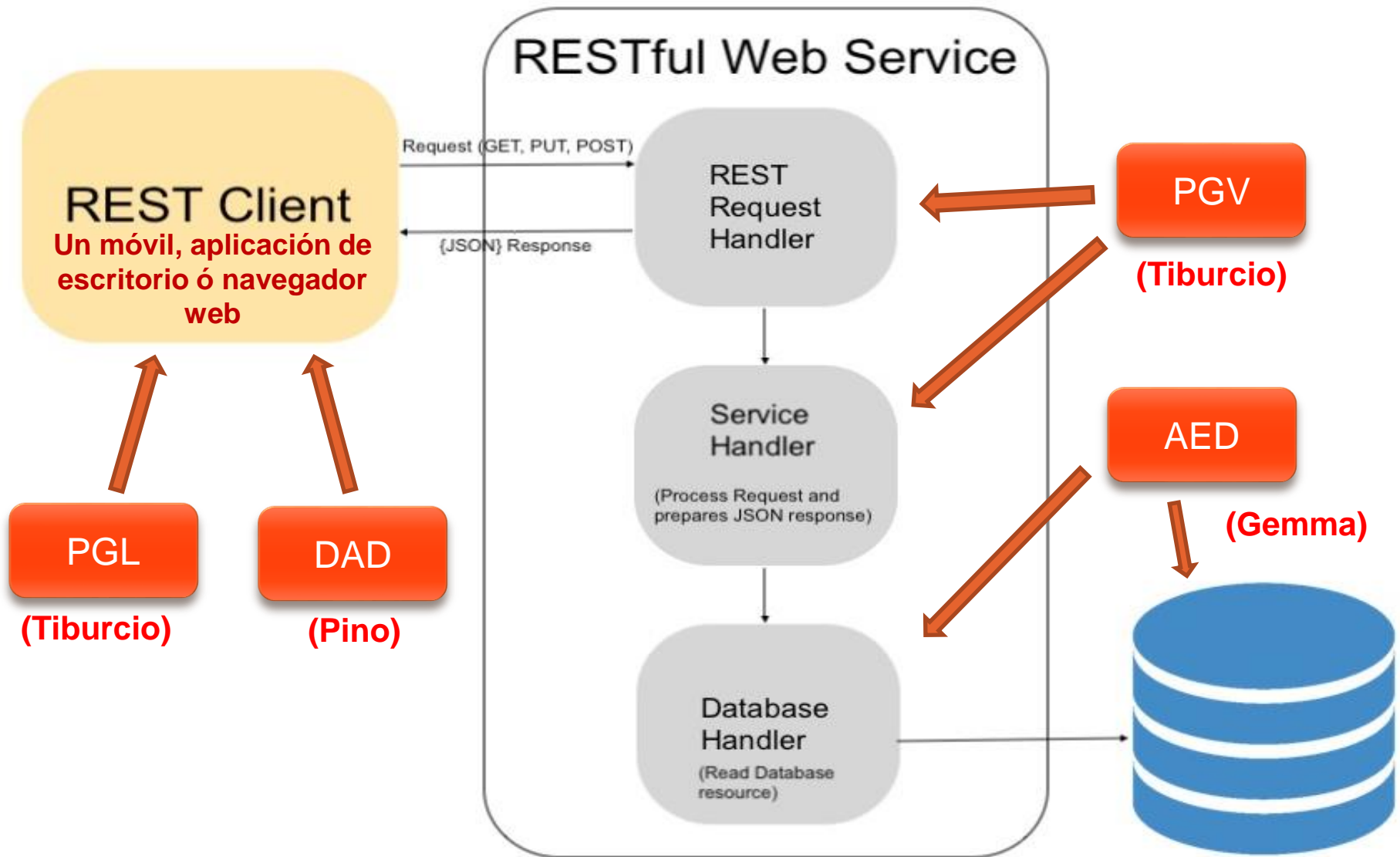
consumiendo API

(Usando Angular)

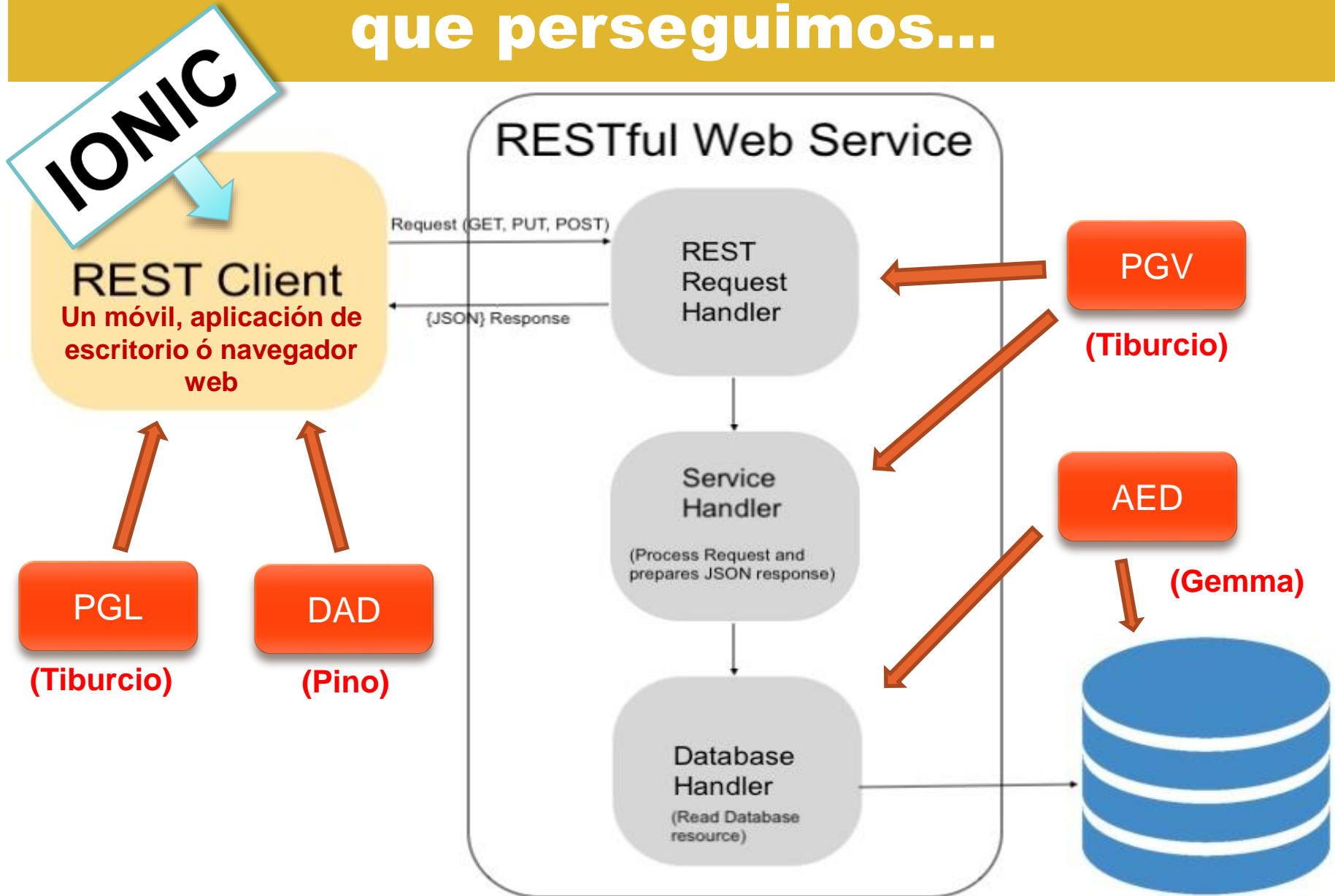
Resumen de pasos basado en la web:

<https://remotestack.io/ionic-http-requests-with-httpclient-get-post-put-delete-tutorial/>

No perdamos nunca la visión global que perseguimos...



No perdamos nunca la visión global que perseguimos...



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.

```
$ 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
```


Al lío... 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...

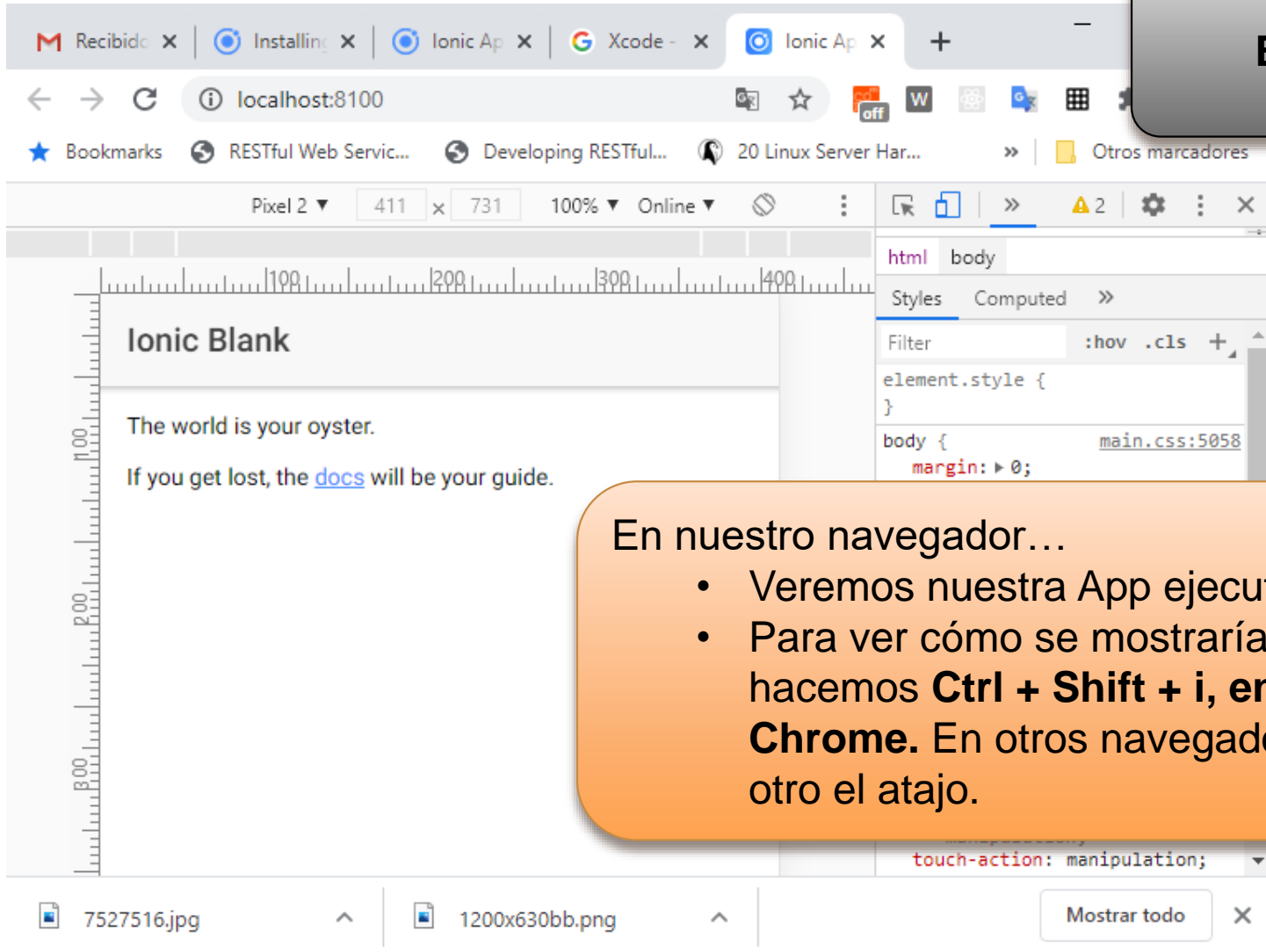
Ejecutamos nuestra primera App con Ionic:

- **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.

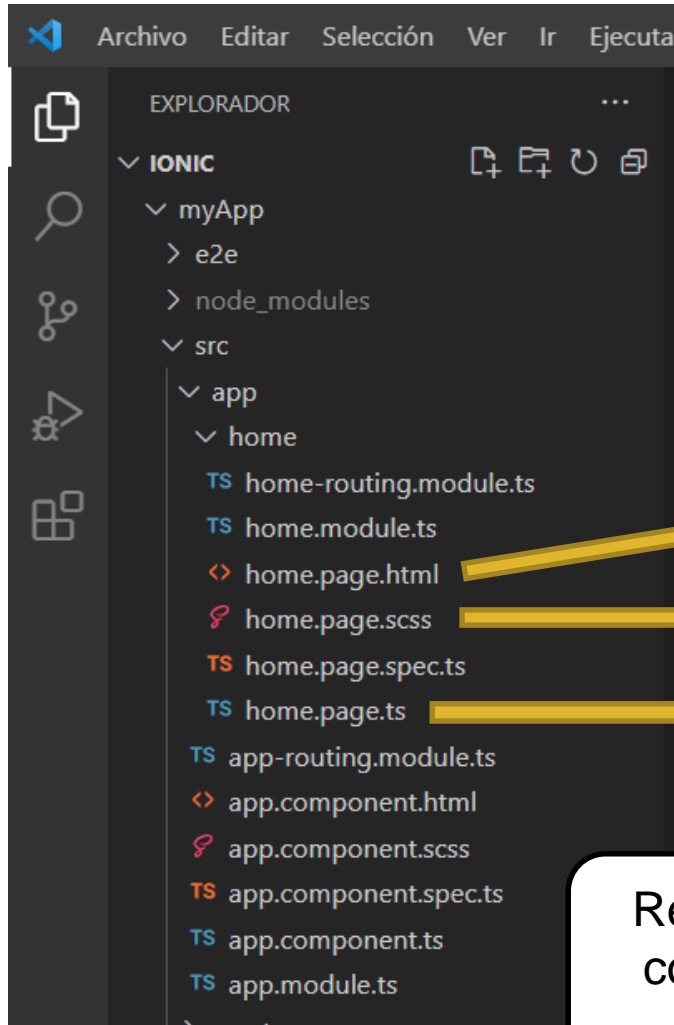
Comencemos a picar
código.

**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

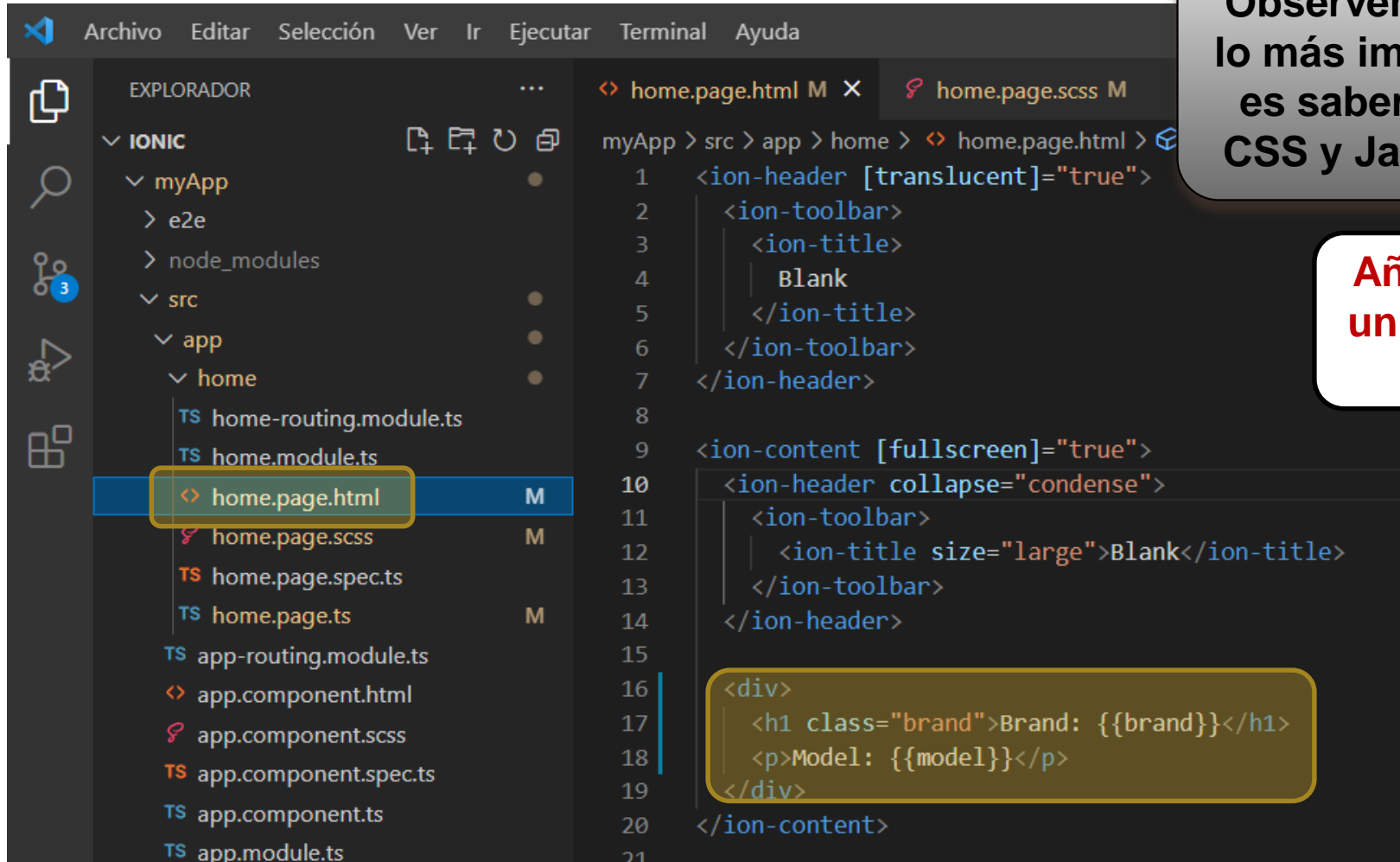
Realmente usamos el framework Angular y trabajamos con SCSS para los estilos y con TypeScript en vez de JavaScript

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 an IDE with a dark theme. On the left, the 'EXPLORADOR' (Explorer) panel displays the project structure. The 'src' folder is expanded, showing 'app' and 'home' subfolders. The 'home' folder contains several files, with 'home.page.html' highlighted. On the right, the 'home.page.html' file is open, showing HTML code for an Ionic page. The code includes an ion-header, ion-toolbar, ion-title, ion-content, and ion-header sections. A yellow box highlights a div element containing a h1 and a p tag with placeholder text.

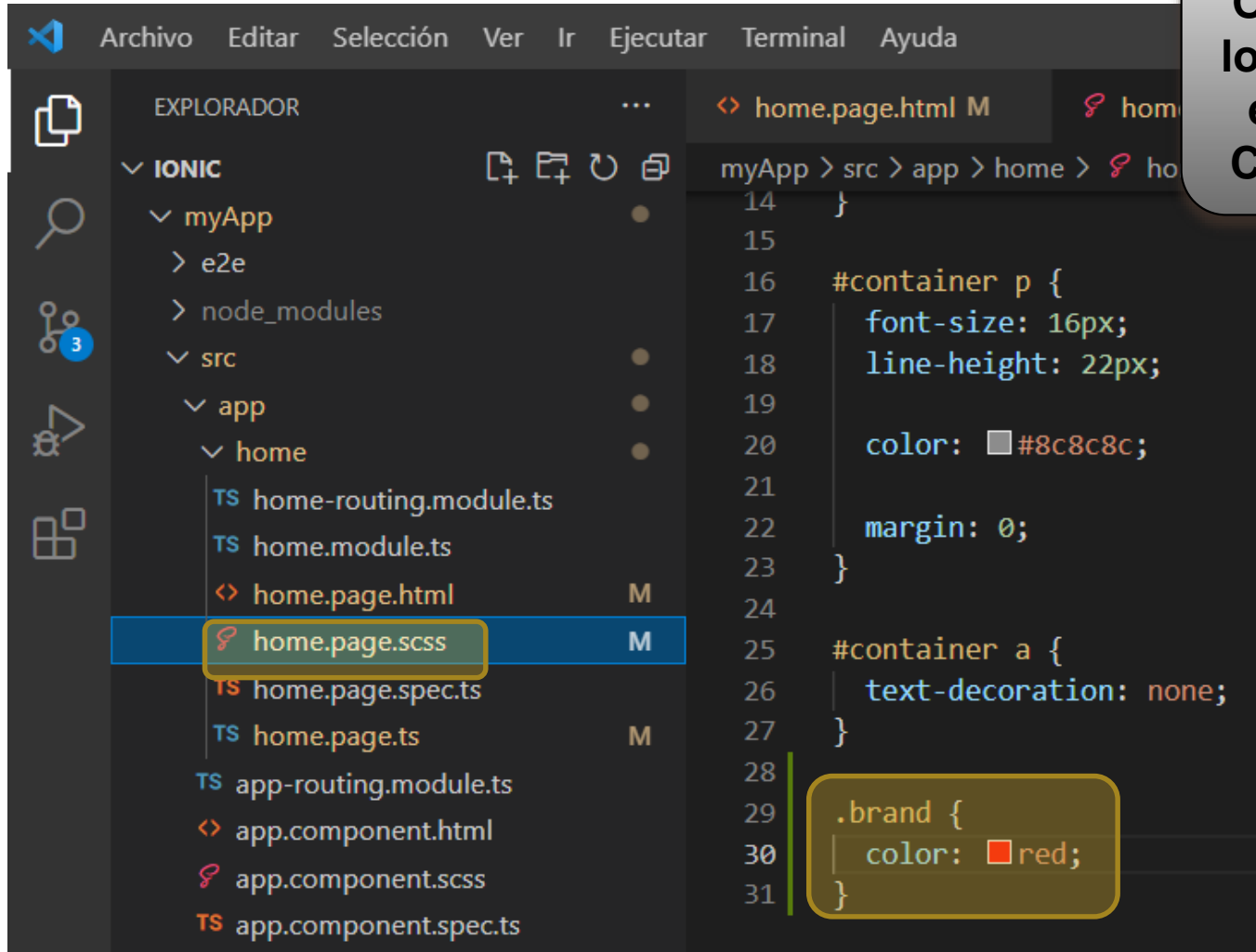
```
1 <ion-header [translucent]="true">
2   <ion-toolbar>
3     <ion-title>
4       Blank
5     </ion-title>
6   </ion-toolbar>
7 </ion-header>
8
9 <ion-content [fullscreen]="true">
10  <ion-header collapse="condense">
11    <ion-toolbar>
12      <ion-title size="large">Blank</ion-title>
13    </ion-toolbar>
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 (highlighted with a yellow box)
 - home.page.spec.ts
 - home.page.ts
 - app-routing.module.ts
 - app.component.html
 - app.component.scss
 - app.component.spec.ts

The main editor displays the content of `home.page.html`, which includes SCSS code for styling. The code is as follows:

```
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 }
```

The SCSS code is highlighted with a yellow box, and the `home.page.scss` file in the Explorer is also highlighted with a yellow box.

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 an Ionic app project. The Explorer sidebar on the left shows the project structure:

- 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** (highlighted)
 - app-routing.module.ts
 - app.component.html

The main editor shows the code for `home.page.html` (labeled as `home.page.ts` in the title bar). The code is as follows:

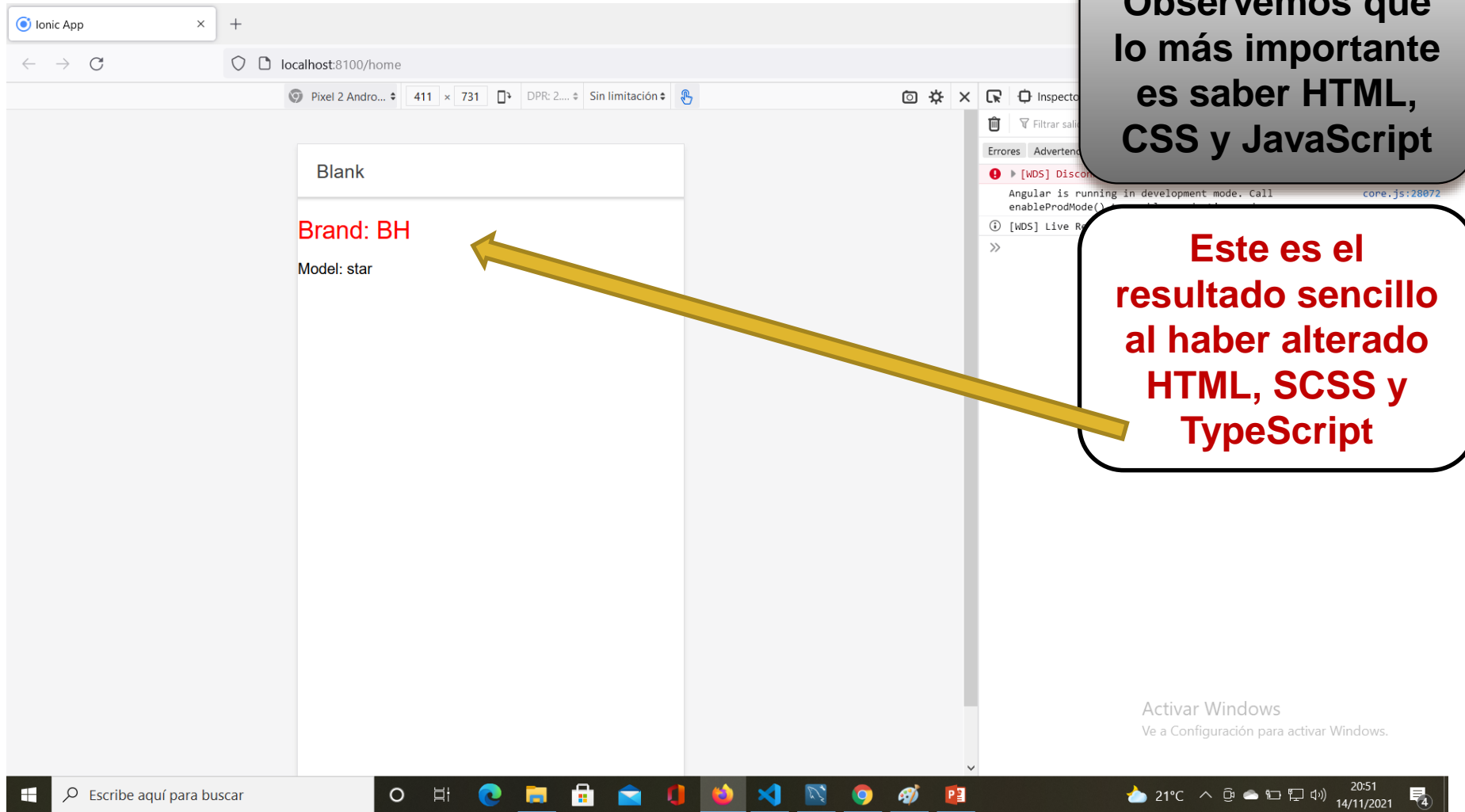
```
1 import { Component } from '@angular/core';
2
3 @Component({
4   selector: 'app-home',
5   templateUrl: 'home.page.html',
6   styleUrls: ['home.page.scss'],
7 })
8 export class HomePage {
9
10   brand: string = "BH";
11   model: string = "star";
12
13   constructor() {}
14
15 }
16
```

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

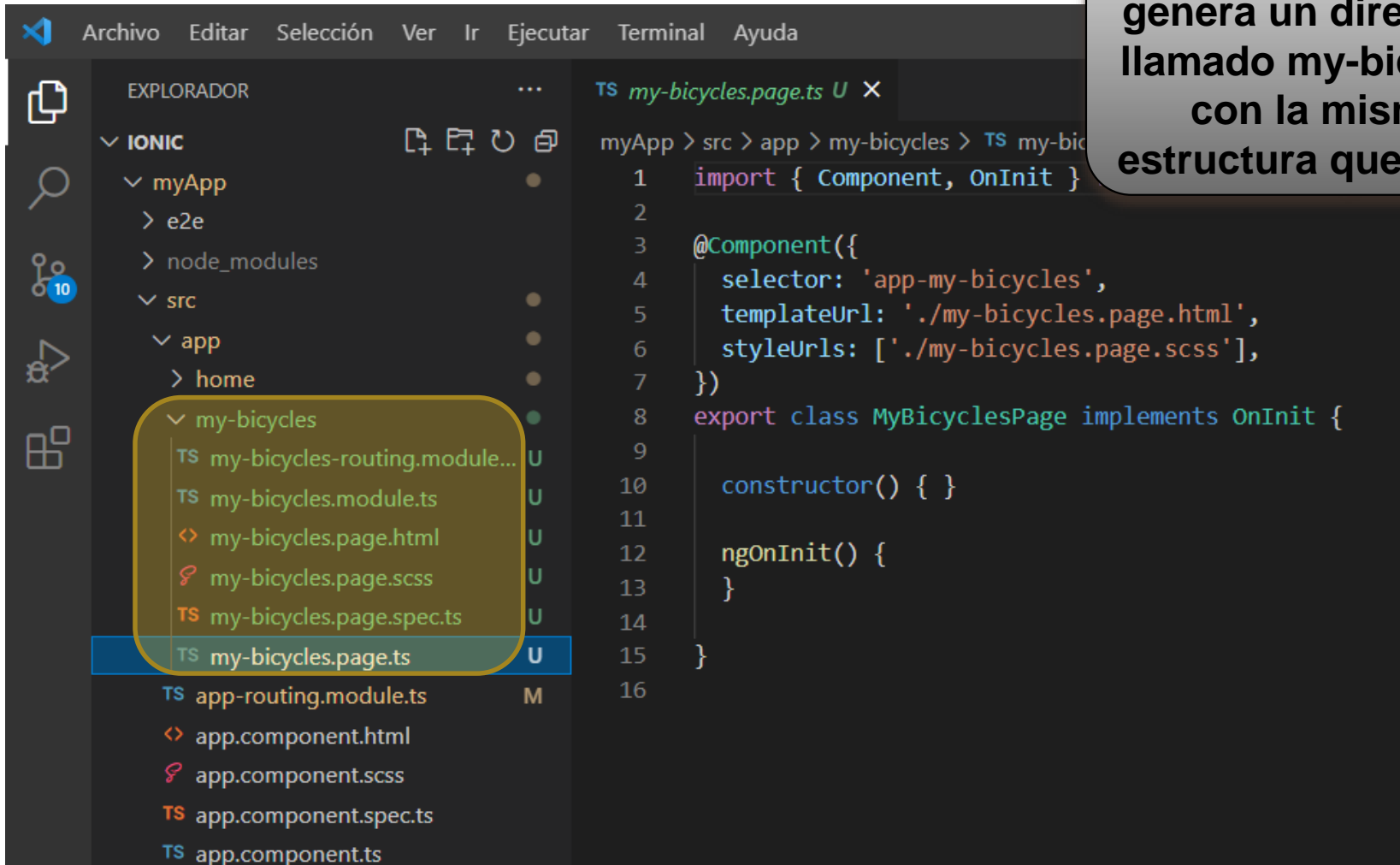
```
tibur@DESKTOP-02362TM MINGW64 /c/MisCosas/Casa/Ionic/myApp
$ ionic generate page my-bicycles
> ng.cmd generate page my-bicycles --project=app
CREATE src/app/my-bicycles/my-bicycles-routing.module.ts (364 bytes)
CREATE src/app/my-bicycles/my-bicycles.module.ts (502 bytes)
CREATE src/app/my-bicycles/my-bicycles.page.html (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)
$
```

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



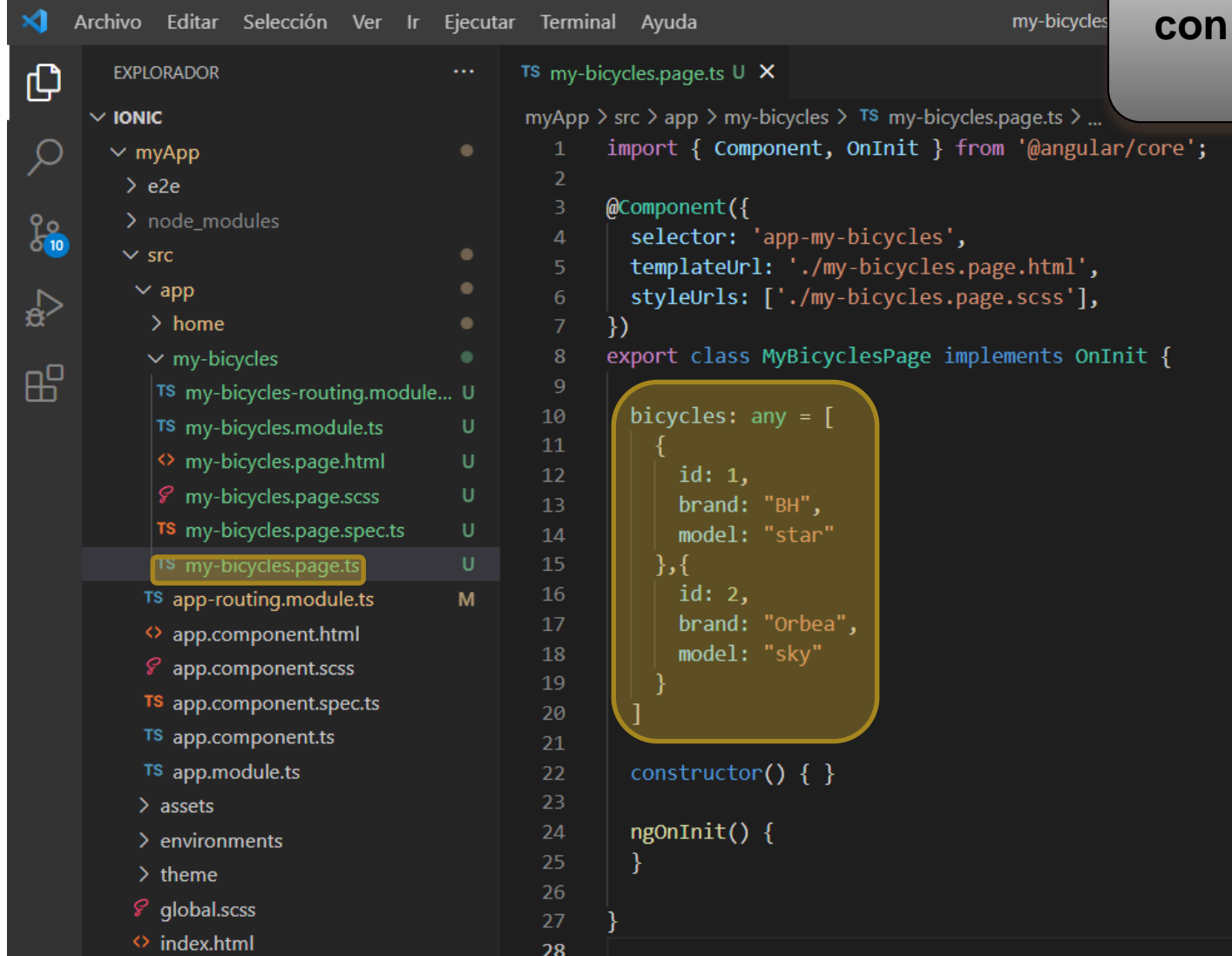
The screenshot shows the Visual Studio Code interface. On the left, the 'EXPLORADOR' (Explorer) sidebar displays the project structure. The 'myApp' folder is expanded, showing subfolders like 'e2e', 'node_modules', 'src', and 'app'. Inside 'src/app', there is a 'home' directory and a newly created 'my-bicycles' directory. The 'my-bicycles' directory contains several 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. On the right, the editor shows the content of 'my-bicycles.page.ts'. It starts with an import statement for 'Component' and 'OnInit' from '@ionic/angular'. Below this, the '@Component' decorator is used to define the page's metadata, including a 'selector', 'templateUrl', and 'styleUrls'. Finally, the 'MyBicyclesPage' class is exported, implementing the 'OnInit' interface, with a 'constructor' and an 'ngOnInit' method.

```
1 import { Component, OnInit } from '@ionic/angular';
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



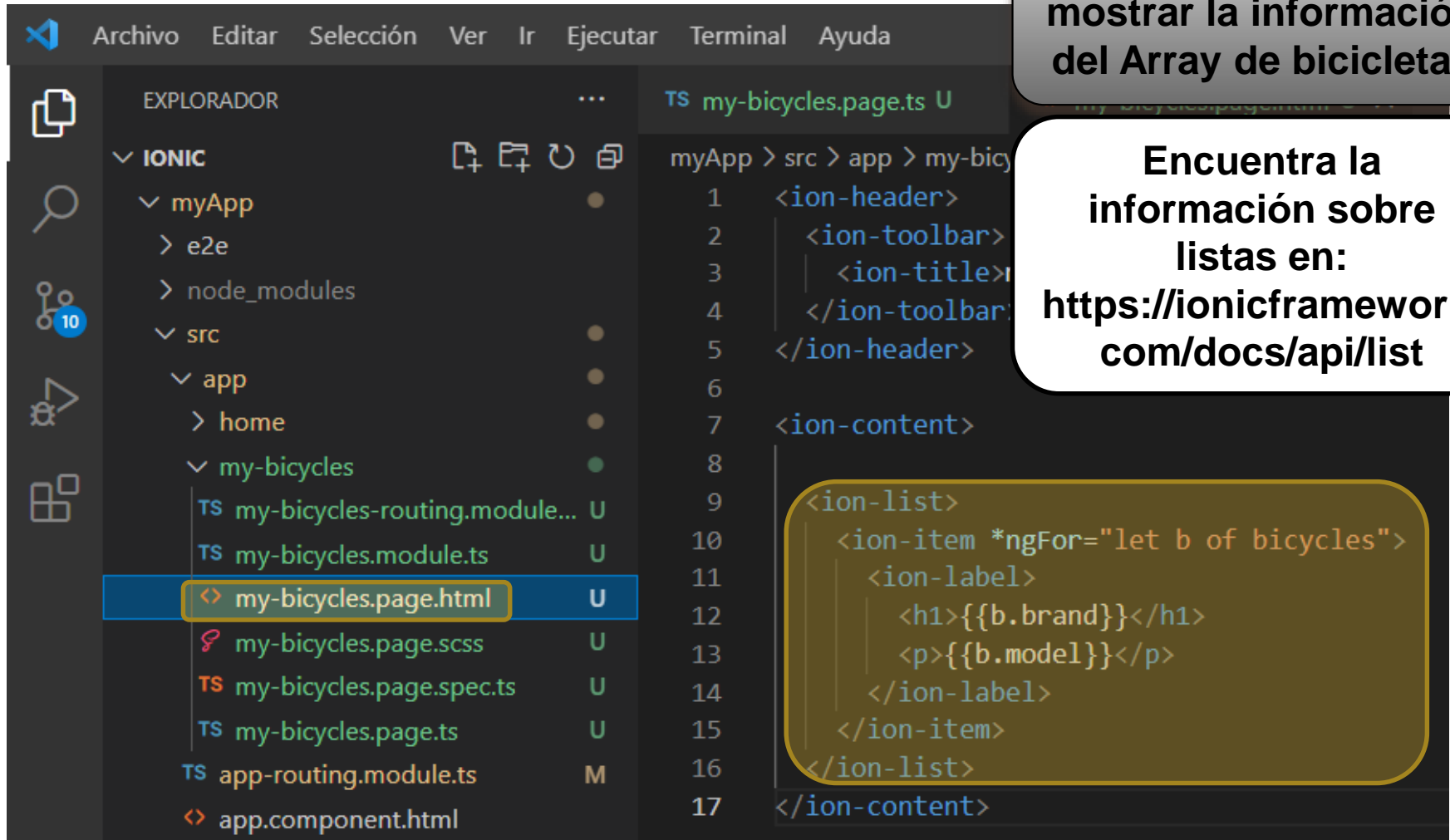
```
myApp > src > app > my-bicycles > TS my-bicycles.page.ts > ...
1  import { Component, OnInit } from '@angular/core';
2
3  @Component({
4    selector: 'app-my-bicycles',
5    templateUrl: './my-bicycles.page.html',
6    styleUrls: ['./my-bicycles.page.scss'],
7  })
8  export class MyBicyclesPage implements 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

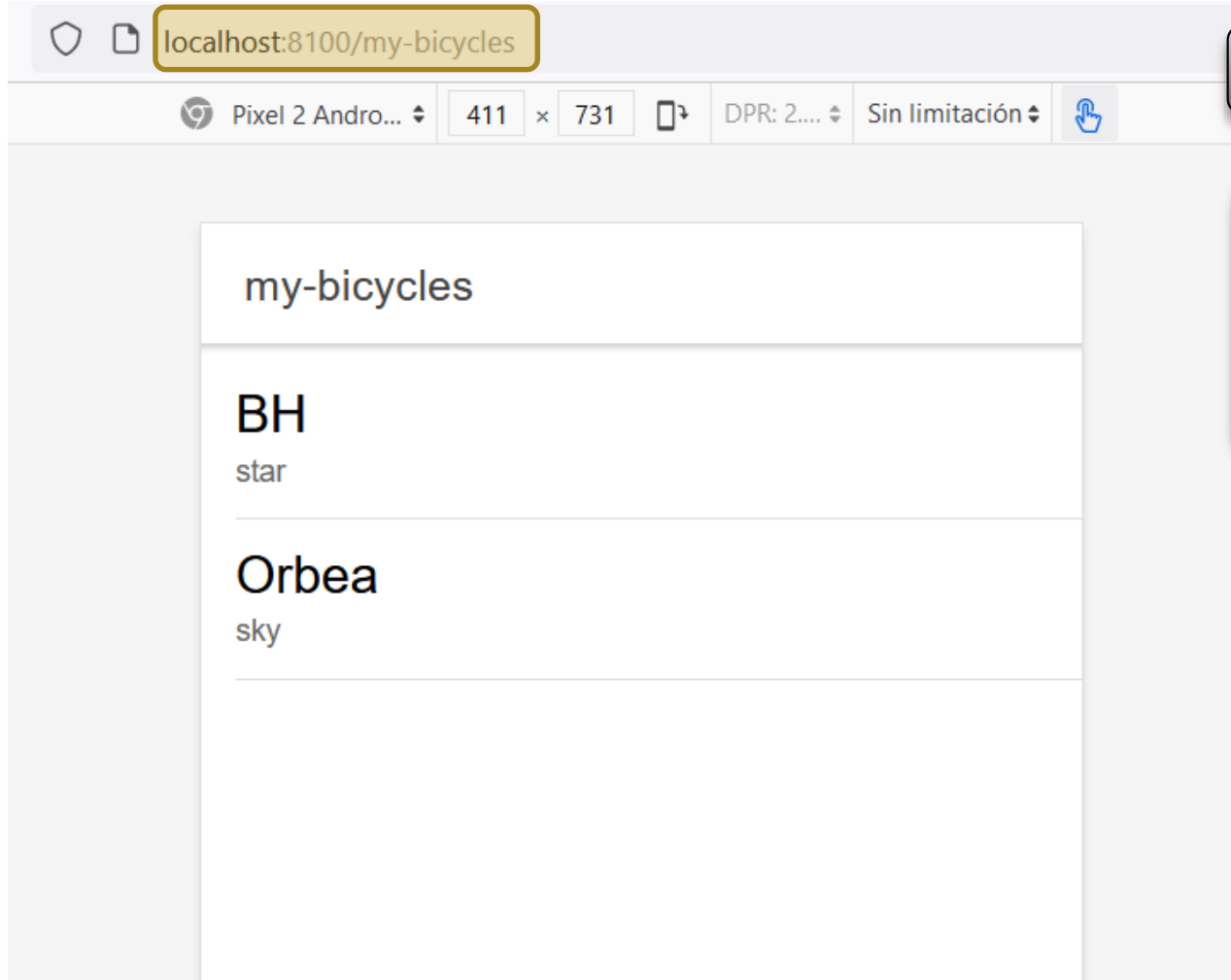
Encuentra la información sobre listas en:
<https://ionicframework.com/docs/api/list>



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

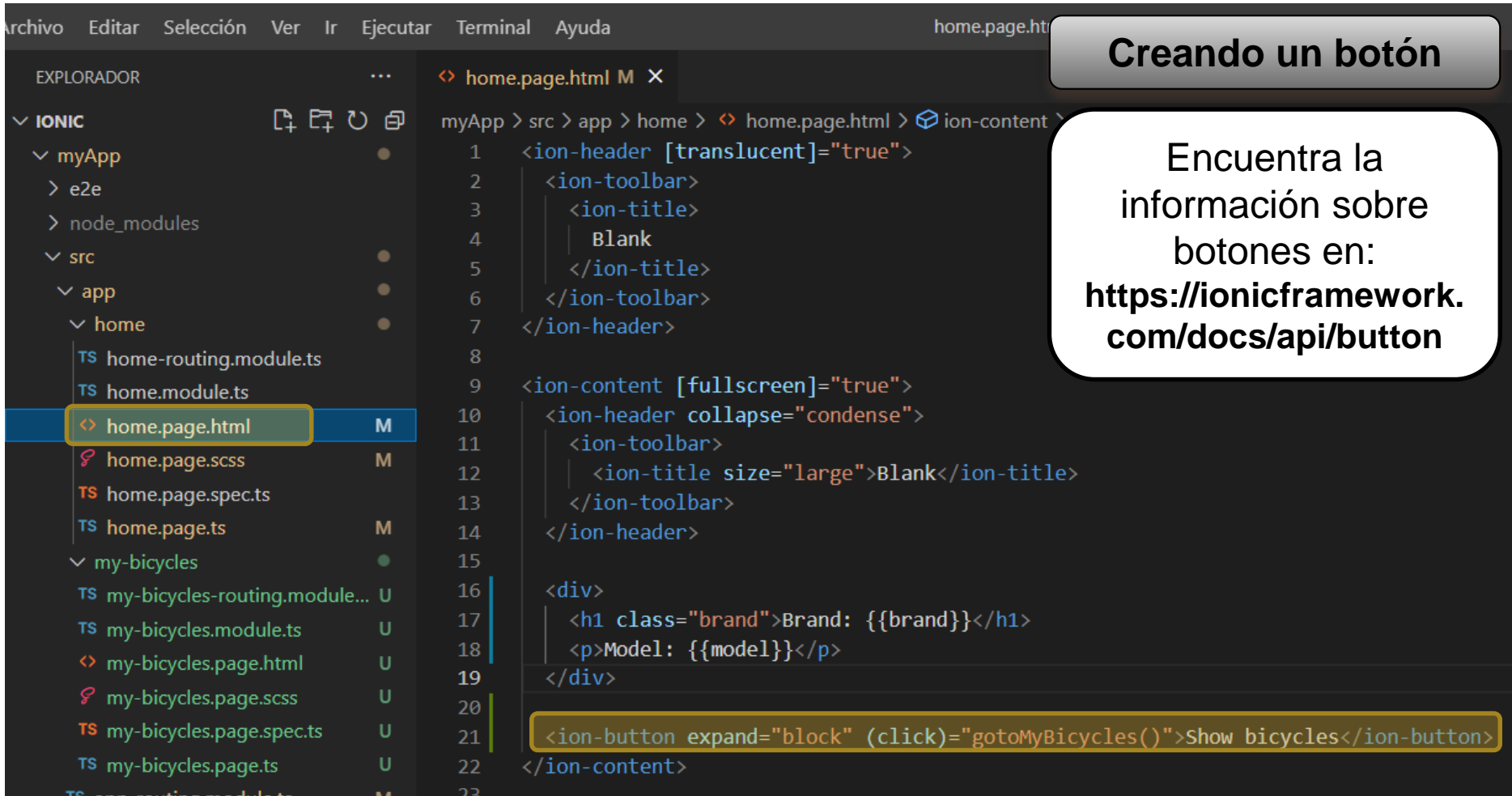
En vez de cambiar la URL a mano vamos a pasar de una página a otra a través de un botón.

Creando una App con Ionic...

Pasar de una página a otra

Creando un botón

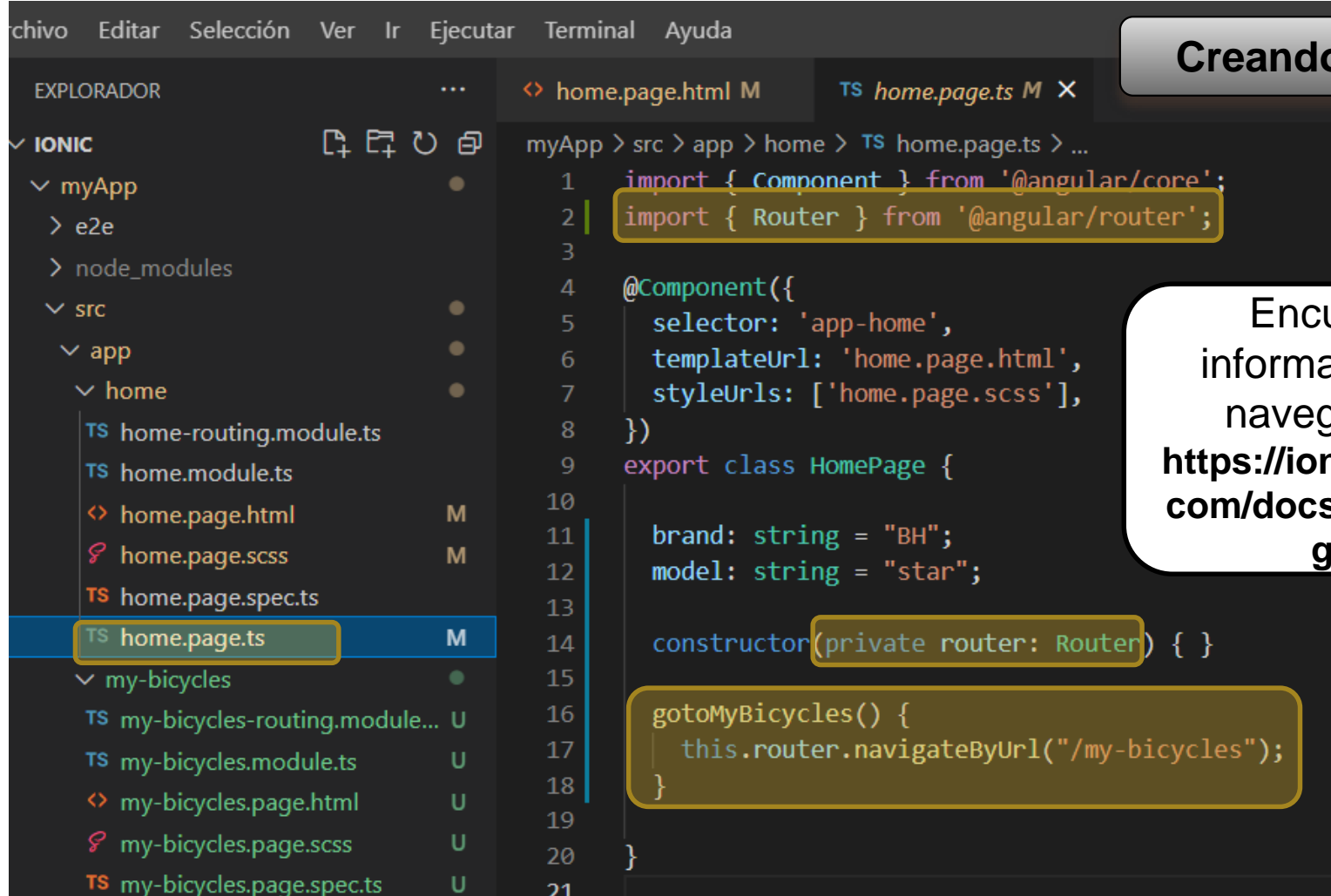
Encuentra la información sobre botones en:
<https://ionicframework.com/docs/api/button>



```
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 a dark theme. On the left, the 'EXPLORADOR' (Explorer) panel shows a project structure with folders 'myApp', 'src', and 'app'. Under 'app', there is a 'home' folder containing several files, including 'home.page.ts' which is selected and highlighted with a blue bar. The main editor area shows the code for 'home.page.ts'. The code includes imports for 'Component' and 'Router' from '@angular/core' and '@angular/router' respectively. It defines a class 'HomePage' with a selector 'app-home', a templateUrl 'home.page.html', and styleUrls ['home.page.scss']. The class has two properties: 'brand' of type 'string' with value 'BH', and 'model' of type 'string' with value 'star'. The constructor takes a 'private router: Router' as an argument. A method 'gotoMyBicycles()' is defined, which calls 'this.router.navigateByUrl("/my-bicycles");'. The file explorer on the left also shows other files like 'home-routing.module.ts', 'home.module.ts', 'home.page.html', 'home.page.scss', 'home.page.spec.ts', and files for 'my-bicycles'.

```
myApp > src > app > home > TS home.page.ts > ...
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
```

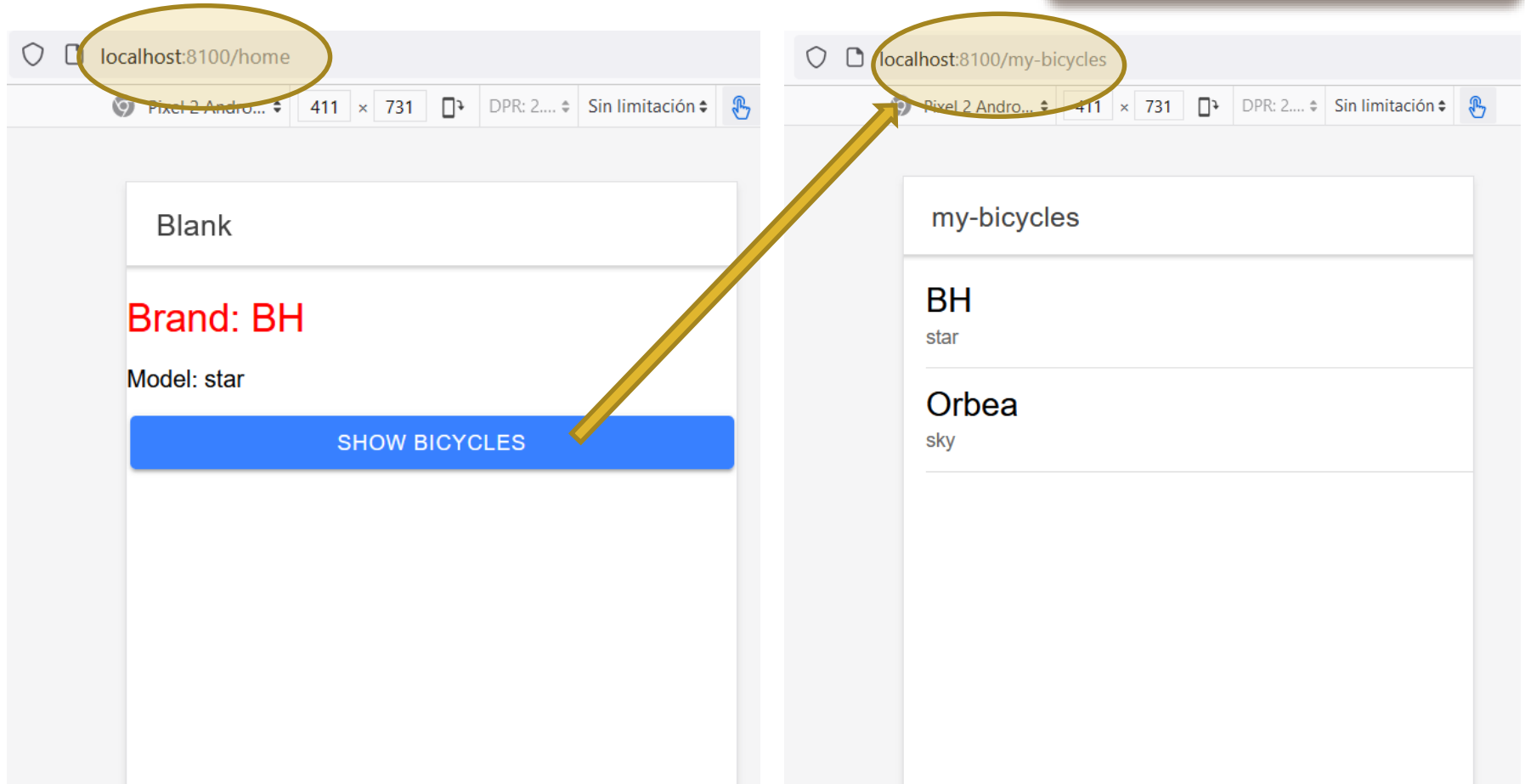
Creando un enlace

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

app-routing.module.ts M

> 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)
$
```

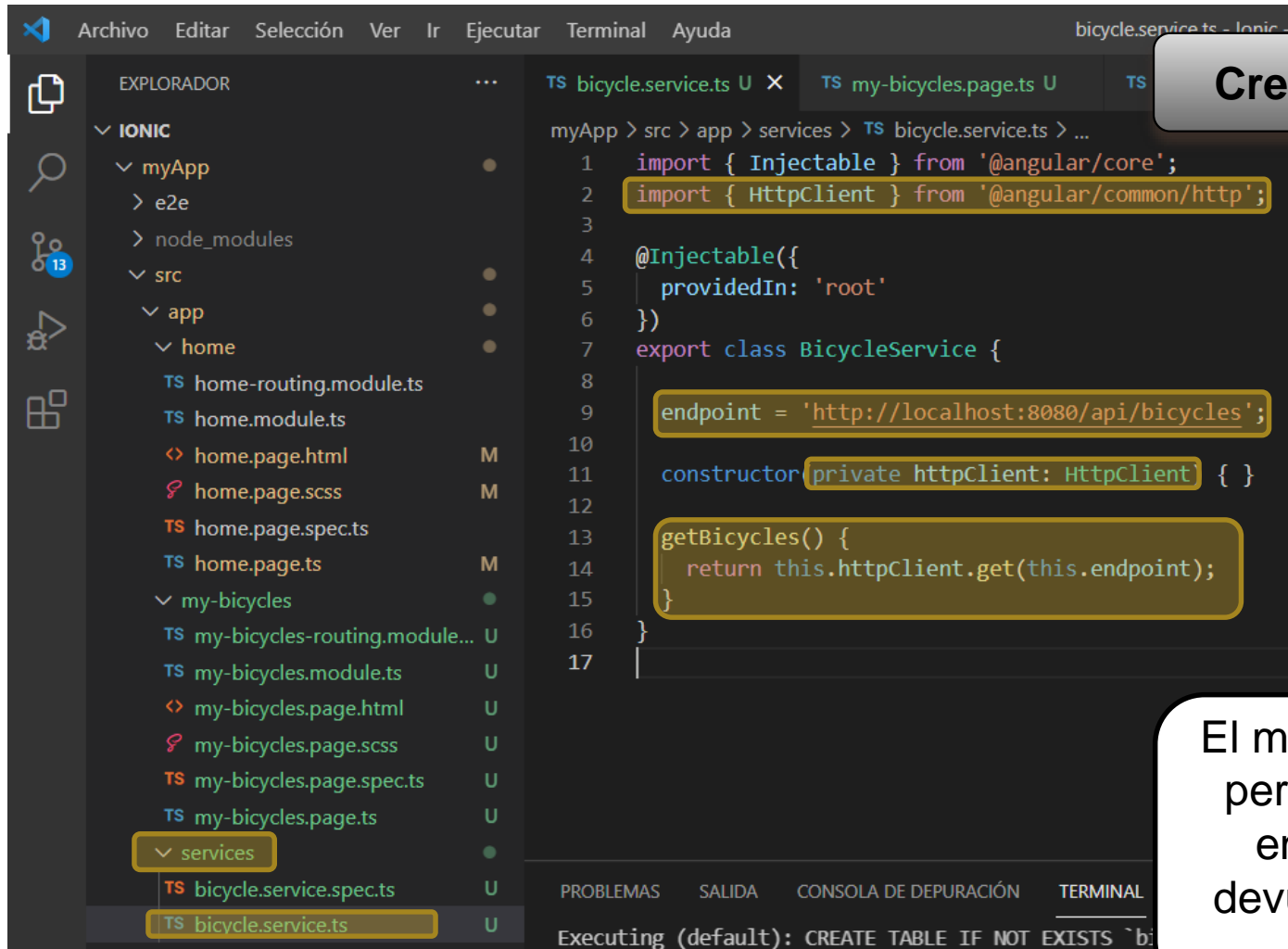
Creamos un servicio

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 Editor panel on the right. 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...
 - 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 Editor panel shows the code for `bicycle.service.ts`:

```
myApp > src > app > services > TS 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  }
```

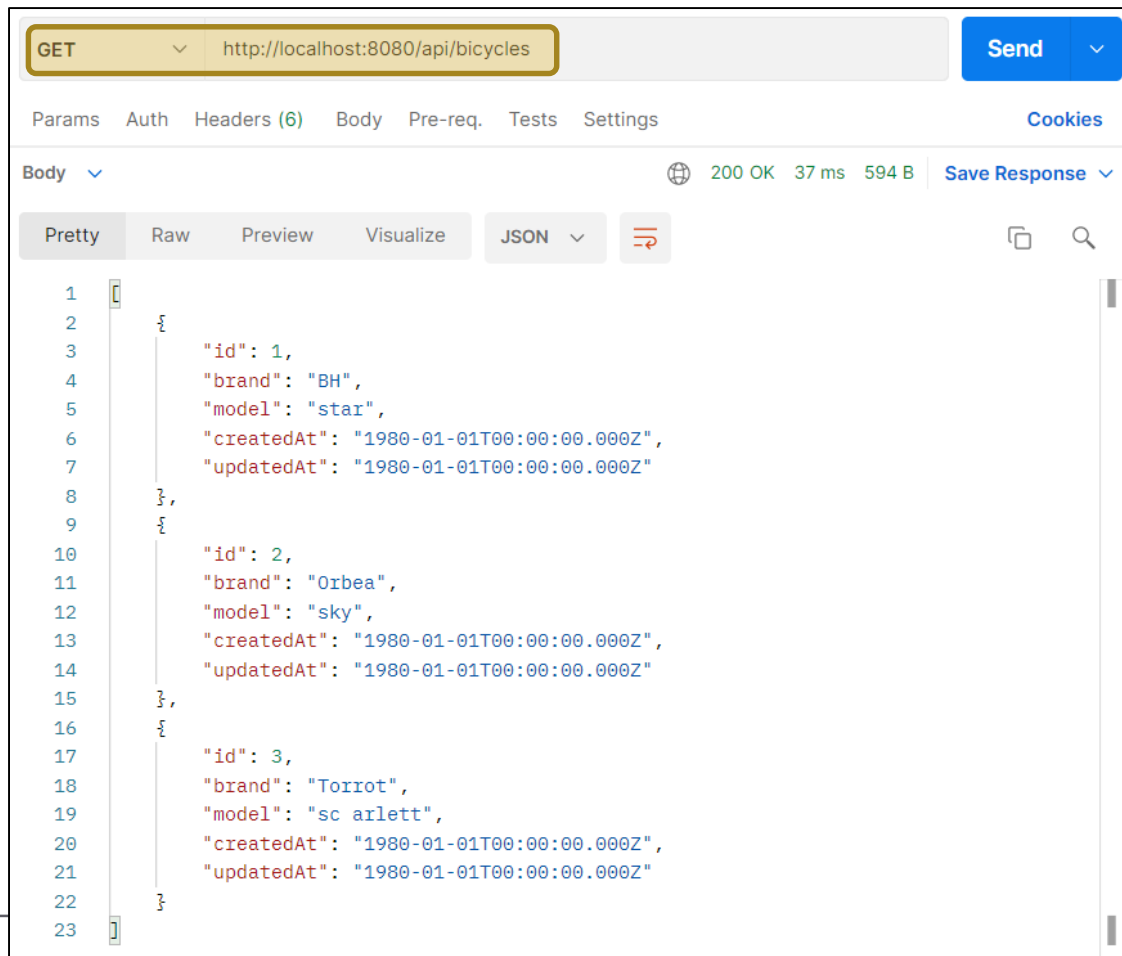
Transformamos nuestra anterior versión en la que usábamos datos hardcoded en **my-bicycles.page.ts** para obtener ahora los datos del servicio

Creando una App con Ionic...

Creando un servicio para consumir una API

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

Arranca la API

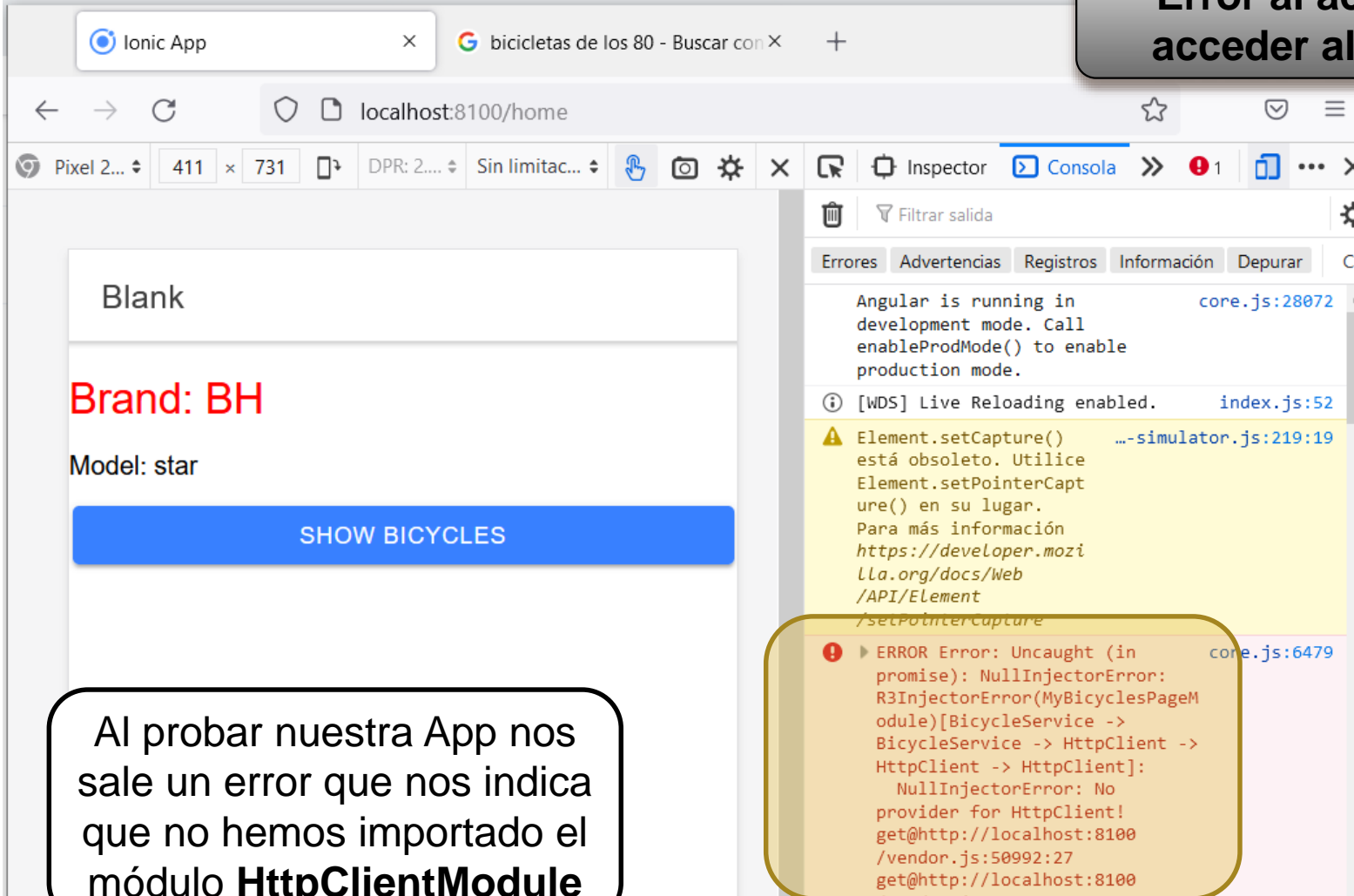


Arranca la API
que habíamos
creado en una
práctica anterior
y asegúrate que
tiene datos que
mostrar usando
POSTMAN

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.modules.ts**
es donde debemos
importar los módulos
que usamos. En este
caso el módulo
HttpClientModule

```
cle.service.ts U    TS my-bicycles.page.ts U    TS app.module.ts M X
b > src > app > TS app.module.ts > ...
import { NgModule } from '@angular/core';
import { BrowserModule } from '@angular/platform-browser';
import { RouteReuseStrategy } from '@angular/router';

import { IonicModule, IonicRouteStrategy } from '@ionic/angular';

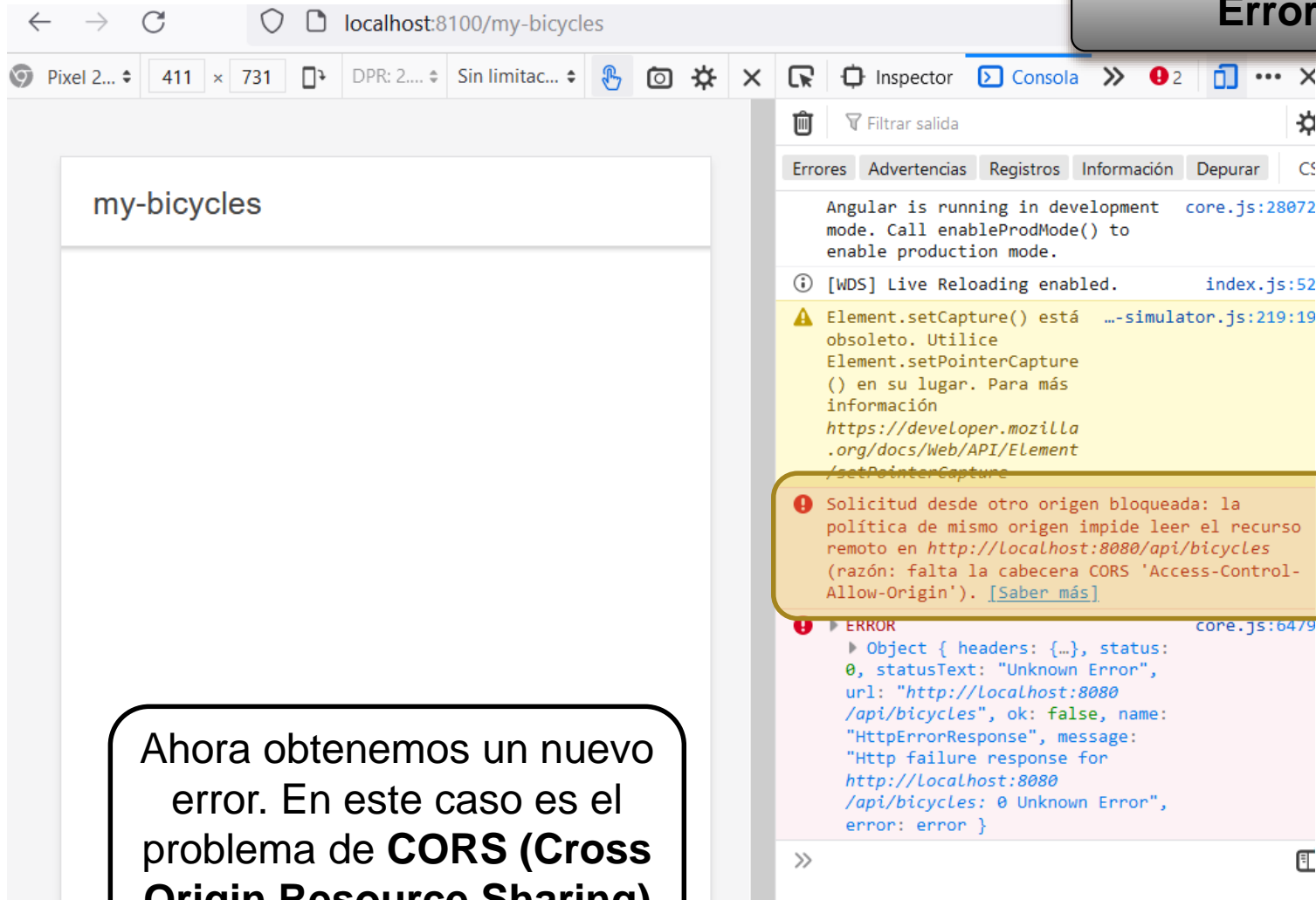
import { AppComponent } from './app.component';
import { AppRoutingModule } from './app-routing.module';
import { HttpClientModule } from '@angular/common/http';

@NgModule({
  declarations: [AppComponent],
  entryComponents: [],
  imports: [BrowserModule, IonicModule.forRoot(), AppRoutingModule, HttpClientModule],
  providers: [{ provide: RouteReuseStrategy, useClass: IonicRouteStrategy }],
  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

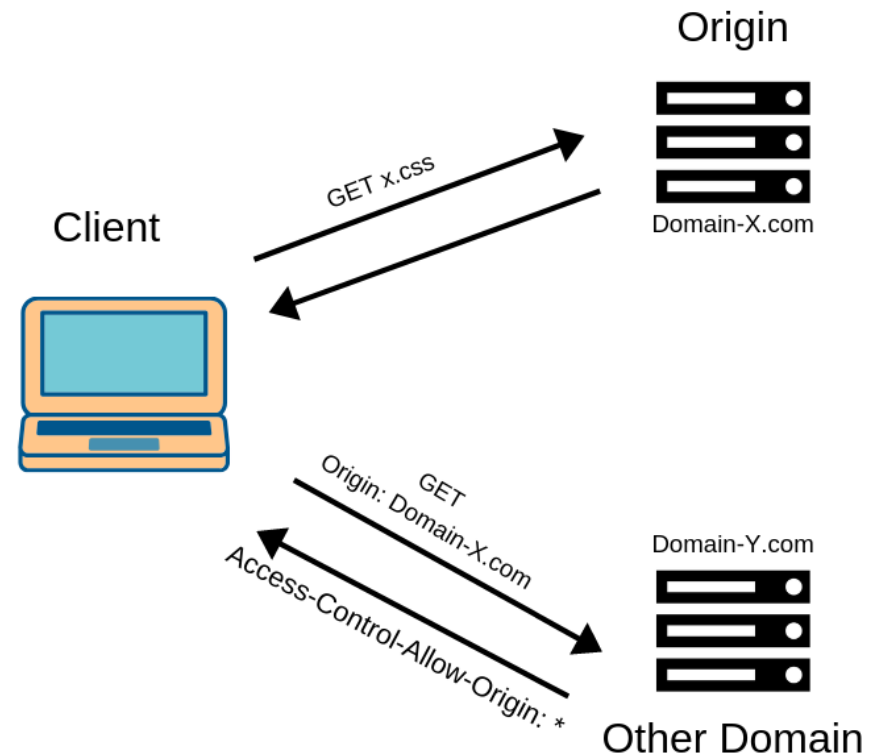
Ahora obtenemos un nuevo error. En este caso es el problema de **CORS (Cross Origin Resource Sharing)**

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...

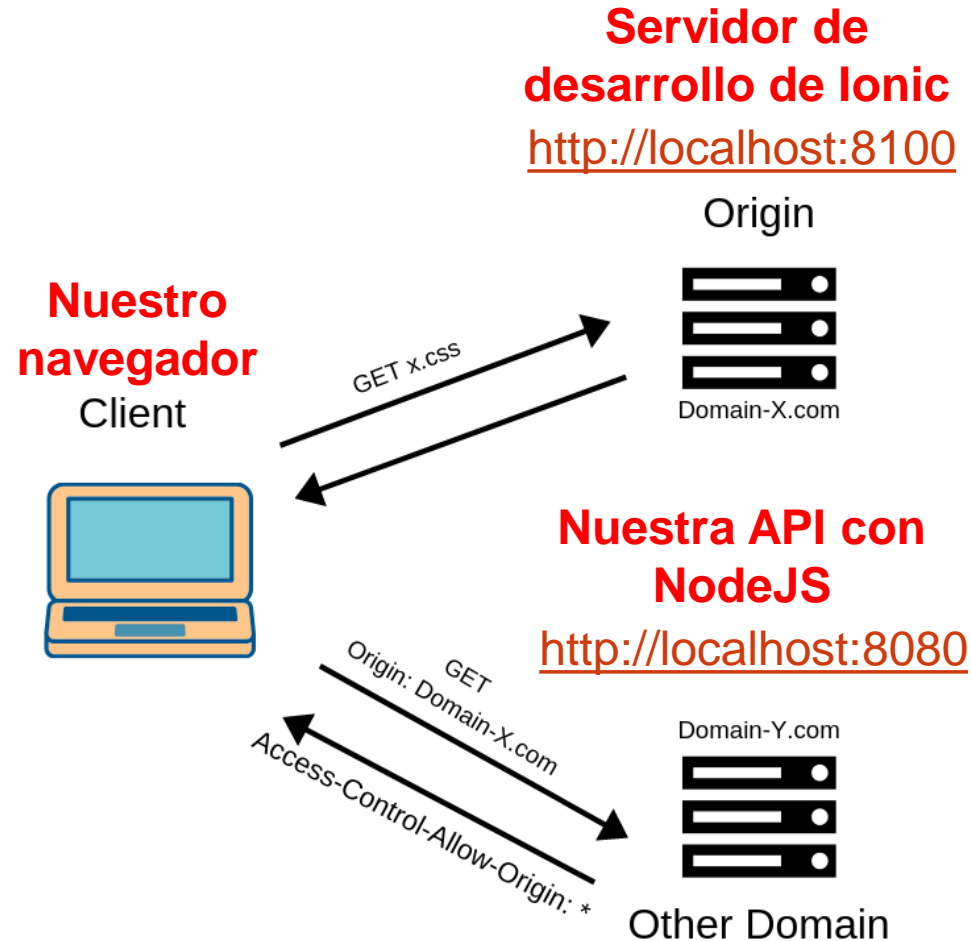
Creando un servicio para consumir una API

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

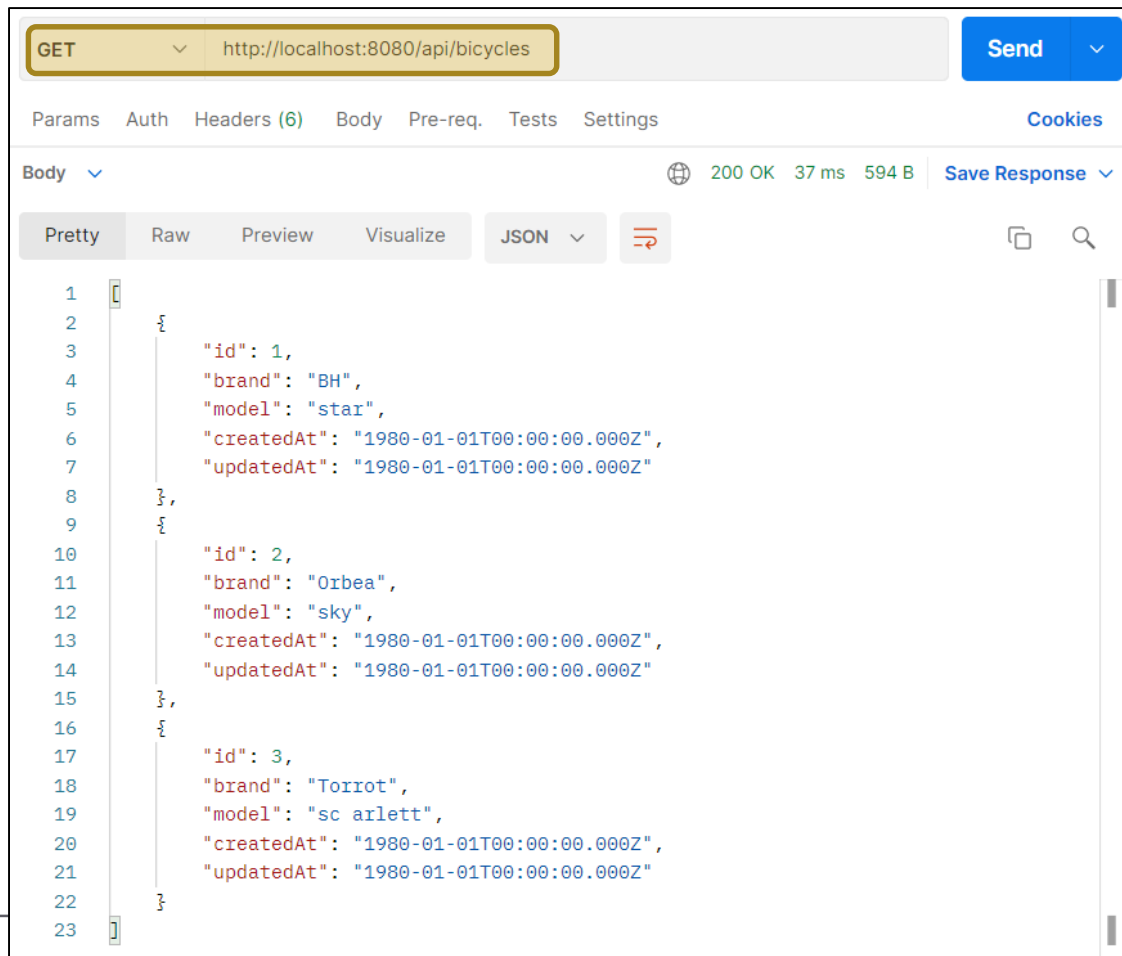
found 0 vulnerabilities

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    {  
11      "id": 2,  
12      "brand": "Orbea",  
13      "model": "sky",  
14      "createdAt": "1980-01-01T00:00:00.000Z",  
15      "updatedAt": "1980-01-01T00:00:00.000Z"  
16    },  
17    {  
18      "id": 3,  
19      "brand": "Torrot",  
20      "model": "sc arlett",  
21      "createdAt": "1980-01-01T00:00:00.000Z",  
22      "updatedAt": "1980-01-01T00:00:00.000Z"  
23    }  
24  }  
25 }
```

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 a red error message: `[WDS] Disconnected!`. Below this, it says: `Angular is running in development mode. Call enableProdMode() to enable production mode.` and `Array(3) [{...}, {...}, {...}]`. At the bottom of the console, it says `[WDS] Live Reloading enabled`.

Prueba final

¡Uhuuuu...!

¡Arriba d'ellos!

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

Sigue aprendiendo...

Termina este ejemplo para conseguir el CRUD completo:

- <https://remotestack.io/ionic-http-requests-with-httpclient-get-post-put-delete-tutorial/>

¡Cuidado con las versiones!

A fecha de la creación de este tutorial deberías trabajar con la versión 5 de Ionic.

Los tutoriales de Ionic 4 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...

- Termina el CRUD siguiendo el tutorial que te he recomendado.
-