

Aplicación híbrida para gestión de objetivos

Bea Couchoud Ruiz

IES Abastos Curso 2020-2021 Grupo 7U

Valencia, 9 de junio de 2021

Tutor individual: Eduardo González Sanz

Índice

- ▶ Introducción
- ▶ Diseño
 - ▶ Tecnologías y herramientas
 - ▶ Requisitos
 - ▶ Gestión del proyecto
 - ▶ Diagramas
 - ▶ Guía de estilos
- ▶ Desarrollo del frontend
 - ▶ Componentes
 - ▶ Servicios
 - ▶ Guards
 - ▶ Páginas y enrutamiento
 - ▶ Formularios y seguridad
- ▶ Desarrollo del backend
 - ▶ Seguridad
 - ▶ Middlewares
 - ▶ Pruebas
- ▶ Conclusiones
 - ▶ Posibles Mejoras
 - ▶ Dificultades
 - ▶ Logros
 - ▶ Resultados
 - ▶ Muestra del resultado final

Introducción

Idea inicial, contexto y objetivos

Contexto

Idea inicial

Desarrollar una **aplicación** que permita de forma **ágil e intuitiva** tomar nota de casi cualquier reto u objetivo que queramos iniciar o mantener de la manera mas personalizada posible, para poder **llevar un control de las mejoras realizadas y crear mayor adherencia a los nuevos hábitos** que se deseen.

Objetivos

El **objetivo principal** del proyecto es crear una aplicación multiplataforma, que permita registrar y visualizar los avances realizados de forma rápida.

También sería interesante poder compartir tus retos y logros con amigos y familiares.

Como **objetivo secundario** y personal, se ha buscado realizar esta aplicación híbrida empleando tecnologías y frameworks que no se han visto a lo largo del curso en combinación con otras que si que se han utilizado.

Diseño

Análisis, planificación y guía de estilos

Tecnologías y herramientas



“Write once, run everywhere”

Requisitos

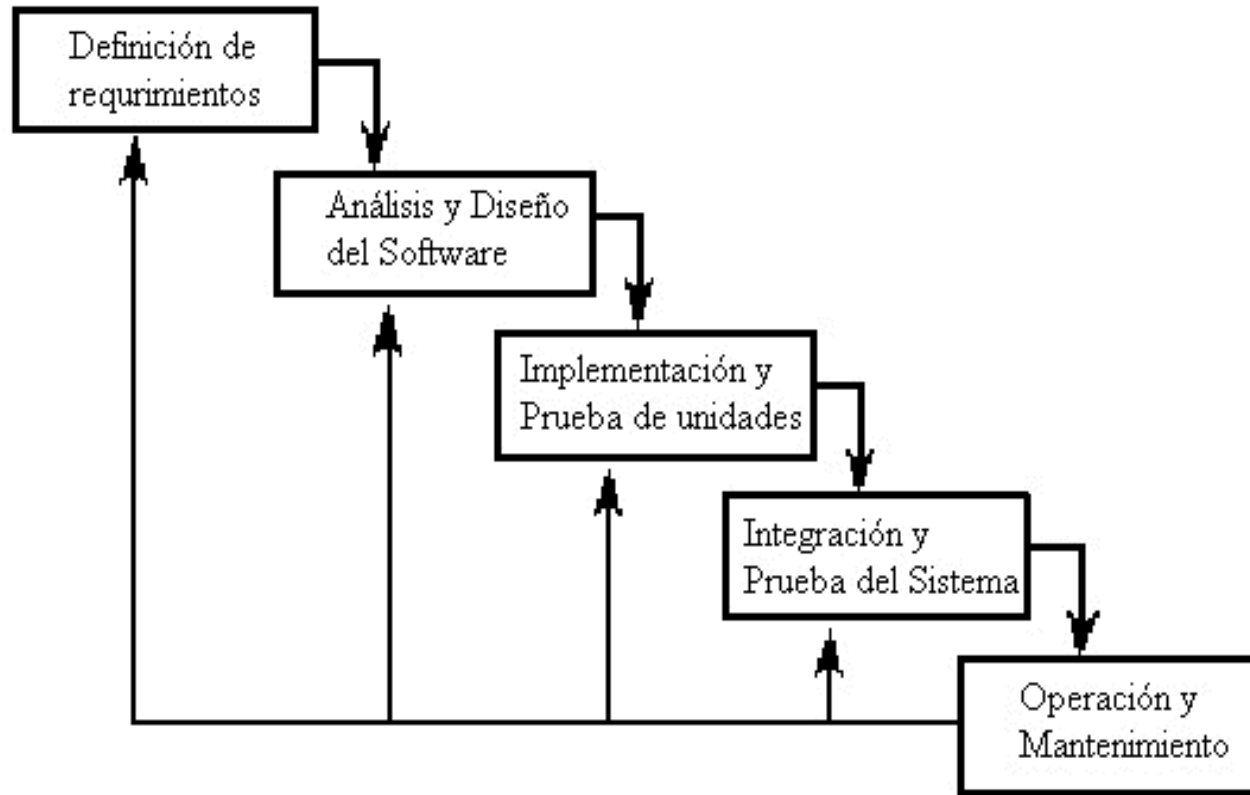
Un **requisito funcional** define una **función del sistema de software** o sus componentes. Los requisitos de **comportamiento** para cada requisito funcional se muestran en los casos de uso. Son complementados por los **requisitos no funcionales**, que se enfocan en el diseño o la implementación.

REQUISITOS FUNCIONALES

- ▶ CRUD usuario
- ▶ CRUD ítems
- ▶ CRUD amigos
- ▶ Consulta de información
- ▶ Inicio y cierre de sesión

REQUISITOS NO FUNCIONALES

- ▶ Tiempo de desarrollo
- ▶ Seguridad
- ▶ Diseño

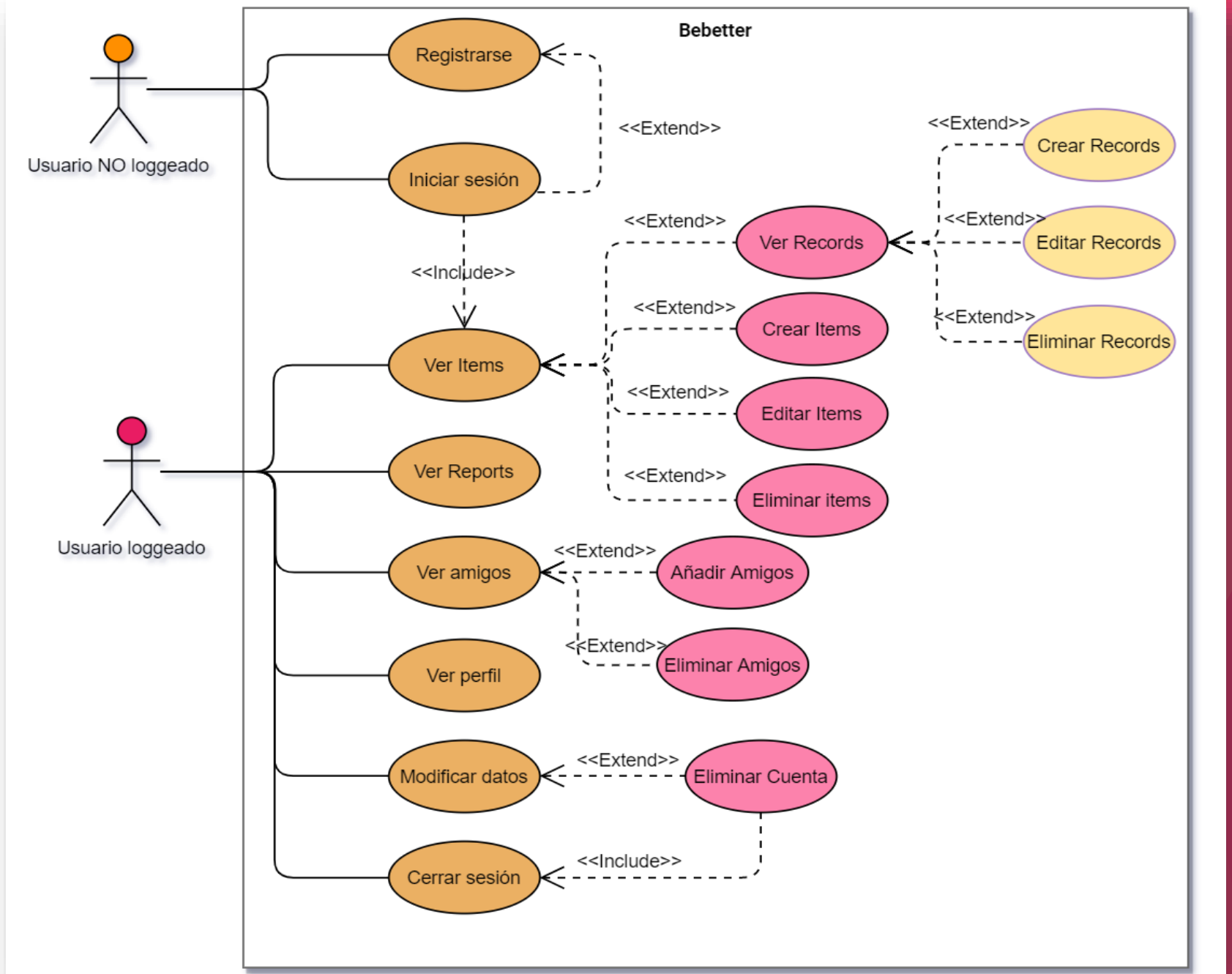


Gestión del proyecto

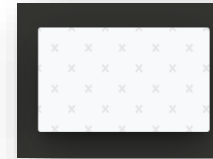
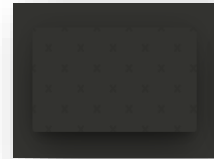
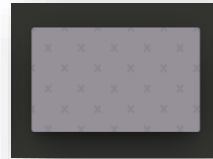
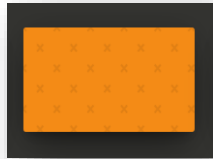
El paradigma elegido es el **desarrollo en cascada** con retroalimentación. Dentro del bloque de Codificación, se ha determinado usar la metodología de **desarrollo SCRUM** y se han determinado sprints semanales.

Diagrama de casos de uso

Facilita la comprensión del sistema y expresa la intención con la que el usuario interactuará con la aplicación. Complementándose con los requisitos anteriormente descritos, establece una firme base a la hora de comenzar con la fase de diseño.



Guía de estilos



Regular 400

Asap es una fuente sans serif contemporánea con esquinas sutilmente redondeadas.

Regular 400 italic

Asap es una fuente sans serif contemporánea con esquinas sutilmente redondeadas.

Medium 500

Asap es una fuente sans serif contemporánea con esquinas sutilmente redondeadas.

Medium 500 italic

Asap es una fuente sans serif contemporánea con esquinas sutilmente redondeadas.

Semi-bold 600

Asap es una fuente sans serif contemporánea con esquinas sutilmente redondeadas.

Semi-bold 600 italic

Asap es una fuente sans serif contemporánea con esquinas sutilmente redondeadas.

Bold 700

Asap es una fuente sans serif contemporánea con esquinas sutilmente redondeadas.

Bold 700 italic

Asap es una fuente sans serif contemporánea con esquinas sutilmente redondeadas.

Light 300

Quicksand es una fuente sans serif con acabados redondeados. Utiliza formas geométricas como base.

Regular 400

Quicksand es una fuente sans serif con acabados redondeados. Utiliza formas geométricas como base.

Medium 500

Quicksand es una fuente sans serif con acabados redondeados. Utiliza formas geométricas como base.

Semi-bold 600

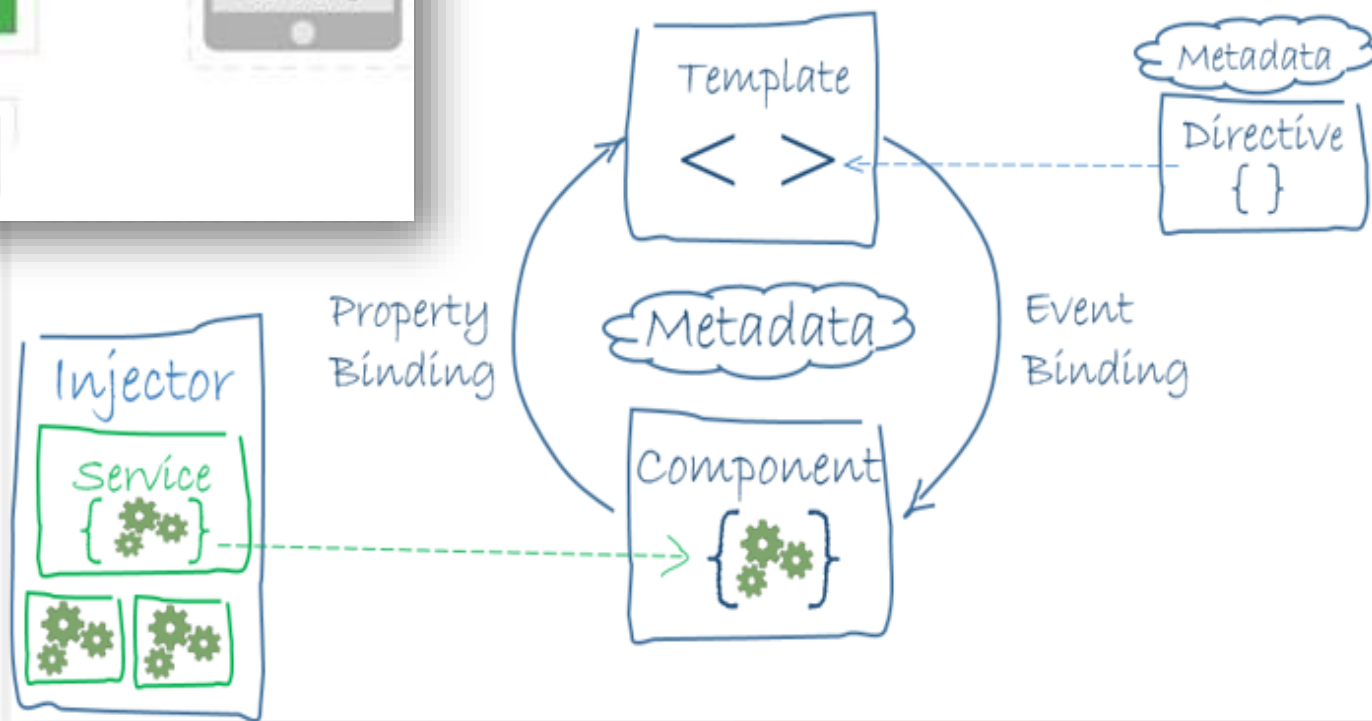
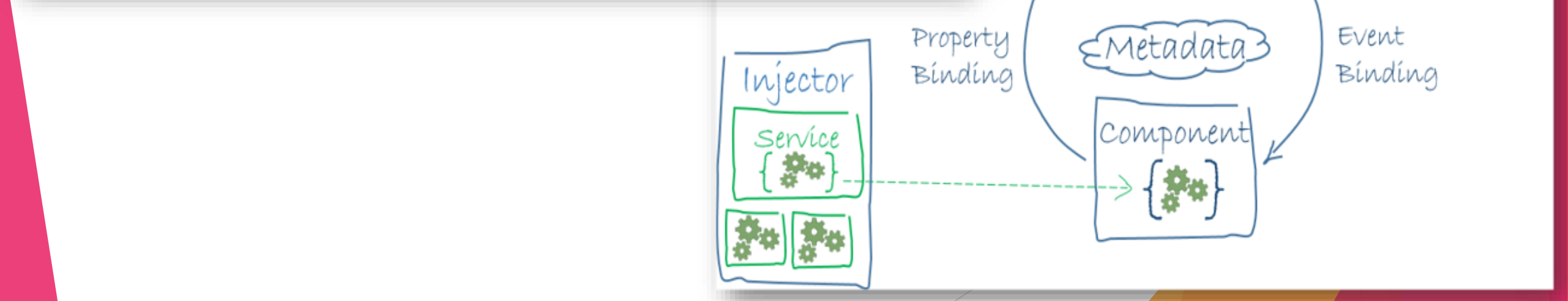
Quicksand es una fuente sans serif con acabados redondeados. Utiliza formas geométricas como base.

Bold 700

Quicksand es una fuente sans serif con acabados redondeados. Utiliza formas geométricas como base.

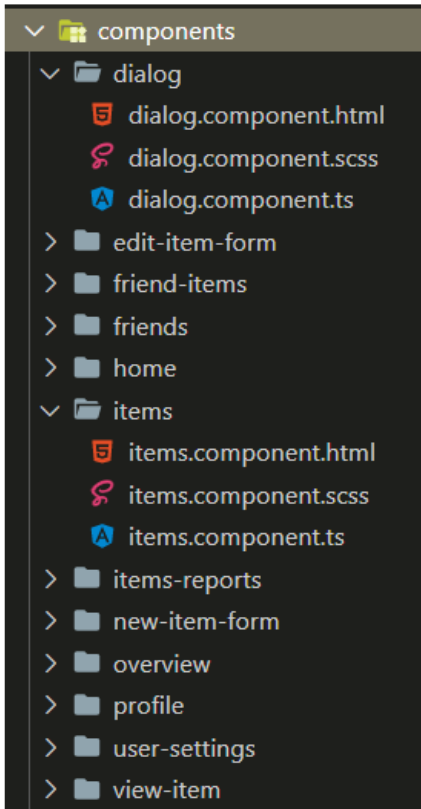
Desarrollo del Frontend

Angular e Ionic



Componentes

Un componente controla un espacio de la pantalla, que se denomina vista.



- ▶ Cada componente contiene una combinación de un archivo **.html** con un **.ts** y algunas veces un **.css**.
- ▶ Un componente es un **elemento con características propias** tanto de comportamiento como de apariencia que se puede mostrar.
- ▶ El **template** es el encargado de definir la vista del componente.
- ▶ Los archivos son más cortos y sencillos de leer, **permite mantener y escalar el código** de manera más sencilla.

Servicios

Un servicio es un **proveedor de datos**, que mantiene la lógica de acceso a los mismos y la operativa relacionada con la capa de negocio.

```
37 private getItem(): void {  
38     this.itemService.getFullItem (this.user.username)  
39     .subscribe(  
40         (item) => {  
41             this.item = item;  
42             this.itemService.setActiveItem(item);  
43         },  
44         (error) => console.log(error)  
45     );  
46 }
```

Llamada a un método de un servicio desde un componente

- ▶ Son **consumidos por los componentes**, que delegan en ellos la responsabilidad de acceder a la información y la realización de operaciones con los datos.
- ▶ La mayoría de métodos incluidos **retornan un objeto Observable**.
- ▶ Mediante el Observable es posible suscribirse a eventos que nos permiten hacer cosas cuando cambia lo que se esté observando, es una de las mejores formas de optimizar una aplicación, ya que **aumenta su rendimiento**.

Guards

Los Guards son, de alguna manera, middlewares que se ejecutan antes de cargar una ruta y determinan si se puede cargar dicha ruta o no.

- ▶ Se ejecutan de manera intermedia antes de determinadas acciones, en función de si devuelven true o false se carga una ruta u otra.
- ▶ Una de las ventajas de utilizarlos, es que al no cargar la ruta evitamos que los usuarios vean una interfaz a la que no tienen acceso.
- ▶ Estos componentes permiten estructurar el código de la aplicación de una manera más organizada y donde la lógica de la ruta está en la ruta en sí y la lógica de permisos y acceso a recursos se encuentra aislada en los guards.

Páginas y enrutamiento

Una de las responsabilidades del front es la de procesar las rutas y determinar cuál será la vista que se deba mostrar en cada dirección.

- ▶ Se aplica **Lazy Loading** para mejorar el rendimiento de la app.

```
app-routing.module.ts M X
better > better > src > app > app-routing.module.ts > ...
1 import { NgModule } from '@angular/core';
2 import { PreloadAllModules, RouterModule, Routes } from '@angular/router';
3 import { SectionPage } from '../pages/section/section.page';
4
5 const routes: Routes = [
6   {
7     path: '',
8     redirectTo: 'signin',
9     pathMatch: 'full'
10  },
11  {
12    path: 'section',
13    component: SectionPage,
14    loadChildren: () => import('../pages/section/section.module').then( m => m.SectionPageModule)
15  },
16  {
17    path: 'signin',
18    loadChildren: () => import('../pages/signin/signin.module').then( m => m.SigninPageModule)
19  },
20  {
21    path: 'signup',
22    loadChildren: () => import('../pages/signup/signup.module').then( m => m.SignupPageModule)
23  }
24 ];
```

```
20 </ion-content>
21 <ion-footer>
22   <ion-item routerDirection="root" [routerLink]="/signin" lines="none" detail="false" >
23     <ion-icon name="log-out" class="ion-margin-end"></ion-icon>
24     <ion-label class="ion-margin-start">Log Out</ion-label>
25   </ion-item>
26 </ion-footer>
27 </ion-menu>
28 </ng-container>
29 <ion-router-outlet id="main-content"></ion-router-outlet>
30 </ion-split-pane>
31 </ion-app>
```

```
section-routing.module.ts M X
better > better > src > app > pages > section > section-routing.module.ts > ...
8 import { UserSettingsComponent } from 'src/app/pages/user-settings/user-settings.component';
9 import { ViewItemComponent } from 'src/app/pages/view-item/view-item.component';
10 import { HomeComponent } from '../home/home.component';
11
12 const routes: Routes = [
13   {
14     path: '',
15     redirectTo: 'Home',
16     pathMatch: 'full',
17   },
18   {
19     path: 'Home',
20     children: [
21       {
22         path: '',
23         data: { title: 'Home' },
24         component: HomeComponent,
25       },
26       {
27         path: 'NewItem',
28         data: { title: 'New Item' },
29         component: NewItemFormComponent,
30       },
31       {
32         path: 'EditItem/:id',
33         data: { title: 'Edit Item' },
34         component: EditItemFormComponent,
35       },
36     ],
37   },
38   {
39     path: 'Reports',
40     component: ReportsComponent,
41   },
42   {
43     path: 'Friends',
44     component: FriendsComponent,
45   },
46 ];
```


Formularios y seguridad

Se han usado formularios reactivos, que realizan un seguimiento del valor, el estado de cambio y la validez de los datos.




```
23 public formUserData: FormGroup;
24 public formPassword: FormGroup;
25
26 constructor(private fb: FormBuilder) { }
27
28 ngOnInit() {
29     this.initFormUserData();
30     this.initFormPassword();
31 }
32
33 private initFormUserData(): void {
34     this.formUserData = this.fb.group({
35         name: [this.user.name, [Validators.required]],
36         email: [this.user.email, [Validators.email, Validators.required]]
37     });
38 }
39
40 private initFormPassword(): void {
41     this.formPassword = this.fb.group({
42         oldPassword: [null, [Validators.minLength(8), Validators.required]],
43         newPassword: [null, [Validators.minLength(8), Validators.required]]
44     });
45 }
46
```

Desarrollo del backend

API REST y MongoDB

Base de datos

- ▶ MongoDB **guarda estructuras de datos BSON** (una especificación similar a JSON) con un esquema dinámico, haciendo que la integración de los datos en ciertas aplicaciones sea más fácil y rápida.
- ▶ Las propias consultas son también JSON, por lo que se programan fácilmente.
- ▶ Ofrece un **modelo de datos flexible**, es decir, los campos pueden variar de un documento a otro, los documentos se describen por sí mismos.
- ▶ Si se necesita, se puede agregar un nuevo campo a un documento sin afectar a todos los demás documentos del sistema, sin actualizar un catálogo central del sistema y sin desconectar el sistema.

Collections						
CREATE COLLECTION						
Collection Name	Documents	Avg. Document Size	Total Document Size	Num. Indexes	Total Index Size	Properties
friends	6	102.7 B	616.0 B	2	72.0 KB	
items	6	321.3 B	1.9 KB	2	72.0 KB	
users	6	182.7 B	1.1 KB	3	108.0 KB	

Seguridad

Se han implementado unas cuantas prácticas para garantizar un mínimo de confianza:

- ▶ Se ha filtrado y saneado la entrada datos, usando mongo-sanitize, un módulo que evita que cualquier palabra que comience con '\$' sea introducida en una query de MongoDB
- ▶ Se ha evitado el uso de \$where ya que es uno de los operadores menos seguros de usar, debido a que, aunque se filtren datos es difícil evitar que se introduzca un 0 o un true como valor en determinadas queries
- ▶ Se han encriptado las contraseñas para mayor seguridad con la ayuda de bcrypt, una librería de encriptación que, básicamente, es una función de hashing de contraseñas, basado en cifrado Blowfish.
- ▶ Se ha implementado una autenticación mediante token. La firma de un token JWT se construye de tal forma que vamos a poder verificar que el remitente es quien dice ser (dificultando un ataque MITM)
- ▶ Se ha limitado el tamaño del cuerpo de las peticiones. Al igual que se controla el tipo de los datos de entrada, también se ha limitado su tamaño máximo para evitar ataques DoS.

Middlewares

Los middleware son funciones que tienen acceso al objeto de solicitud (req), al objeto de respuesta (res) y a la siguiente función de middleware en el ciclo de solicitud/respuestas de la aplicación.

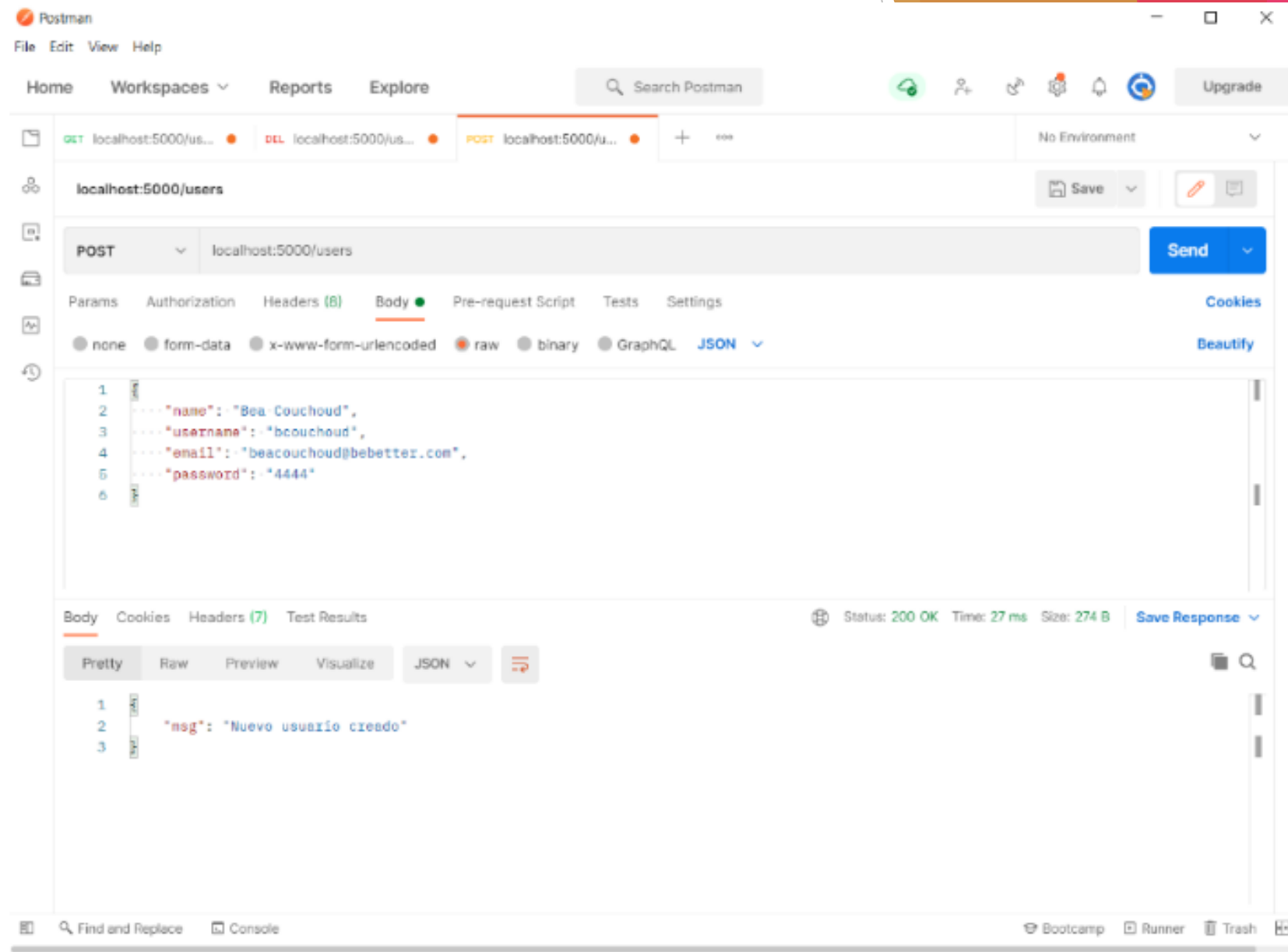
- ▶ Pueden utilizarse a nivel de aplicación, para redireccionar, para el control y manejo de errores, etc.
- ▶ Para el funcionamiento de la app se han desarrollado dos middlewares uno para verificar que los tokens de las distintas peticiones eran válidos y otra para verificar que el usuario y el email no estaban en uso antes de insertar un nuevo usuario en la base de datos
- ▶ Se han usado varios middlewares de terceros: mongo-sanitize, bcrypt, jsonwebtoken, bodyparser, mongoose...



POSTMAN

Pruebas

Para elaborar test que permitan validar el comportamiento de la API se ha utilizado **Postman**. Postman es una herramienta que permite realizar peticiones a APIs y generar colecciones de peticiones que nos permitan probarlas de manera rápida y sencilla.



Conclusiones

Posibles mejoras, dificultades, logros y resultados



POSIBLES MEJORAS

- ▶ Mejorar el responsive
- ▶ Traducir a más idiomas
- ▶ Subida de aplicación a tiendas online



DIFICULTADES

- ▶ Gestión de tiempo
- ▶ Variedad de pantallas
- ▶ Uso de tecnologías no vistas a lo largo del curso
- ▶ Problemas de compatibilidad



LOGROS

- ▶ Diseño muy parecido al mockup inicialmente planteado
- ▶ Puesta en práctica de conocimientos adquiridos a lo largo de todo el grado
- ▶ Aprendizaje de nuevas tecnologías de forma autodidacta



RESULTADOS

- ▶ App híbrida
- ▶ Funcional y segura
- ▶ Diseño intuitivo

DEMO



BEBETTER

Sign In

Hi there! Nice to see you again.

Email

bcouchoud@gmail.com

Password

.....

SIGN IN

or use one of your Google account



SIGN IN WITH GOOGLE

Forgot password?

Sign Up