

TEMA1

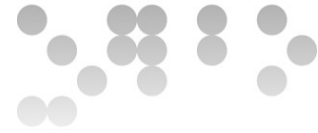
Programación reactiva: Formularios reactivos

Desarrollo front-end avanzado

Máster Universitario en Desarrollo de sitios y
aplicaciones web

Semestre 20192

Estudios de Informática, Multimedia y Telecomunicación



1.1. Programación Reactiva

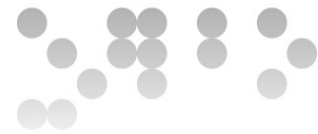
La programación reactiva es un **paradigma de programación** que consta de dos fundamentos: 1) Manejo de flujos de datos asíncronos; y 2) Uso eficiente de recursos.

- 1. Manejo de flujos de datos asíncronos:** La programación reactiva está basada en un patrón de diseño llamado **Observador**. Este patrón consiste en que cuando se produce un cambio en el estado de un objeto, otros objetos son notificados y actualizados acorde a dicho cambio. Es decir, se dispone de un conjunto de objetos observados y otros que están suscritos a los cambios (notificaciones) de estos objetos observados. Por lo tanto, y más importante, los eventos se realizan de forma asíncrona. De esta manera los observadores no están constantemente preguntando al objeto observado si ha cambiado su estado.
- 2. Uso eficiente de recursos:** Los sistemas reactivos permiten que los clientes (i.e. los objetos observadores que esperan el cambio sobre los observados) realicen otras tareas hasta que sean notificados del cambio, en lugar de hacer *polling*, es decir, en lugar de estar permanentemente comprobando si se ha producido un cambio.

Un claro ejemplo de uso de la programación reactiva son los diferentes eventos de **click** sobre una interfaz gráfica, ya que se genera un flujo de eventos asíncronos (puede hacerse click en un segundo y otro al cabo de 3 segundos o cualquier intervalo de tiempo), los cuales pueden ser observados y se puede reaccionar en consecuencia a lo que suceda en dicho flujo (cada vez que se dispare uno, o cuando se acumulen unos cuantos eventos en un determinado intervalo de tiempo, etc.).

La Programación Reactiva suele ir acompañada de un conjunto de operadores que permite la manipulación completa de los flujos de datos, y ahí es donde se puede aprovechar el verdadero potencial de este paradigma de programación. Este conjunto de operadores en nuestro contexto será proporcionado por la biblioteca Rx (ReactiveX), concretamente la versión para JavaScript (RxJS).

En el siguiente tema profundizaremos sobre la Programación Reactiva utilizando RxJS. En este tema vamos a introducir los formularios reactivos existentes en el propio framework Angular y así sustituir los clásicos formularios dirigidos por el modelo.

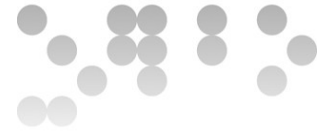


1.2. Formularios reactivos Vs Dirigidos por el modelo

Para introducir la Programación Reactiva se puede comenzar utilizando los formularios reactivos proporcionados por Angular. En principio, Angular permite desarrollar formularios utilizando los dos paradigmas de programación, pero las principales características de uno y otro son las siguientes (<https://stackoverflow.com/a/41685151/3890755>):

- **Formularios dirigidos por el modelo (Template Driven Forms)**
 - o Fáciles de usar.
 - o Se ajustan adecuadamente en escenarios simples, pero no son útiles para escenarios complejos.
 - o Uso de formularios similar al utilizado en AngularJS (predecesor).
 - o Manejo automático del formulario y los datos por el propio framework de Angular.
 - o La creación de tests unitarios en este paradigma es complejo.
- **Formularios Reactivos (Reactive Forms)**
 - o Más flexibles.
 - o Se ajustan adecuadamente en escenarios complejos.
 - o No existe *data-binding*, sino que se hace uso de un modelo de datos inmutable.
 - o Más código en el componente y menos en el lenguaje de marcas HTML.
 - o Se pueden realizar transformaciones reactivas, tales como:
 - Manejo de eventos basados en intervalos de tiempo.
 - Manejo de eventos cuando los componentes son distintos hasta que se produzca un cambio.
 - Añadir elementos dinámicamente.
 - o Fáciles de testear.

Para construir formularios reactivos, tenemos que tener en cuenta que es necesario incluir un nuevo módulo denominado **ReactiveFormsModule**, el cual nos permitirá acceder a los componentes, directivas y *providers* reactivos tales como **FormBuilder**, **FormGroup** y **FormControl**.



En la guía de Angular2 de Rangle.io (<https://angular-2-training-book.rangle.io/handout/forms/reactive-forms/reactive-forms.html>) se muestra el paso a paso de un formulario reactivo de *login*.

Es importante resaltar que el servicio (*provider*) **FormBuilder** debe ser inyectado para poder construir los formularios haciendo uso de **FormGroup** y **FormControl**.

Para tener una completa guía de los formularios reactivos puedes seguir el tutorial mostrado en la documentación oficial (<https://angular.io/guide/reactive-forms>) y, finalmente, si quieres ver ejemplos en los que se comparan los formularios dirigidos por modelo con los reactivos puedes leer el siguiente tutorial (<https://blog.angular-university.io/introduction-to-angular-2-forms-template-driven-vs-model-driven/>).

Observa que los *templates* (i.e. ficheros .html) quedan bastante limpios, puesto que solamente se definen las directivas **FormControl** y **FormGroup**.

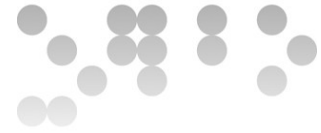
1.3. Validaciones de los formularios

Angular Reactive Forms (<https://angular.io/guide/form-validation#reactive-form-validation>) incorpora una serie de validaciones que se definen a la hora de construir el formulario (mira <https://angular.io/api/forms/Validators>).

De esta manera es bastante simple definir en el control de un formulario, su valor inicial y un *array* con las diferentes validaciones que permitirán controlar el cambio de estado de dicho control.

En el siguiente código se muestra la definición de un formulario reactivo denominado *heroForm*. Este formulario se define usando la clase **FormGroup**, en la cual el constructor recibe un objeto en el que la *key* será el nombre del campo del formulario y el valor será un control (instancia de la clase **FormControl**). Observa que el constructor de **FormControl** recibe dos parámetros:

1. El valor inicial del control en el formulario, que es igual al valor del atributo *name* del objeto *hero*.
2. *Array* con las validaciones que debe superar dicho control. En el ejemplo se usan validaciones comunes que incorpora Angular como son **Validators.required** o **Validators.minLength**. Además, en el ejemplo se muestra un validador propio denominado **forbiddenNameValidator** el cual es un validador construido específicamente para nuestro caso particular.



```
this.heroForm = new FormGroup({
  'name': new FormControl(this.hero.name, [
    Validators.required,
    Validators.minLength(4),
    forbiddenNameValidator(/bob/i) // <-- Here's how //
you pass in the custom validator.
  ]),
  'alterEgo': new FormControl(this.hero.alterEgo),
  'power': new FormControl(this.hero.power,
    Validators.required)
});
```

El control del estado del formulario tanto para dar *feedback* al usuario como para realizar diferentes validaciones viene definido por los siguientes estados:

```
/* field value is valid */
.ng-valid {}

/* field value is invalid */
.ng-invalid {}

/* field has not been clicked in, tapped on, or tabbed
over */
.ng-untouched {}

/* field has been previously entered */
.ng-touched {}

/* field value is unchanged from the default value */
.ng-pristine {}

/* field value has been modified from the default */
.ng-dirty {}
```

Como puedes observar, existen los siguientes pares:

- valid / invalid
- untouched / touched
- pristine / dirty

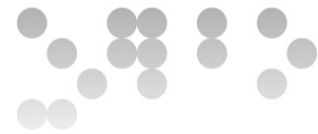


Además, si lees la documentación oficial sobre la clase **FormControl** (<https://angular.io/api/forms/FormControl>), puedes ver que existen los atributos `valid`, `invalid`, `pristine`, `dirty`, `touched`, `untouched` para poder tener control programáticamente sobre el control. Igualmente, puedes leer la documentación del **FormGroup** (<https://angular.io/api/forms/FormGroup>), el cual permite saber si el grupo completamente es válido (cumple con las validaciones) y así permitir habilitar un botón de envío del formulario.

1.4. Validaciones propias

Aunque la clase **Validators** permite disponer de un conjunto de validadores de base, a veces requerimos construir nuestros propios validadores (p.e. dos campos coinciden para el uso de *passwords*, o cualquier otra idea). Para ello necesitamos recurrir a crear nuestros propios validadores:

- <https://angular.io/guide/form-validation#custom-validators>
- https://angular-2-training-book.rangle.io/handout/forms/reactive-forms/reactive-forms_custom_validation.html
- <https://blog.thoughttram.io/angular/2016/03/14/custom-validators-in-angular-2.html>



PRÁCTICA

Presentación

Esta Práctica se centra en ampliar los conocimientos adquiridos en las asignaturas de los semestres anteriores y ampliarlo dando a conocer a los alumnos los formularios reactivos y el comienzo de la programación reactiva.

Objetivos

Los objetivos que se desean lograr con el desarrollo de esta Práctica son:

- **Sentar las bases de la aplicación global que se va a desarrollar**, haciendo uso de todo lo aprendido a lo largo del máster.
- Recordar los conceptos fundamentales de **componentes, router, servicios, módulos** de una aplicación *single-page application* (SPA) haciendo uso del *framework* Angular.
- Crear **formularios reactivos complejos** y de manera reutilizable.

Formato y fecha de entrega

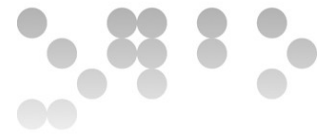
Se deberá entregar todo el proyecto comprimido en formato .zip sin incluir el directorio “node_modules”. La fecha de entrega será la mostrada en el aula de la asignatura

La estructura de directorios que se utilizará es la proporcionada en el proyecto UOCJob que se encuentra en el aula virtual. En esta estructura de directorios se han separado dos directorios principalmente:

- 1. shared:** Aquí se encontrarán los servicios, modelos, tuberías, componentes comunes, así como el estado de la aplicación (se profundizará en el siguiente tema acerca de esto).
- 2. views:** Aquí se encontrarán las vistas con las que interactuara el usuario.

Enunciado

A lo largo de toda la asignatura se va a desarrollar un proyecto completo en el que se hará uso de todo lo aprendido a lo largo del máster e introduciendo patrones de diseño *front-end* avanzados, así como técnicas avanzadas de maquetación o de despliegue haciendo uso del *framework* Angular.



La elección de Angular ha sido debido a que es el *framework* más completo conceptualmente, el cual permitirá al alumno el día de mañana poder moverse entre los diferentes *frameworks* que existen en el mercado sin necesidad de reaprender conceptos, sino solo adaptarse a la sintaxis y los problemas de configuración propios de cada uno de los *frameworks*.

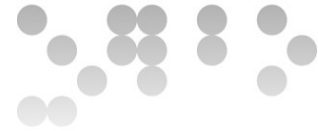
IMPORTANTE: Un enunciado puede ser ambiguo, puede tener varias interpretaciones, pero en la vida real no hay enunciados, hay clientes. Por lo tanto, en caso de duda, id al foro y preguntad, puesto que el consultor va a actuar como puro cliente que en caso de duda resolverá las dudas del dominio del problema.

Se va a construir una aplicación interna para un centro educativo en la que se quiere poner en contacto a **alumnos** que han concluido sus estudios con diferentes **empresas** que ofertan trabajo. Es decir, se va a construir una bolsa de empleo entre ex-alumnos y empresas del sector. Debes tener en cuenta de que habrá dos tipos de usuarios: estudiantes y empresas, todos ellos se autenticarán mediante e-mail (será el *login*) y password.

Por lo tanto, basado en proyectos pasados dispondremos de las siguientes pantallas iniciales (que desarrollarás) junto a su puntuación:

- Configurar el entorno de router entre diferentes pantallas – **1 punto**.
- Configurar el fake-service y los objetos necesarios para el correcto funcionamiento de la aplicación – **1 punto**.
- Pantalla **LOGIN** - **1 punto**.
- Pantalla **DASHBOARD** (esta pantalla no se realizará en esta primera PEC, solamente desarrollaremos el componente para poder realizar el enrutamiento hacia los demás componentes).
- Pantalla **PROFILE** (ESTUDIANTE)
 - Datos personales - **2 puntos**.
 - Formación Académica - **2 puntos**.
 - Idiomas - **1 punto**
- Pantalla **JOB OFFERS** (ESTUDIANTE) y **MY JOB OFFERS** (ESTUDIANTE) – **2 puntos**

La elaboración de las siguientes pantallas solo serán tenidas en cuentas para alcanzar una mayor puntuación (**la práctica está puntuada sobre 10 puntos, y estas preguntas suman un extra a la calificación**) y serán tomadas en cuenta en caso de que se quiera obtener una calificación mayor (Esta PEC seguirá estando puntuada sobre 10 puntos pero con la posibilidad



de obtener más puntuación para reforzar la calificación en los otros apartados):

- Pantalla **PROFILE** (ESTUDIANTE)
 - Experiencia Laboral (**0.50 puntos**)
- Pantalla **JOB OFFERS** (EMPRESA) (**0.75 puntos**)
- Pantalla **CONFIGURACIÓN** (**0.50 puntos**)

Se deberán crear componentes base para cada una de estas pantallas y poder navegar entre ellas usando el *router* de Angular. **NO hay que preocuparse ahora mismo del aspecto visual puesto que en temas posteriores se hará uso de bibliotecas propias para maquetar nuestros componentes.**

Sigue el enunciado de manera secuencial para poder avanzar en el desarrollo de esta Práctica.

1) FAKE BACK-END: Angular-in-memory-web-api

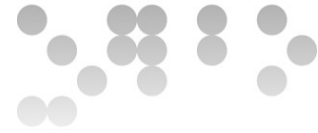
Debido a que, sería muy tedioso construir un *back-end* completo para hacer uso de nuestra aplicación, vamos a simular un *back-end* de modo que podamos desarrollar toda nuestra aplicación *front-end* mientras otra parte del equipo está desarrollando el *back-end* o simplemente para mostrarle al cliente la aplicación *front-end* funcionando previamente al desarrollo de nuestro *back-end*.

Para ello, Angular proporciona un módulo específicamente para esto que permite almacenar toda la información en memoria simulando las llamadas al *back-end* y realizando las operaciones de CRUD.

Este módulo se llama **angular-in-memory-web-api**. **Es muy importante instalarlo y configurarlo para poder seguir avanzando la práctica.** En el siguiente enlace se muestra cómo configurar dicho módulo.

<https://www.techiediaries.com/angular-inmemory-web-api/>
<https://blog.ng-classroom.com/blog/angular/Angular-HttpClient/>
<https://codeburst.io/crud-with-angular-5d8f39805c49>

Este módulo lo utilizaremos para no tener la complejidad extra del desarrollo del *back-end*. Al usar este módulo sólo nos tendremos que centrar en el *front-*



end, puesto que gracias a este módulo toda la información que estaría en el back-end (i.e. base de datos) la estaríamos simulando en la memoria. La principal ventaja de esto es que podemos ver cómo interactúa nuestra aplicación sin tener una de las partes del sistema. Por lo tanto, solamente deberíamos configurar el módulo y los diferentes archivos `.json` que actuarán como sustitutos de la base de datos del servidor.

2) Pantalla Login

Esta pantalla de *login* deberá mostrar el primer formulario reactivo que construiremos, el cual estará compuesto por los siguientes componentes:

- Logotipo de un Centro Educativo (UOC, por ejemplo).
- Campo Email
- Campo *Password*
- Botón *Login*
- Botón “Recuperar contraseña”

A continuación puedes ver un ejemplo de prototipo de dicho formulario.

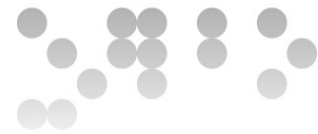
Diagram illustrating the layout of the Login screen prototype:

- Logo placeholder (square with an 'X').
- Email input field.
- Password input field.
- Login button.
- Remember Password button.

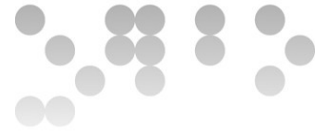
En caso de que no estén correctamente escritos un correo electrónico o falte alguno de los campos, el **botón Login** deberá estar deshabilitado.

3) Pantalla Profile (Estudiante)


Esta pantalla deberá mostrar la información del usuario de manera resumida. Para ello los datos deberán aparecer en varios bloques claramente diferenciados:



- **Datos Personales:**
 - Nombre y apellidos
 - Correo electrónico.
 - Fecha de nacimiento.
 - Teléfono de contacto.
 - NIF/NIE.
 - Sobre mí (una breve descripción).
 - Otras competencias (una breve descripción).
 - Permisos de conducir (un breve texto).
 - Dirección (un breve texto).
- **Formación Académica** (una tabla resumen en la que cada fila contendrá la siguiente información):
 - Nivel, con uno de los siguientes dos valores:
 - Ciclo Formativo (Grado superior, Grado medio o Formación Básica).
 - Universidad (Grado, Diplomado, Licenciado/Ingeniero, Máster, Doctorado).
 - Título (El título académico que tiene), algunos ejemplos:
 - Nivel (Ciclo formativo) - Desarrollo de Aplicaciones Web.
 - Nivel (Universidad) - Ingeniero en Informática
 - Centro (Nombre del IES)
 - Fecha
 - Certificado (Documento con el documento del título o certificado).
 -
- **Idiomas** (una tabla resumen en la que cada fila contendrá la siguiente información):
 - Nivel (A1, A2, B1, B2, C1, C2).
 - Idioma.
 - Fecha.



Esta pantalla mostrará los datos que se irán modificando en los diferentes formularios reactivos que crearás en cada una de las siguientes pantallas. A continuación puedes ver un prototipo de esta pantalla.



Nombre y apellidos

Email: Correo@uoc.com

Fecha de nacimiento: 10/12/1997

Teléfono: 646225443

NIF/NIE: 268986785L

Permisos de conducir: B, A1, A2

Acerca de mí...

Otras competencias

Formación Académica

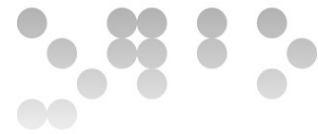
Nivel	Título	Centro	Fecha	Certificado
Universidad	Ingeniero en Informática	Universidad Oberta de Cataluña	10/10/2010	
Ciclo formativo	Desarrollo de aplicaciones Web	IES Politécnico Jesús Marín	10/04/2005	

Experiencia Laboral

Empresa	Puesto	Fechas
Satander	Programador Junior	01/10/2011-01/12/2012
Spotify	Programador Senior	01/01/2013-

Idiomas

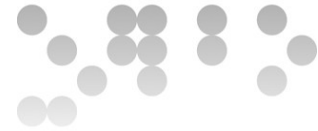
Nivel	Idioma	Fecha
C2	Español	-
B1	Inglés	01/01/2013-



4) Pantalla Profile (Estudiante) – Datos Personales

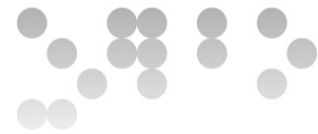
Debe aparecer un formulario para completar los datos del usuario. Este formulario debe tener rellenos aquellos campos para los que tengamos su información almacenada previamente o en blanco si aún no se han relleno/guardado. Los campos vienen descritos a continuación. Cada descripción incluye las validaciones que como mínimo tienes que realizar. Cuando una validación no sea satisfactoria, la web deberá mostrar un mensaje de error a medida que se van produciendo en tiempo real. Además el botón guardar no estará habilitado a menos que se satisfagan todas las validaciones:


- **Datos Personales:**
 - **Nombre:**
 - Longitud entre 3 y 55 caracteres. Sin espacios en blanco al principio ni al final, ni caracteres especiales. Campo obligatorio.
 - **Apellidos:**
 - Longitud entre 3 y 55 caracteres. Sin espacios en blanco al principio ni al final, ni caracteres especiales. Campo obligatorio.
 - **Avatar:**
 - El avatar será una fotografía que, por ahora no haremos nada más que dejar el espacio reservado en nuestro diseño para la subida de éste a través de arrastrar y soltar (se usará un componente específico para esta tarea más adelante).
 - **Fecha de Nacimiento:**
 - Formato: DD/MM/YYYY. Más adelante introduciremos un DatePicker e incluso se podrá validar la fecha usando una potente biblioteca como MomentJS.
 - **Teléfono:**
 - Números. No se pondrán restricciones, puesto que hoy en día puede haber números internacionales.
 - **Teléfono Alternativo:**
 - Igual que el teléfono.
 - **Tipo de Documento:**
 - Un desplegable con las siguientes opciones: NIF/NIE, Otro, Pasaporte.



- **Número de documento:**
 - En función del tipo de documento seleccionado, se deberá tener en cuenta la validación de un NIF/NIE, Pasaporte u otro (libre). Esto lo debes hacer con *custom validations*.
- **Dirección**
 - **Dirección:** String sin restricciones.
 - **Provincia:**
 - Una lista desplegable con las diferentes provincias (deberán venir desde *back-end*, aunque se simulará para esta Práctica).
 - **Municipio:**
 - Una lista desplegable relacionada con las diferentes provincias. Ahora mismo hay que pensar cómo se desarrollaría este punto de relación, el cual se resolverá utilizando *redux* más adelante.
- **Más datos:**
 - **Sobre mí:**
 - Texto largo.
 - **Otras competencias**
 - Texto largo.
 - **Permiso de conducción:** Texto breve libre.

Puedes ver un ejemplo de prototipo de esta pantalla en la siguiente imagen.





140x140

Email/nickname

Nombre

Apellidos

F.Nacimiento

Teléfono

Teléfono 2

DNI/NIF

Número

Dirección

Provincia

Municipio

Permisos de conducir

Sobre mí

B I U
[List Icon] [List Icon] [List Icon]
Normal Font ▾ Size ▾ [Color Icon] [Color Icon]

Otras competencias

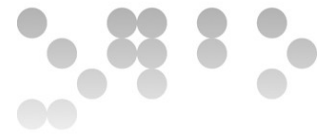
B I U
[List Icon] [List Icon] [List Icon]
Normal Font ▾ Size ▾ [Color Icon] [Color Icon]

Guardar

5) Pantalla Profile (Estudiante) – Formación Académica

Aparecerá una tabla con la información de resumen de las diferentes formaciones agregadas por parte del usuario. Cada una de estas filas contendrá un campo denominado acciones en la que se agregarán los botones “borrar” y “editar”.

Además, existirá un botón denominado “Nueva” para agregar nuevas formaciones. El botón *borrar* eliminará una formación, mientras que el botón *editar* y *nueva* redirigirán al mismo formulario (componente) el cual tendrá un funcionamiento u otro en función de si la acción es “nueva” o “editar”. Ahora mismo, puedes falsear el estado del formulario de la manera más óptima que veas, puesto que más adelante se hará uso del router/redux.



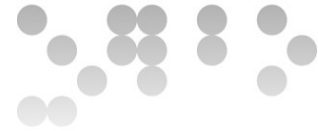
Este formulario estará compuesto por los siguientes campos a validar:

- **Tipo de título:**
 - Lista desplegable: Título universitario, ciclo formativo, otro título.
- **Opción Título Universitario:**
 - Nombre de centro: String libre.
 - Título: String libre.
 - Fecha del título: DD/MM/YYYY.
 - Formación bilingüe (checkbox).
 - Certificado (Posibilidad de subir un fichero).
- **Opción Ciclo formativo:**
 - Centro educativo: Lista desplegable con todos los centros educativos.
 - Familia profesional: Lista desplegable con todas las familias profesionales existentes.
 - Grado: Lista desplegable con las opciones – Grado Superior, Grado Medio y FP Básica.
 - Ciclo: Lista desplegable con las opciones en función de la familia profesional existente y el grado (combo doblemente combinado).
 - Fecha de obtención del título: DD/MM/YYYY.
 - Formación dual (checkbox).
 - Formación bilingüe (checkbox).
 - Certificado (Posibilidad de subir un fichero).

6) Pantalla Profile (Estudiante) – Idiomas

Aparecerá una tabla con la información de resumen de los diferentes idiomas agregados por parte del usuario. Cada una de estas filas contendrá un campo denominado acciones en la que se agregarán los botones “*borrar*” y “*editar*”.

Además, existirá un botón denominado “*Nueva*” para agregar nuevos idiomas. El botón *borrar* eliminará un idioma, mientras que el botón *editar* y *nueva* redirigirán al mismo formulario (componente) el cual tendrá un funcionamiento u otro en función de si la acción es “*nueva*” o “*editar*”.



Este formulario estará compuesto por los siguientes campos a validar:

- Idioma: Lista desplegable con varios idiomas, y la opción Otros. En caso de que se haya marcado Otros, aparecerá un recuadro para escribir un idioma con una longitud entre 3-255 caracteres y no podrá estar en blanco.
- Nivel: Lista desplegable con las opciones A1, A2, B1, B2, C1, C2.
- Fecha de obtención del título: DD/MM/YYYY.

7) Pantalla JOB OFFERS (Estudiantes)

En esta pantalla se muestra una tabla con las diferentes ofertas propuestas por las empresas para encontrar trabajo. Sólo se mostrarán las ofertas en las que el candidato cumpla con su formación. Es decir, que habrá una tabla con los siguientes datos:

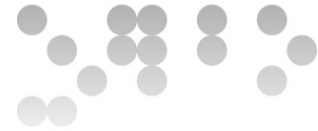
- Puesto: Nombre del puesto de trabajo
- Empresa: Nombre de la empresa.
- Familia profesional (Para la que va destinada la oferta de trabajo).
- Fecha.
- Provincia.

Se dispondrán de varios filtros (no implementados ahora mismo) para filtrar por provincia y fecha.

Pantalla JOB OFFERS (Estudiantes) – Detalle

Al hacer clic sobre alguna de las filas se mostrará la oferta (detalle), la cual será un formulario/pantalla de visualización con los siguientes datos:

- **Oferta**
 - Empresa (nombre).
 - Puesto.
 - Descripción (texto largo).
- **Municipio**
 - Provincia



- Municipio
- **Destinado a persona de**
 - Familia: Familia profesional
 - Tabla con los títulos a los que va destinada la oferta.
- **Botón de inscribirse a la oferta.**
- **Oferta**
 - Empresa (nombre).
 - Puesto.
 - Descripción (texto largo).
- **Municipio**
 - Provincia
 - Municipio
- **Destinado a persona de**
 - Familia: Familia profesional
 - Tabla con los títulos a los que va destinada la oferta.
- Botón de inscribirse a la oferta.**

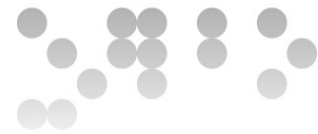
Pantalla MY JOB OFFERS (Estudiantes)

Se mostrará la misma pantalla que para *Job Offers* pero solamente se mostrarán los datos relativos a las ofertas en las que el alumno se ha inscrito.

La diferencia que existirá es que habrá una columna denominada *Estado* que indicará el estado en el que se encuentra la oferta.

Pantalla MY JOB OFFERS (Estudiantes) – Detalle

Igual que en la pantalla *Job Offers*, pero el botón *inscribirse* de la oferta se cambiará por el de *borrarse* de la oferta.



PANTALLAS EXTRAS PARA ALCANZAR UNA CALIFICACIÓN MAYOR

A continuación se describen las pantallas que corresponden a la parte extra de la Práctica. Para comenzar esta parte se deben haber realizado las pantallas mínimas de la aplicación puesto que serán las que se utilizarán en posteriores PECs

Pantalla Profile (Estudiante) – Experiencia laboral

Aparecerá una tabla con la información de resumen de las diferentes experiencias laborales agregadas por parte del usuario. Cada una de estas filas contendrá un campo denominado acciones en la que se agregarán los botones “borrar” y “editar”.

Además, existirá un botón denominado “**Nueva**” para agregar nuevas experiencias. El botón *borrar* eliminará una experiencia, mientras que el botón *editar* y *nueva* redirigirán al mismo formulario (componente) el cual tendrá un funcionamiento u otro en función de si la acción es “nueva” o “editar”.

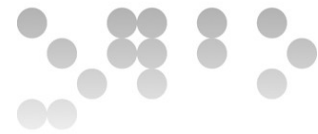
Este formulario estará compuesto por los siguientes campos a validar:

- Nombre de la empresas: Alfanumérico entre 3 – 255 caracteres. Sin espacios en blanco inicialmente.
- Fecha Inicio: DD/MM/YYYY.
- Fecha final: DD/MM/YYYY.
- Puesto desempeñado: Alfanuméricos entre 3 – 255 caracteres. Sin espacios en blanco inicialmente.
- Tareas realizadas en el puesto desempeñado. Un texto largo.

Pantalla Profile (Empresa) – Datos

Esta pantalla es muy parecida a la pantalla creada para un estudiante, puesto que se encargará de gestionar la información relativa a la de una empresa. En este caso, la información que necesitamos gestionar es la siguiente:

- Nombre Comercial: String obligatorio entre 3 y 255 caracteres sin espacios en blanco.
- Razón Social: Igual que el nombre comercial.



- CIF: String libre, puesto que pueden existir CIF internacionales.
- Dirección: String.
- Provincia/Municipio: Igual que en el perfil del usuario.
- URL: Página Web de la empresa.
- Datos de contacto:
 - Persona de contacto – Nombre y Apellidos.
 - Persona de contacto – Teléfono.
 - Persona de contacto – Correo electrónico.

Pantalla JOB OFFERS (Empresa)

La pantalla *Job Offers* de empresa debe mostrar una tabla con las diferentes ofertas propuestas por la empresa. Se dispondrá de un botón *editar* que permitirá acceder a dicha oferta y poder conocer los candidatos que se han suscrito a dicha oferta. Se debe dar respuesta a cada una de las ofertas suscritas por parte de los alumnos (DESCARTANDO O SELECCIONANDO).

Además, al igual que en otras pantallas, se podrán crear nuevas ofertas que puedan visualizar los estudiantes. En este caso, a partir de los campos anteriormente descritos en el enunciado, trata de obtener los diferentes campos. Puedes debatir con los compañeros en el foro y con el profesor asociado qué campos serían los adecuados.

Pantalla CONFIGURACIÓN (Empresa)

Esta pantalla se utiliza con un formulario que permite configurar diferentes cuestiones personalizadas de los usuarios tales como:

- **Idioma:** El idioma de la aplicación (será un desplegable). Inicialmente la aplicación estará en un único idioma, pero con un módulo se podrá hacer internacional fácilmente (lo haremos en otra práctica).
- **Notificaciones:** Si desea recibir notificaciones de nuevas ofertas a su correo electrónico.



- Habrá un checkbox para cada una de las provincias, de este modo cuando surja una oferta en la provincia marcada y sobre su formación será avisado por correo electrónico.
- Habrá un checkbox que automáticamente marque todas las provincias o las desmarque.

Bibliografía/Webgrafía

Toda la comunidad de la UOC dispone de un extenso catálogo de publicaciones que pueden ser consultadas utilizando el usuario de la UOC por ello se recomiendan las siguientes lecturas:

Router

- https://learning.oreilly.com/videos/learning-angular-covering/9780134848426/9780134848426-LAJ5_01_06_01
- <https://learning.oreilly.com/library/view/angular-router/9781787288904/pr01.html>

Formularios reactivos

- https://learning.oreilly.com/library/view/build-reactive-websites/9781680506495/f_0054.xhtml
- https://learning.oreilly.com/library/view/reactive-programming-with/9781484226193/A431785_1_En_7_Chapter.html