Examen técnico Angular

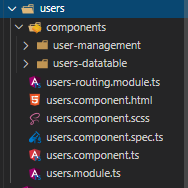
Recursos:

* Cuentas con el “package.json” configurado con todas las librerías necesarias para realizar la solución. No obstante, configurar los estilos o inyectar los módulos según los requerimientos (Por ejemplo, incluir el datepicker, la modal de confirmación, entre otras).
* Utilizar el archivo fakeBackend para hacer las llamadas necesarias.
* Bootstrap 4: <https://getbootstrap.com/>
* Ngx Bootstrap: <https://valor-software.com/ngx-bootstrap/#/>
* Ngx Datatable: <https://github.com/swimlane/ngx-datatable>
* Ngx Toastr: <https://www.npmjs.com/package/ngx-toastr>
* Moment js: <https://momentjs.com/>
* Ng select: <https://ng-select.github.io/ng-select#/data-sources>
* Material icons: <https://material.io/resources/icons/?style=baseline>

## ¿Qué se valora de la prueba?

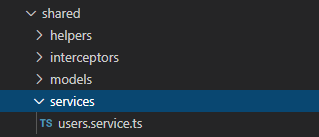
* El uso de las diferentes librerías y como moverse por las documentaciones.
* Aunque no se haga como se pide, se consiga alguna otra manera de hacerlo.
* *Clean code* y estructura rápidamente entendible.

# Estructura general del módulo de usuarios



La gestión de los usuarios se hará mediante un servicio llamado users.service que ya tienes creado.

Tendrás que hacer uso de los métodos Create, Read, Update, Delete.



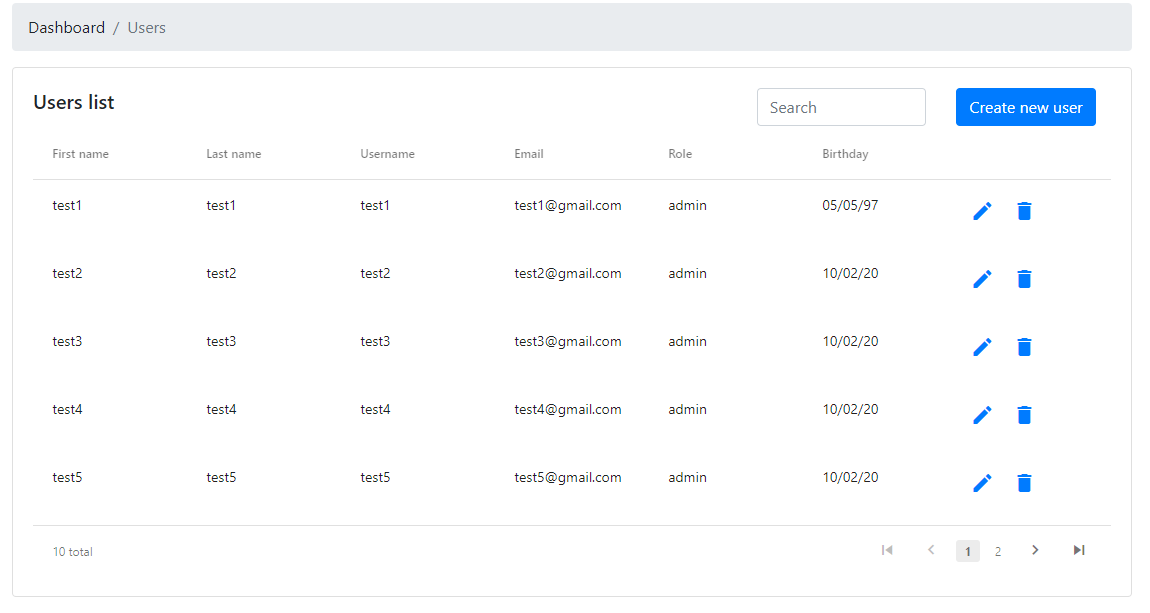
Requisitos USERS.SERVICE**:**

* Crear todas las llamadas o métodos necesarios para crear/actualizar/eliminar/obtener usuario/obtener listado de usuarios.
* Cuando la llamada devuelva un 200 se informará con un toast que la acción se ha hecho bien, con el uso de ngx-toastr.

# Componentes

Users

Listado de los usuarios con los botones para un CRUD y un input para filtrar los resultados en client-side.



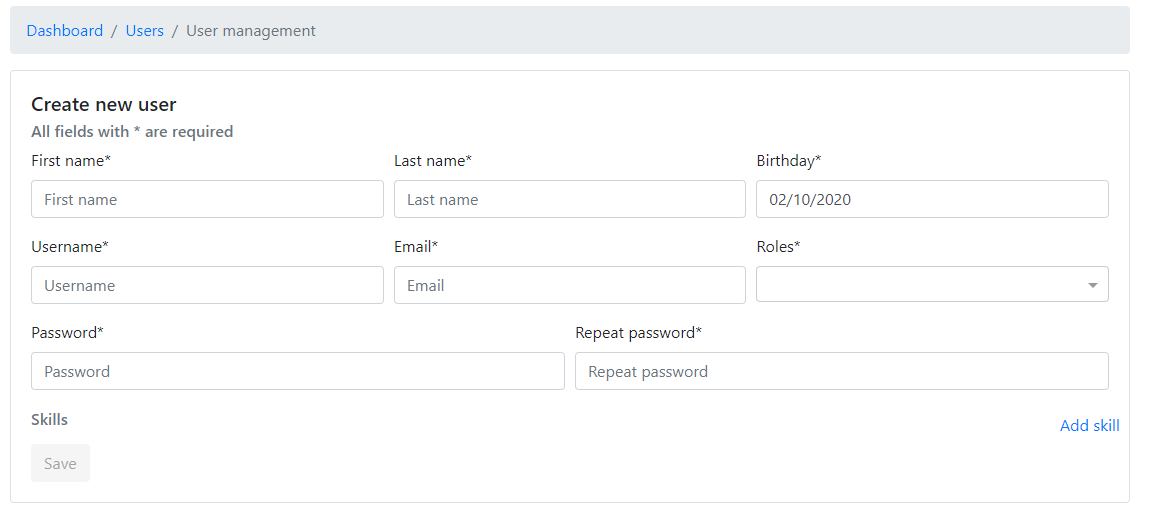
Los íconos son de material icons, ya tienes el link añadido al index.html.

Requisitos:

* **Buscador:** Leer la documentación de ngx-datatable y crear un filtro para todas las columnas en client-side.
* **Columna Birthday:** Mostrar el formato que se muestra en la imagen.
* **Crear usuario:** Crear un routerLink para acceder a la página de user-management.
* **Editar usuario:** Crear un routerLink para acceder a la página de user-management/{id}, después en la página tendrás que recoger el id.
* **Eliminar usuario:** Hacer uso de la modal de ngx Bootstrap para crear una ventana emergente de confirmación.
  + La modal tiene que ser de tipo componente con el nombre de confirm- modal.component
  + Tiene que estar ubicada en app/shared/ui-components
  + Es muy importante que sea genérica, eso quiere decir que como parámetros se le deberá pasar el título y el mensaje que se visualizará en la modal.
  + Cuando la modal te haga confirmación devolverá un true para poder eliminar el usuario.
* Importante: al ser genérica y por tanto reutilizable para otras vistas, eso quiere decir que no puede haber ningún tipo de lógica de usuarios dentro del componente.

UserManagement:

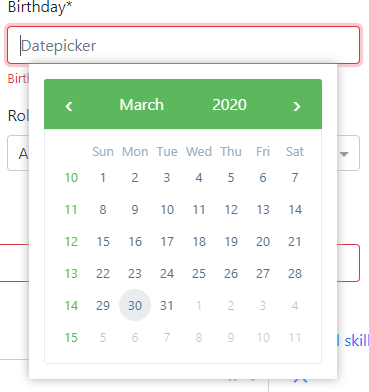
Formulario para gestionar cuando se crea y se edita un usuario.



Importante**: s**e proporciona una función en la clase Guid de la carpeta helpers para generar el GUID en caso de que añadas un nuevo usuario.

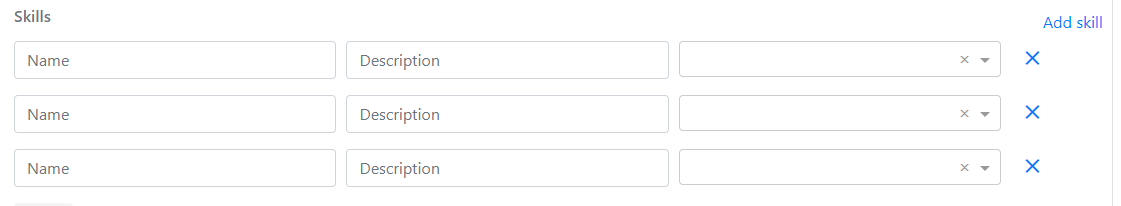
Requisitos**:**

* El título de la card se modificará dependiendo de si está editando o creando.
  + Cuando se cree tendrá el nombre de “Create new user” y cuando se edite se mostrará “Edit user: {nombre del usuario}”.
* Todos los campos son requeridos menos las skills (abajo se explica con más detalle).
* Para el birthday hacer uso del datepicker de ngx-bootstrap



* Se ha de crear una clase con el nombre de CustomValidators donde habrá un validador para la fecha y otros para contraseña.
  + Fecha 1: Con el nombre de “minimumAge” se validará que la fecha introducida sea mayor de edad. (Nota: Puedes hacer uso de la librería moment.js)
  + Contraseña 2: Con el nombre de mustMatch se comprobará que las contraseñas coincidan entre ellas.
* Las contraseñas también han de tener una validación de mínimo de caracteres.
* Cuando estés en modo edición se tendrán que eliminar las validaciones de los dos campos de contraseña y esconder los campos.
* Validar que el email tenga el formato correcto.
* El selector de roles se ha de hacer uso del Ng select.

Skills**:**



Requisitos skills**:**

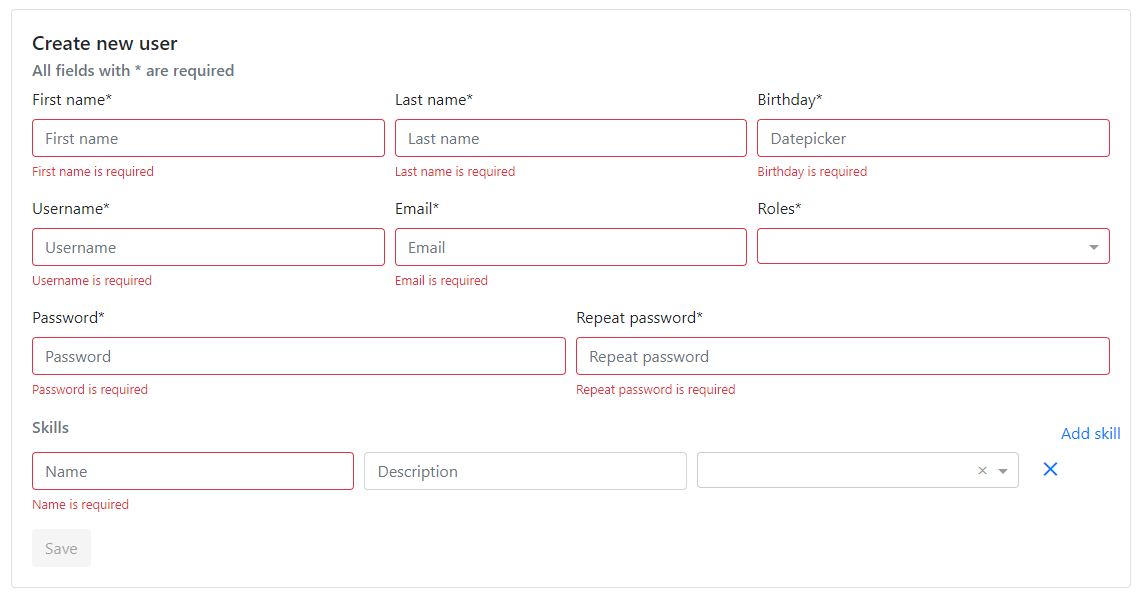
* El listado de skills no es requerido, pero en caso de que se añada un formulario si que será requerido rellenar el nombre.
* En caso de que haya formularios de skills vacíos, el botón para crear/modificar usuario estará deshabilitado.
* Se podrá crear tantas skills como se quiera de manera dinámica con el botón de “Add skill” y eliminar con la X.
* Los campos de descripción y categoría no son requeridos.

# Imagen del formulario con errores

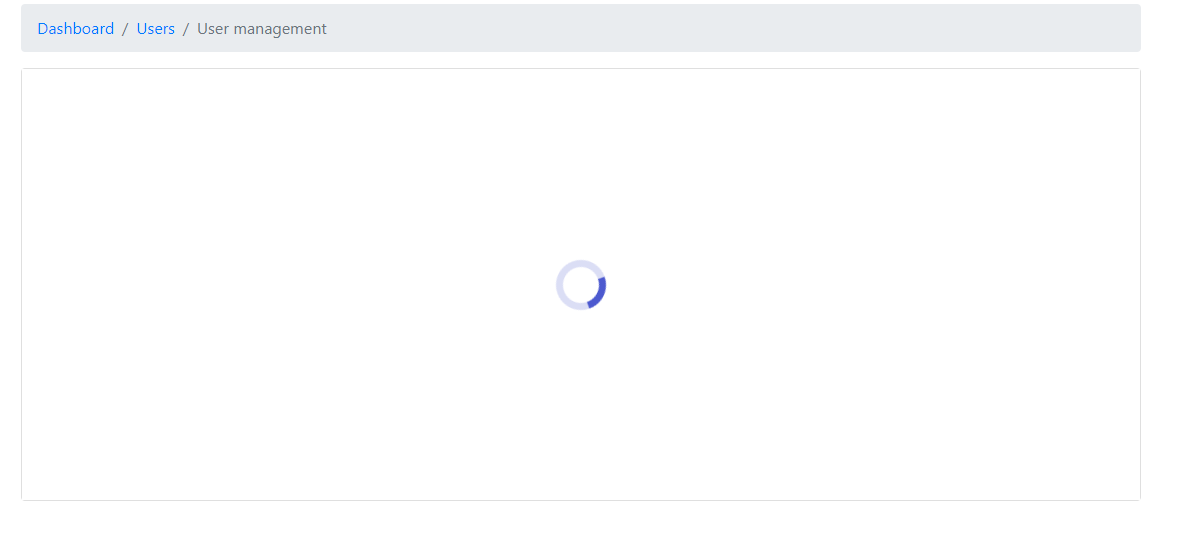
Es importante que las validaciones de la UX estén bien, los mensajes se han de mostrar de manera dinámica.

* Cuando sales de cada campo y está vacío se tiene que mostrar el error.
* Cuando estás rellenando el campo se tiene que validar y en caso de que cumpla las validaciones se tiene que esconder los errores y en caso de que tenga mas de una validación se tiene que ir actualizando el mensaje de cada campo.
* El botón para guardar estará deshabilitado hasta que el formulario no sea válido.

(Ejemplo: Cuando hagas touch en email y lo dejes vacío “Field is required”, cuando has introducido texto pero el formato no es correcto “Format is not valid”)



## Imagen del formulario cuando se están obteniendo o guardando los datos



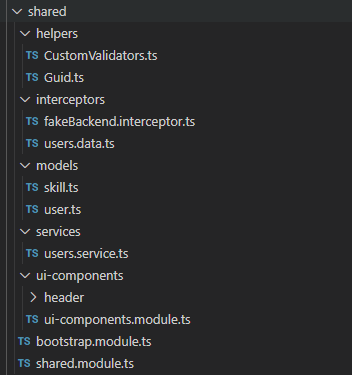
Cuando se acceda a un usuario para editar o se haga click en guardar se mostrará un spinner para que el usuario sepa que se está haciendo alguna acción.

UserDatatable

Subcomponente de Users donde estará la ngx datatable con el listado de usuarios.

* El buscador estará en el padre (Users) y si le deberá pasar a este componente el valor del input para poder filtrar la tabla.

## Estructura de la carpeta shared



**CustomValidators:** Crear la clase CustomValidators para los dos métodos explicados anteriormente (Fecha 1 y Contraseña 2).

**UserService**: Para hacer las llamadas al fakeBackend.

* GET api/users – Obtener el listado de usuarios.
* GET api/users/{id} – Obtener un usuario.
* POST api/users/create – Obtener un nuevo usuario.
* POST api/users/update – Modificar un usuario.
* DELETE api/users/{id} – Eliminar un usuario.

**UIcomponents:**

* Crear confirm-modal como se ha explicado anteriormente en el requisito **Eliminar usuario**.