

CODEACHIDO ANGULAR TODOLIST

18/abril/2019

Angular :es un entorno de desarrollo frontend (no se mete con el manejo de la información de los sistemas solo hace la parte visual);Es un framework basado en componentes que usa el paradigma de POO.Los componentes son las clases y nos permiten hacer escalarle, reutilizar código tambien funciona como base para aplicaciones móviles híbridas usando la tecnología unic; podemos programar en Android y iOS- con el mismo código a esto se le llaman Aplicaciones híbridas

Node: es un ambiente para desarrollo javascript para hacer backend donde hacemos uso de las herramientas de compilacion,dependencias etc para hacer un mini servidor y así poder levantar la pagina

Git -versionamiento de código

Npm :gestor de dependencias

-g es de forma global

Cli: comandand ligh interfase; Componentes,directivas etc crea archivos con código

Ng new nombre :ng dice haz una madre para angular new indicar nuevo proyecto en angular luego el nombre

- Pregunta si lo quieres agregar y decimos que si
- Css dar estilos estilos hoja de cascada
- Scss reprosesador de estilos permite poner programación fors ifs etc y se compilan creando un css normal. <<<
- Sass (tipo phyton),Less, Stylus

Spa: single page application, aqui Haremos sitios que no necesiten refrescarse

- ng serve —open comando, accion y una bandera para la accion compila de tope a js y no es visible el código que compilo hasta que sea para producción
 - Abre navegador o usar localhost:4200
 - Terminamos el proceso en la terminal con control-c
- Ahora abrimos Visual Code y abrimos el proyecto todolist

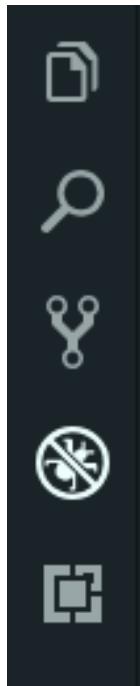
Entramos a nuestra cuenta de GitHub

Crearemos un nuevo repositorio para guardar los cambios y avances dentro de el proyecto

Abrimos la terminal con control y ` (a lado de la p) en mac

Node modules son las dependencias del proyecto

En visual code sus partes son



- Archivos
 - Buscador
 - Debug
 - Control de git
 - Extensiones
-
- Volvemos a usar el comando ng serve —open pero dentro de la terminal en visual code
 - Creamos un repositorio en GitHub y subiremos a GitHub
 - Git add .
 - Git remote add origin
 - Git config user.name "nombre de usuario de la cuenta"
 - Git config user.email "correo de el usuario "
- Para confirmar podemos usar git config —get user.name

Recargamos la pagina y observamos que no esta package en GitHub, esto es debido a que esta es muy pesada y se encuentra en git ignore.

angular.json es un archivo de configuración inicial

package.json dependencias y scripts

Tsconfigs es para **js-typescript**

app.component.json ???

- Vamos a src/app/app.component.html borramos todo menos la etiqueta

```
<router
```

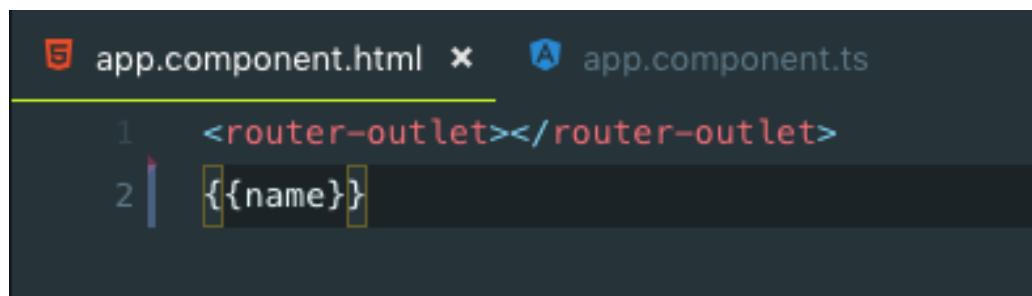
Dentro de **app.components.ts** encontramos los decoradores, en el html

recargamos los de este archivo

En **environment.ts** veremos los atributos, agregaremos name con nuestro nombre

```
export class AppComponent {  
  title = 'todolist';  
  name='Pamela';  
}
```

En html pondremos llaves dobles y el name; a esto le llamamos **Interpolación**; que es decirle que meta el valor del atributo y es así que nosotros tenemos nuestro nombre en el navegador



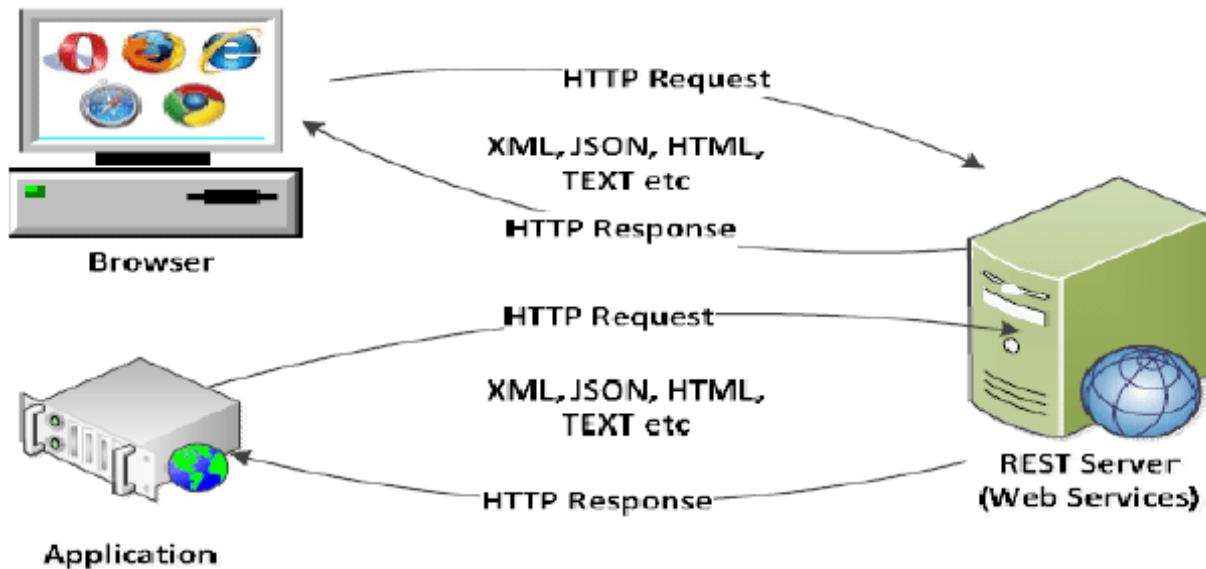
The screenshot shows a code editor with two tabs: 'app.component.html' and 'app.component.ts'. The 'app.component.html' tab contains the following code:

```
1 <router-outlet></router-outlet>  
2 {{name}}
```

The 'app.component.ts' tab is visible but empty.

- App(aquí va todo el código)
- Assets (archivos adjuntos en la aplicación)
- environments(

Trabajamos abajo un esquema **REST**



El cliente hace petición para la información y nuestro trabajo es hacerlo vistoso; cuando entramos a una aplicación http hacemos una petición al servidor que tenga relacionado el dominio del nombre de la página

En environments/environments.ts

Agregamos el siguiente código que es la url para desarrollo

```

4
5   export const environment = {
6     production: false,
7     apiURL: 'http://192.168.1.66:300/api'
8   };

```

Api es : La **interfaz de programación de aplicaciones**, conocida también por la sigla **API**, en inglés, *application programming interface*, es un conjunto de subrutinas, funciones y procedimientos (o métodos, en la Programación orientada a Objetos) que ofrece cierta biblioteca para ser utilizado por otro sw como una Capa de abstracción.

- Index.html es lo que podemos ver, se carga
Podemos utilizar <div *ngIf="name=='hola'"> para hacer un if
 - Ng generate component login (genera el componente /module/service podemos poner otra cosa)
- Esto creara otra carpeta y un componente llamado login
- Usaremos bootstrap <https://www.npmjs.com/package/ngx-bootstrap> esta es una dependencia que podemos instalar con el comando: npm install ngx-bootstrap --save

Este lo instalara sobre nuestro archivo package.json y así sabrá que dependencias se usaran en caso de mantenimiento

Tenemos app-routing-module.ts este es el configurado de rutas especiales, haremos un objeto dentro de la linea 4 y escribiremos el siguiente código

```
import { Routes, RouterModule } from '@angular/router';
import { LoginComponent } from 'src/app/login/login.component';

const routes: Routes = [
  {
    path: 'login',
    component: LoginComponent
  },
];

```

Terminamos el proceso en la terminal con control c

Y ahora creamos otro componente en la terminal con **ng generate componentes signup** así creamos el componente "signup"

Definiremos una ruta en este arreglo de objetos; crearemos otro objeto de la siguiente forma

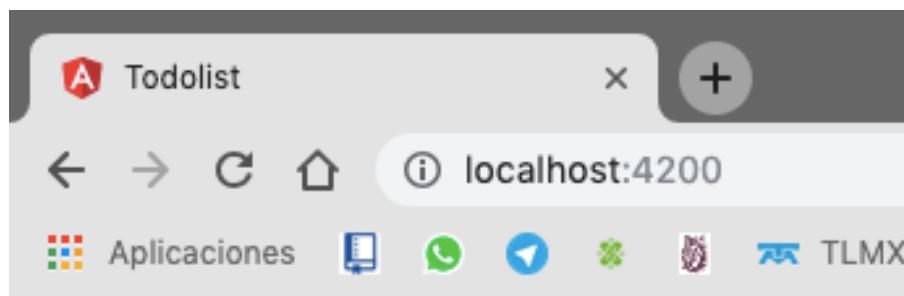
```
import { SignupComponent } from 'src/app/signup/signup.component';

const routes: Routes = [
  {
    path: 'login',
    component: LoginComponent
  },
  {
    path: 'signup',
    component: SignupComponent
  }
];

```

Posteriormente crearemos botones para redireccionar a signup y login en app.component.html

```
<div class="container">
  <h1>Todo List</h1>
  <router-outlet></router-outlet>
  <p>Este es un parrafo</p>
</div>
```



Todo List

Este es un parrafo

Abrimos package.json y nos vamos a las dependencias ponemos npm install bootstrap —save

Ahora abrimos en environment angular.json

```
"styles": [
  "./node_modules/bootstrap/scss/bootstrap.scss",
  "src/styles.scss"]
```

en app.component.html

```
<div class="text-center">
  <h1>Todo List</h1>
</div>
```

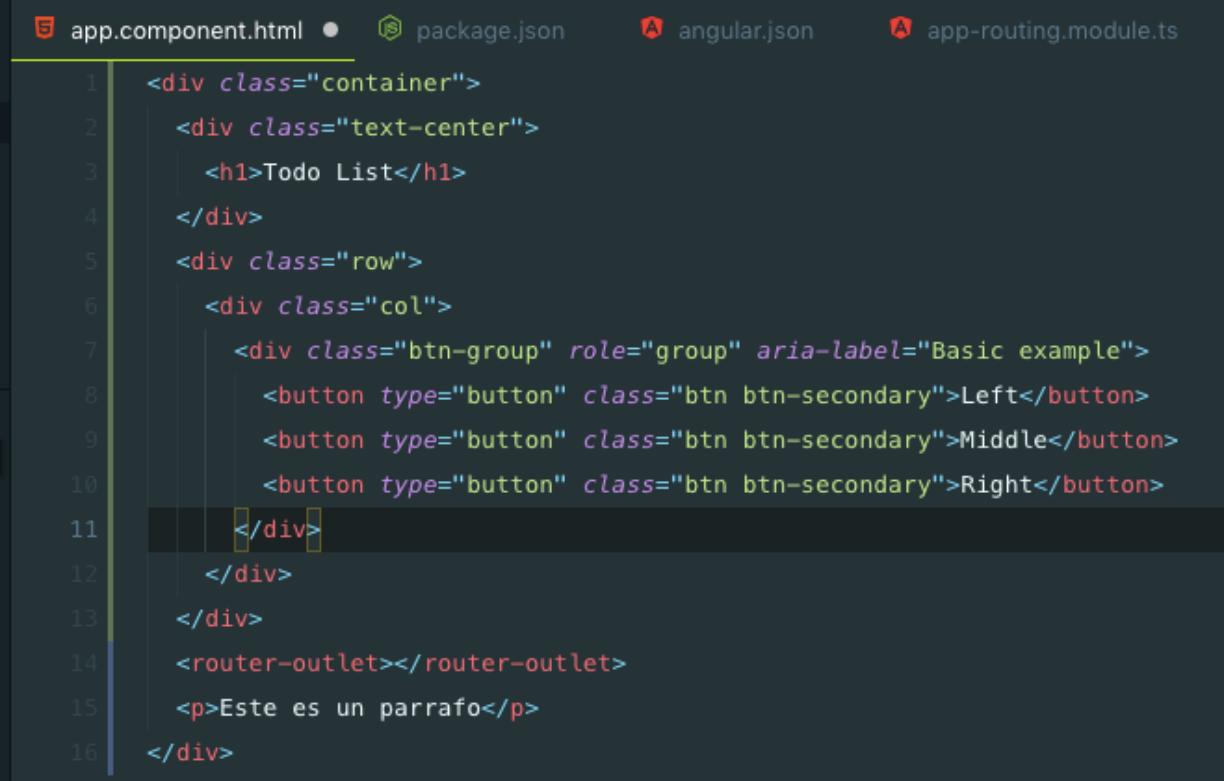
Centramos el titulo

Y agregamos

```
<div class="row">
  <div class="col">
  </div>
</div>
```

Ahora en bootstrap en componentes (<https://getbootstrap.com/docs/4.0/components/button-group/>) copiamos el código de basic example y lo ponemos en el div col;

Acomodamos el código con **Alt shift F**.



```
1  <div class="container">
2    <div class="text-center">
3      <h1>Todo List</h1>
4    </div>
5    <div class="row">
6      <div class="col">
7        <div class="btn-group" role="group" aria-label="Basic example">
8          <button type="button" class="btn btn-secondary">Left</button>
9          <button type="button" class="btn btn-secondary">Middle</button>
10         <button type="button" class="btn btn-secondary">Right</button>
11       </div>
12     </div>
13   </div>
14   <router-outlet></router-outlet>
15   <p>Este es un parrafo</p>
16 </div>
```

Modificamos los botones de la siguiente forma

```
<div class="btn-group" role="group" >
  <button type="button" class="btn btn-secondary">Login</button>
  <button type="button" class="btn btn-secondary">Signup</button>
</div>
```

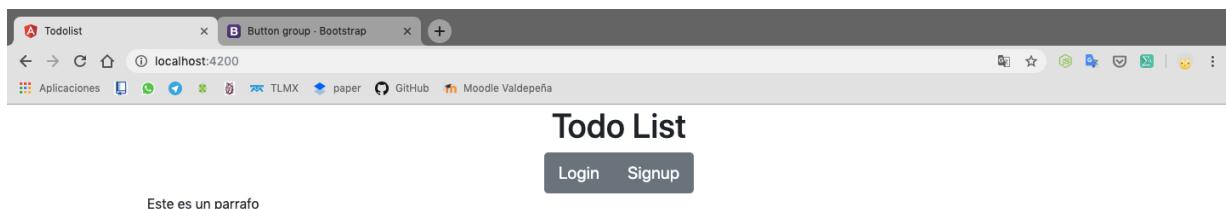
Centramos los botones. Los hacemos más grandes y por lo tanto el código queda de la siguiente manera

```

<div class="row">
  <div class="col text-center">
    <div class="btn-group btn-group-lg " role="group" >
      <button type="button" class="btn btn-secondary">Login</button>
      <button type="button" class="btn btn-secondary">Signup</button>
    </div>
  </div>
</div>

```

Y nuestra pantalla de la siguiente manera



Las etiquetas **a** son para los links por lo tanto cambiamos los botones y cambiamos el código de la siguiente manera para colocarle los colores a los botones

```

app.component.html
app.component.scss
packa

```

```

1 .active{
2   background-color: #007bff !important;
3   border-color: #007bff !important;
4   color: #fff !important;
5 }

```

Ahora creamos la clase active

```
<div class="row">
  <div class="col text-center">
    <div class="btn-group btn-group-lg" role="group" >
      <a class="btn btn-light" [routerLink]=["/login"]>Login</a>
      <a class="btn btn-light" [routerLink]=["/signup"]>Signup</a>
    </div>
  </div>
</div>
```

Ahora en los botones después del routerLink colocamos routerLinkActive="active"

```
<a class="btn btn-light" [routerLink]=["/login"]" routerLinkActive="active">Login</a>
```

RAMAS

Git checkout -b nombre. (Crear la rama en la maquina)

Git branch

Ahora de la pagina <https://getbootstrap.com/docs/4.0/components/card/> copiamos Body en login.html y colocamos un

```
<div class="card m-2">
```

Para posteriormente pegar la tarjeta dentro de este

Ahora en <https://getbootstrap.com/docs/4.0/components/forms/> dentro de lo que sería el titulo

Ahora subimos a g

it de la siguiente forma

```
MacBook-Air-de-Pamela:todolist pamelaruiz$ git branch
* login-design
  master
MacBook-Air-de-Pamela:todolist pamelaruiz$ git add .
MacBook-Air-de-Pamela:todolist pamelaruiz$ git commit -m "login design"
[login-design 50e7f73] login design
 1 file changed, 23 insertions(+), 3 deletions(-)
   rewrite src/app/login/login.component.html (100%)
MacBook-Air-de-Pamela:todolist pamelaruiz$ git push -u origin login-design
```

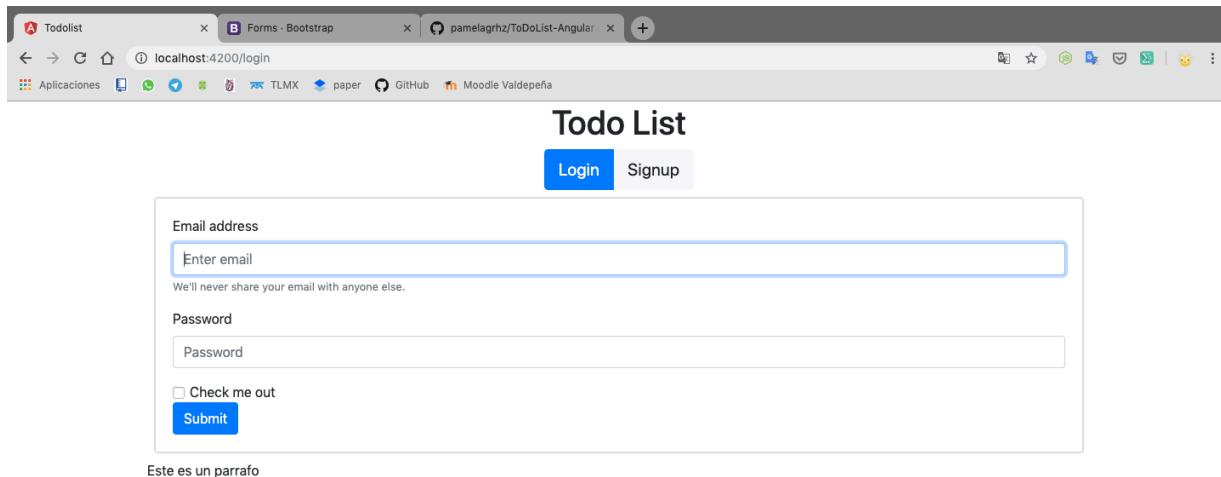
Git branch

Git add .

Git commit -m "comentario"

Git push -u origin login-design(el mismo nombre de la rama)

FINALIZAMOS CON EL RESULTADO



TODOLIST 2/4

Primero regresaremos a la rama Origin master y podremos ver que regresa el archivo como estaba antes, ahora creamos una nueva rama que es signup-design

```
MacBook-Air-de-Pamela:todolist pamelaruiz$ git branch
* login-design
  master
MacBook-Air-de-Pamela:todolist pamelaruiz$ git checkout master
Switched to branch 'master'
Your branch is up to date with 'origin/master'.
MacBook-Air-de-Pamela:todolist pamelaruiz$ git checkout -b signup-design
Switched to a new branch 'signup-design'
MacBook-Air-de-Pamela:todolist pamelaruiz$ git branch
  login-design
  master
* signup-design
MacBook-Air-de-Pamela:todolist pamelaruiz$ █
```

Ahora en signup.html pegamos el primero formulario de <https://getbootstrap.com/docs/4.0/components/forms/>

For en los label nos dice a que esta relacionado placeholder es lo que aparece en el recuadro cuando no hay nada escrito

Type pueden ser : text, email, password y submit (boton)

Y nosotros les damos el formato que queremos, quedando el codigo de la siguiente manera

```
<form>
  <div class="form-group">
    <label for="signupEmail">Email address</label>
    <input type="email" class="form-control" id="signupEmail" placeholder="Enter email">
  </div>
  <div class="form-group">
    <label for="signupName">Name</label>
    <input type="text" class="form-control" id="signupName" placeholder="Enter your Name">
  </div>
  <div class="form-group">
    <label for="signupLastName">Last Name</label>
    <input type="text" class="form-control" id="signupLastName" placeholder="Enter your Last Name">
  </div>
  <div class="form-group">
    <label for="signupUserName">User-Name</label>
    <input type="text" class="form-control" id="signupUserName" placeholder="Enter your User-Name">
  </div>
  <div class="form-group">
    <label for="exampleInputPassword1">Password</label>
    <input type="password" class="form-control" id="exampleInputPassword1" placeholder="Password">
  </div>
  <div class="form-group">
    <label for="exampleInputPassword2">Confirm your Password</label>
    <input type="password" class="form-control" id="exampleInputPassword2" placeholder="Password">
  </div>
  <button type="submit" class="btn btn-primary btn-block">Submit</button>
</form>
```

Quedando como resultado

Todo List

[Login](#) [Signup](#)

Email address

Enter email

Name

Enter your Name

Last Name

Enter your Last Name

User-Name

Enter your User-Name

Password

Password

Confirm your Password

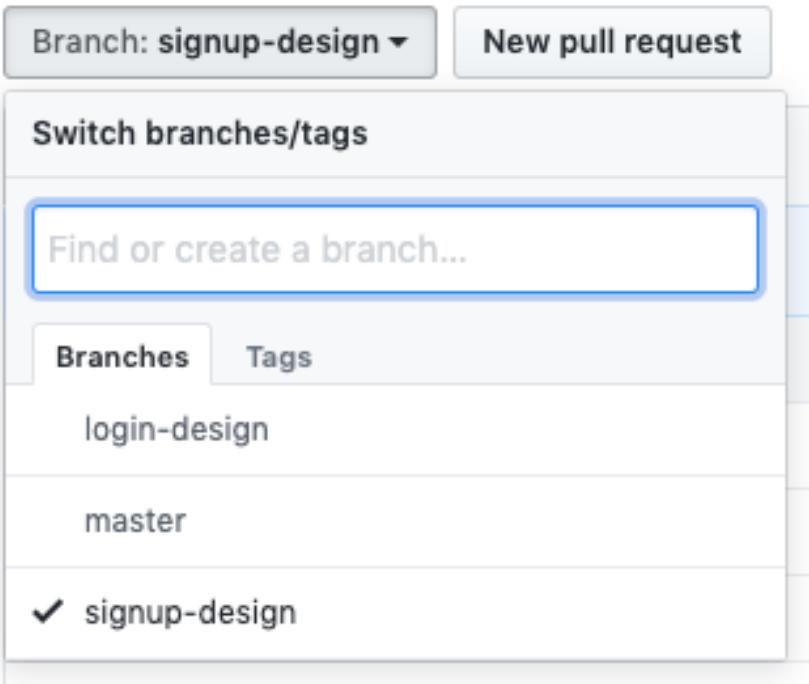
Password

Submit

Este es un parrafo

Y lo guardamos en una nueva rama en GitHub

```
MacBook-Air-de-Pamela:todolist pamelaruiz$ git add .
MacBook-Air-de-Pamela:todolist pamelaruiz$ git commit -m "form signup"
[signup-design f6a3609] form signup
 2 files changed, 29 insertions(+), 5 deletions(-)
  rewrite src/app/signup/signup.component.html (100%)
MacBook-Air-de-Pamela:todolist pamelaruiz$ git push -u origin signup-design
Enumerating objects: 14, done.
Counting objects: 100% (14/14), done.
Delta compression using up to 4 threads
Compressing objects: 100% (7/7), done.
Writing objects: 100% (8/8), 858 bytes | 858.00 KiB/s, done.
Total 8 (delta 5), reused 0 (delta 0)
remote: Resolving deltas: 100% (5/5), completed with 5 local objects.
remote:
remote: Create a pull request for 'signup-design' on GitHub by visiting:
remote:   https://github.com/pamelagrhz/ToDoList-Angular/pull/new/signup-design
remote:
To https://github.com/pamelagrhz/ToDoList-Angular.git
 * [new branch]      signup-design -> signup-design
Branch 'signup-design' set up to track remote branch 'signup-design' from 'origin'.
MacBook-Air-de-Pamela:todolist pamelaruiz$ █
```



Ahora combinaremos las ramas dejando en la rama master los cambios de las dos ramas usando:

Git checkout master

Git branch(saber en que rama estamos)

git merge login-design(juntar la rama login design con la rama master)

Creamos otra rama llamada

Ahora en style.scss

Ahora entramos a la pagina <https://github.com/FortAwesome/angular-fontawesome>

Colocamos el código

Installation

```
$ yarn add @fortawesome/fontawesome-svg-core  
$ yarn add @fortawesome/free-solid-svg-icons  
$ yarn add @fortawesome/angular-fontawesome
```

Sustituyendo \$yarn add por (nom install —save)

Npm install —save @fortawesome/fontawesome-svg-core @fortawesome/free-solid-svg-icons @fortawesome/angular-fontawesome

Seguimos moviendo en style scss

right: 8px; cuento se mueve desde la derecha

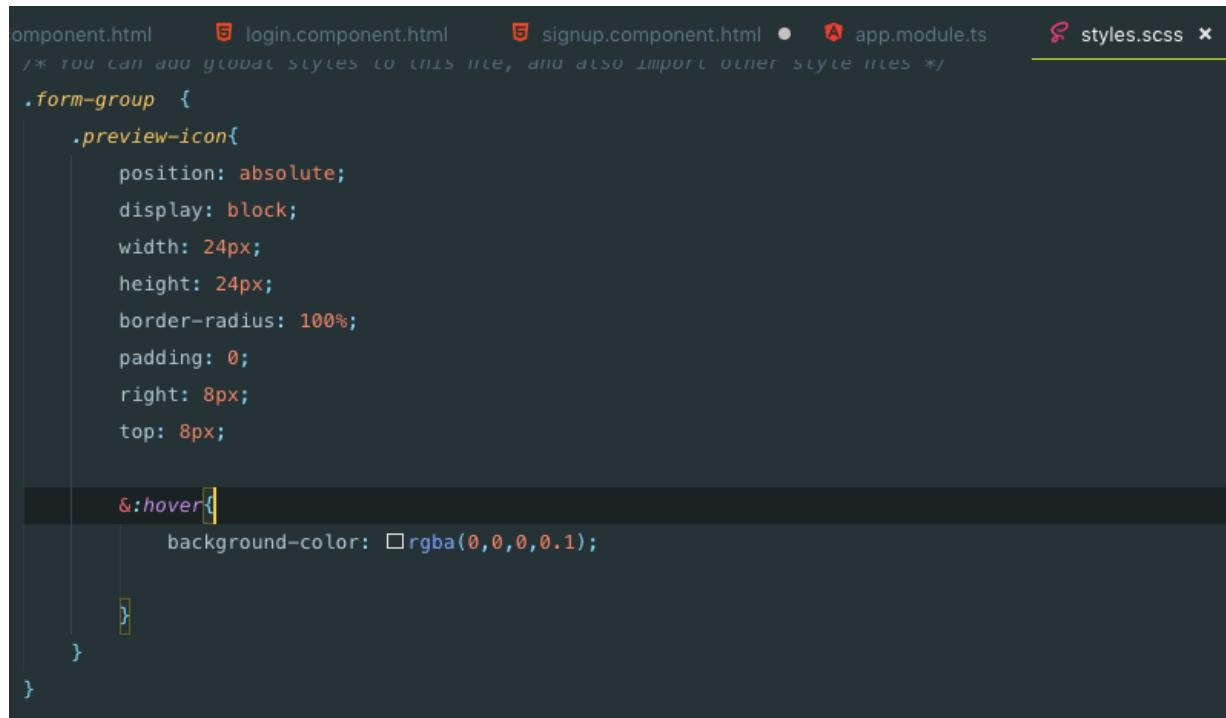
padding: 0; espacio entre el contenido y el borde //esto viene al inspeccionar en el navegador

```
<div class="form-group">
  <label for="exampleInputPassword1">Password</label>
  <input type="password" class="form-control" id="exampleInputPassword1" placeholder="Password">
  <span class="preview-icon">
    <fa-icon [icon]=""></fa-icon>
  </span>
</div>
```

&:hover{} //cuando pase el cursor arriba cambie el estilo

background-color: rgba(0,0,0,0.1);valores del rojo, verde y azul

Vamos a signup.html a la contraseña y abajo de esta pondremos una etiqueta sean con una clase preview-icon



```
/* You can add global styles to this file, and also import other style files */
.form-group {
  .preview-icon{
    position: absolute;
    display: block;
    width: 24px;
    height: 24px;
    border-radius: 100%;
    padding: 0;
    right: 8px;
    top: 8px;

    &:hover{
      background-color: #rgba(0,0,0,0.1);
    }
  }
}
```

```
1 import { BrowserModule } from '@angular/platform-browser';
2 import { NgModule } from '@angular/core';
3 import { FontAwesomeModule } from '@fortawesome/angular-fontawesome';
4
5 import { AppRoutingModule } from './app-routing.module';
6 import { AppComponent } from './app.component';
7 import { LoginComponent } from './login/login.component';
8 import { SignupComponent } from './signup/signup.component';
9
10
11 @NgModule({
12   declarations: [
13     AppComponent,
14     LoginComponent,
15     SignupComponent
16   ],
17   imports: [
18     BrowserModule,
19     AppRoutingModule,
20     FontAwesomeModule
```

An ap modle vamos a meter un constructor

va a cambiar el valor

The screenshot shows a code editor with two tabs: 'signup.component.ts' and 'signup.component.html'. The 'signup.component.ts' tab is active, displaying the following TypeScript code:

```
1 import { Component, OnInit } from '@angular/core';
2
3 @Component({
4   selector: 'app-signup',
5   templateUrl: './signup.component.html',
6   styleUrls: ['./signup.component.scss']
7 })
8 export class SignupComponent implements OnInit {
9   firstStatus: boolean=false;
10  constructor() { }
11
12  ngOnInit() {
13  }
14
15 }
16
```

Cmd d para seleccionar varias

OJITO!

```
<form class="card mt-3">
<div class="card-body">
<div class="form-group">
  <label for="signupEmail">Email address</label>
  <input type="email" class="form-control" id="signupEmail"
placeholder="Enter email">
</div>
<div class="form-group">
  <label for="signupName">Name</label>
  <input type="text" class="form-control" id="signupName"
placeholder="Enter your Name">
</div>
<div class="form-group">
  <label for="signupLastName">Last Name</label>
  <input type="text" class="form-control" id="signupLastName"
placeholder="Enter your Last Name">
  ..
```

```

</div>
<div class="form-group">
  <label for="signupUserName">User-Name</label>
  <input type="text" class="form-control" id="signupUserName"
placeholder="Enter your User-Name">
</div>
<div class="form-group custom-icon">
  <label for="exampleInputPassword1">Password</label>
  <input [type]="passwordPreviews.first ?'password':'text'" class="form-
control" id="exampleInputPassword1"
placeholder="Password">
  <span class="preview-icon" (click)="passwordPreviews.first=!
passwordPreviews.first">
    <fa-icon [icon] ="getIcon(passwordPreviews.first)"></fa-icon>
  </span>
</div>
<div class="form-group custom-icon">
  <label for="exampleInputPassword2">Password</label>
  <input [type]="passwordPreviews.confirm ?'password':'text'" class="form-
control" id="exampleInputPassword2"
placeholder="Password">
  <span class="preview-icon" (click)="passwordPreviews.confirm=!
passwordPreviews.confirm">
    <fa-icon [icon] ="getIcon(passwordPreviews.confirm)"></fa-icon>
  </span>
</div>
<button type="submit" class="btn btn-primary btn-block" >Submit</
button>
</div>
</form>

```

En las Lineas:

<input [type]="passwordPreviews.first ?'password':'text'" class="form-
control" id="exampleInputPassword1" placeholder="Password">

passwordPreviews es un objeto en singup componente ts que tiene tanto first como confirm para la confirmación de el mail siendo booleanos verdaderos ; ahora que el "?" Funciona como un if para cambiar el tipo a password o text dependiendo la bandera del ojito

<span class="preview-icon" (click)="passwordPreviews.first=!
passwordPreviews.first">

En esta linea al dar un click cambia passwordPreviews ya sea first o confirm(al

seleccionar el ojo)

```
<fa-icon [icon]="getIcon(passwordPreviews.first)"></fa-icon>
```

En esta linea obtenemos el icono dependiendo passwordPreviews

Utilizando en singup.component.ts el siguiente código

```
getIcon(value: boolean): string {  
  if (value) {  
    return 'eye';  
  }  
  else {  
    return 'eye-slash';  
  }  
}
```

Cambiamos el código a

```
export class SignupComponent implements OnInit {  
  passwordPreviews = {  
    first: true,  
    confirm: true  
  }  
}
```

Ng g module shared //haremos otro modulo como appmodule aqui pondremos los componentes que podremos utilizar en nuestra aplicación

En app module importamos SharedModule ahora en sharedmodule.ts

```
],  
exports: []
```

En la terminal creamos las carpetas siguientes para obtener

La Alerta

```
MacBook-Air-de-Pamela:todoList pamelaruiz$ ng g component shared/components/alert-modal  
CREATE src/app/shared/components/alert-modal/alert-modal.component.scss (0 bytes)  
CREATE src/app/shared/components/alert-modal/alert-modal.component.html (30 bytes)  
CREATE src/app/shared/components/alert-modal/alert-modal.component.spec.ts (657 bytes)  
CREATE src/app/shared/components/alert-modal/alert-modal.component.ts (289 bytes)  
UPDATE src/app/shared/shared.module.ts (295 bytes)  
MacBook-Air-de-Pamela:todoList pamelaruiz$ █
```

SharedModule

En alert-modal.component.html copiamos el primer modal de la pagina [https://valor-software.com/ngx-bootstrap/#/modals_\(template\)](https://valor-software.com/ngx-bootstrap/#/modals_(template)).

Y en la la 3er linea cambiamos el titulo por {{modalTitle}} ,Y en la antepenultimate linea colocamos un [innerHTML]=""modalBody"

```
ip.component.ts      A alert-modal.component.ts      S alert-modal.component.html x  S app.component.html      A
<ng-template #template>
  <div class="modal-header">
    <h4 class="modal-title pull-left">{{modalTitle}}</h4>
    <button type="button" class="close pull-right" aria-label="Close" (click)="modalRef.hide()">
      <span aria-hidden="true">&times;</span>
    </button>
  </div>
  <div class="modal-body" [innerHTML]="modalBody">
  </div>
</ng-template>
```

En alert-modal.components.ts creamos modalTitle y ModalBody como string y creamos el constructor mostrado

```
A alert-modal.component.ts x  A alert.service.ts      S app.component.html
13   export class AlertModalComponent implements OnInit {
14     @ViewChild('template') template: TemplateRef<any>;
15     modalRef: BsModalRef;
16     modalTitle: string;
17     modalBody: string;
18
19     constructor(
20       private modalService: BsModalService,
21       private alertService: AlertService
22     ) { }
```

Y en shared module.ts:

Escribimos en las importaciones el shared module y al darle click se importa la primera linea.

```
import { SharedModule } from 'src/app/shared/shared.module';

@NgModule({
  declarations: [
    AppComponent,
    LoginComponent,
    SignupComponent,
  ],
  imports: [
    BrowserModule,
    AppRoutingModule,
    FontAwesomeModule,
    SharedModule
  ]
})
```

```
import { AlertModalComponent } from './components/alert-modal/alert-modal.component';
@NgModule({
  declarations: [AlertModalComponent],
  imports: [
    CommonModule
  ],
  exports: [AlertModalComponent]
})
export class SharedModule {}
```

En shared module.ts

Dentro de exports colocamos el alertModalComponent al igual que en las declaraciones y creamos el ModalModule.forRoot();
Y exportamos la clase sharedModule

```
app.component.ts      ⚠ shared.module.ts ✎ app.component.html      ⚠ alert-modal.component.ts
import { AlertModalComponent } from './components/alert-modal/alert-modal.component';
import { ModalModule } from 'ngx-bootstrap/modal';

@NgModule({
  declarations: [AlertModalComponent],
  imports: [
    CommonModule,
    ModalModule.forRoot()
  ],
  exports: [AlertModalComponent]
})
export class SharedModule { }
```

```
import { ModalModule } from 'ngx-bootstrap/modal';
```

```
@NgModule({
  declarations: [AlertModalComponent],
  imports: [
    CommonModule,
    ModalModule
```

En singups.component.ts

Creamos el constructor de alertService y al darle click importamos la primer linea.

```
signup.component.ts ✘ app.component.html ⚡ alert-modal.component.ts ⚡
3   import { AlertService } from 'src/app/shared/services/alert.service';
4
5   @Component({
6     selector: 'app-signup',
7     templateUrl: './signup.component.html',
8     styleUrls: ['./signup.component.scss']
9   })
10  export class SignupComponent implements OnInit {
11    passwordPreviews = {
12      first: true,
13      confirm: true
14    }
15    /* firstStatus: boolean=false;
16    firstStatus2: boolean=false; */
17    passText: string = 'contraseña';
18    constructor(
19      private alertService: AlertService
20    ) {
21    }
22 }
```

Agregamos `(click)="signup($event)` para este evento en el botón

```
signup.component.html ● app.component.html ⚡ alert-modal.component.ts ⚡ alert-modal.component.html ⚡
<button type="submit" class="btn btn-primary btn-block" (click)="signup($event)">Submit</button>
```

Ahora en `signup.components.ts` creamos la siguiente función para mostrar el texto que va dentro del module

```
A signup.component.ts ● A alert-modal.com
33
34     signup(event: Event):void{
35         event.preventDefault();
36         this.alertService.show({
37             title:'Alert!',
38             body:'Are you sure?'
39         });
40     }
```

DUDAS!

```
 1 import { Injectable } from '@angular/core';
 2 import { Subject } from 'rxjs';
 3
 4 @Injectable({
 5   providedIn: 'root'
 6 })
 7 export class AlertService {
 8   alertSubject: Subject<any>;
 9
10   constructor() {
11     this.alertSubject = new Subject();
12   }
13   show(obj: any): void {
14     this.alertSubject.next(obj);
15   }
16 }
17
```

Inyección de dependencias.

Poo
Componentes mañana

```
ngOnInit() {
    this.alertService.alertSubject
        .subscribe((obj) => {
            this.modalTitle = obj.title;
            this.modalBody = obj.body;
            this.openModal(this.template);
        })
}
```

((())=>{}) es una Funcion de flecha

TODOLIST 3/4

Clase:

En **informática**, una **clase** es una plantilla para la creación de **objetos de datos** según un modelo predefinido. Las clases se utilizan para representar entidades o conceptos, como los **sustantivos** en el lenguaje. Cada clase es un modelo que define un conjunto de **variables** -el estado, y **métodos** apropiados para operar con dichos datos -el comportamiento. Cada objeto creado a partir de la clase se denomina **instancia** de la clase.

Ejemplo: CARRO

Abstraccion de un objeto de la vida real, con atributos, propiedades, metodos y acciones.

Un lenguaje para ser OO debe tener

- Herencia (argumentos en clase padre y pasar a clase hija)
- Encapsular(manejar las variables en un ambiente privado para protegerlas (castillo y rey))g)
-

signup



```
class Hijo extends Padre {  
    private nombre: string;  
  
    constructor(nombre:string, piel:string) {  
        super(piel);  
        this.nombre = nombre;  
    }  
}  
"  
" morena");
```

Componentes

/login

Comp
/signup

```
class Padre {  
    protected piel:string;  
    constructor(piel:string){  
        this.piel = piel;  
    }  
}
```

```
class Hijo ex  
private nom
```

```
constructor(ho  
super(pie  
this.nombr
```

```
var nuevoHijo=new Hijo("Juan","morena");
```



Protected : clase padre a clases hijas

Private : misma clase o getters y setters

Public: todas las clases

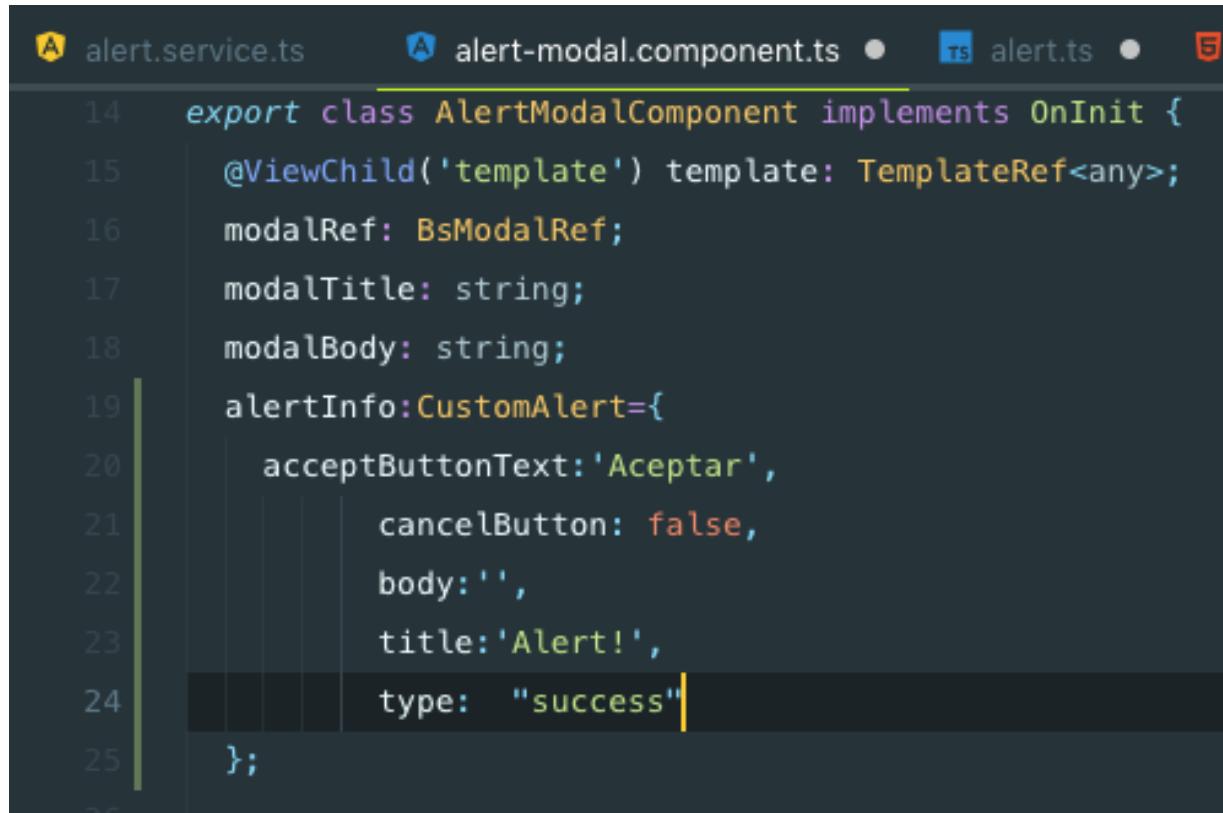
En app creamos una carpeta llamada models y adentro una llamada alert.ts donde colocaremos la información que va a manejar nuestra alerta, colocando el siguiente código

? En ese lugar es para colocarlo opcional

```
1  export interface CustomAlert{  
2      title?:string;  
3      body:string;  
4      cancelButton?: boolean; //propiedad para indicar si existe el boton de cancelar  
5      cancelButtonText?: string; //definir el texto que tendrá el boton cancelar  
6      acceptButtonText?: string; //definir el boton de aceptar  
7      type?: 'success'|'error';  
8  }
```

En typescript tenemos los tipos primitivos: string,number,boolean,buffer(conjunto de información para archivos y cosas media)

Agregamos esta linea



```
14  export class AlertModalComponent implements OnInit {
15    @ViewChild('template') template: TemplateRef<any>;
16    modalRef: BsModalRef;
17    modalTitle: string;
18    modalBody: string;
19    alertInfo:CustomAlert={
20      acceptButtonText:'Aceptar',
21      cancelButton: false,
22      body:'',
23      title:'Alert!',
24      type: "success"
25    };
26
```

- Y cambiamos any por CustomAlert en alert modal component ts y en alert service ts importando en ambos archivos

En el archivo colocamos el código: Lo que tiene el obj lo va a sobreescribir aquí

```
A alert.service.ts      A alert-modal.component.ts ●  S app.componen
24      ) { }
25
26      ngOnInit() {
27          this.alertService.alertSubject
28              .subscribe((obj) => {
29                  /*  this.modalTitle = obj.title;
30                  this.modalBody = obj.body; */
31                  this.alertInfo=Object.assign(<CustomAlert>{
32                      acceptButtonText:'Aceptar',
33                      cancelButton: false,
34                      body:'',
35                      title:'Alert!',
36                      type: "success"
37                  },obj);
38                  this.openModal(this.template);
39          })
40      }

```

Ahora en el html cambiamos las líneas

```
<h4 class="modal-title pull-left">{{alertInfo.title}}</h4>
Y <div class="modal-body" [innerHTML]="alertInfo.body">
```

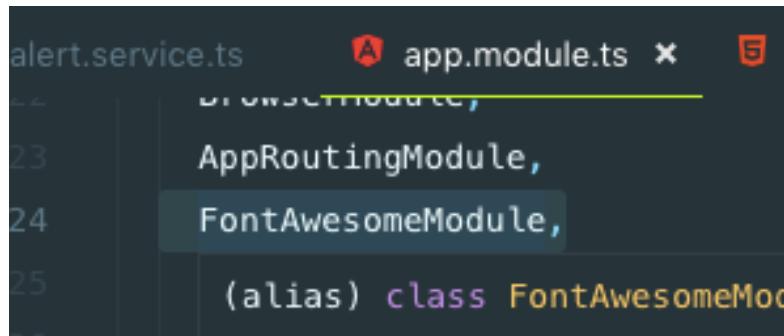
```
A alert.service.ts      S alert-modal.component.html ●  A alert-modal.component.ts      ts alert.ts      A signup.component
3      <ng-template #template>
4          <div class="modal-header">
5              <h4 class="modal-title pull-left">{{alertInfo.title}}</h4>
6              <button type="button" class="close pull-right" aria-label="Close" (click)="modalRef.hide()">
7                  <span aria-hidden="true">&times;</span>
8              </button>
9          </div>
10         <div class="modal-body" [innerHTML]="alertInfo.body">
11             </div>
12     </ng-template>
```

```
7      <span aria-hidden="true">&times;</span>
8    </button>
9  </div>
10 <div class="modal-body" [innerHTML]="alertInfo.body"></div>
11 <div.modal-footer| Emmet Abbreviation ⓘ
12 </ng-template>   ↩ div.modal-footer
13
```

```
0      </div>
1    <div class="modal-body" [innerHTML]="alertInfo.body"></div>
2    <div class="modal-footer text-right">
3      button.btn.btn-danger| Emmet Abbreviation ⓘ
4    </div>
5  </ng-template>
```

```
<div class="modal-body" [innerHTML]="alertInfo.body"></div>
<div class="modal-footer text-right">
  <button class="btn btn-danger" *ngIf="alertInfo.cancelButton">{{alertInfo.cancelButtonText}}</button>
  <button class="btn btn-primary">{{alertInfo.acceptButtonText}}</button>
</div>
</ng-template>
```

Cortamos esta linea y vamos a shared module.ts en import lo pegamos(e importamos)



```
alert.service.ts      A app.module.ts ×  ⌂
  ...
  23      AppRoutingModule,
  24      FontAwesomeModule,  ⌂
  25        (alias) class FontAwesomeMod
```

De igual forma con esta linea pero la pegamos en SharedModule con todo y el constructor , NO EN IMPORT

```
alert.service.ts      A app.module.ts ●      A shared.module.ts ●      S alert-mo
29   ...
30   export class AppModule {
31     constructor(){
32       library.add(fas); //para agregar un paquete de iconos
33     }
34 }
```

y con estas dos líneas se pegan en sharedModule.ts en la parte superior donde se encuentran todos los import

```
A alert.service.ts      A app.module.ts ●      A shared.module.ts ●      S alert-modal.
4   import {library} from '@fortawesome/fontawesome-svg-core';
5   import {fas} from '@fortawesome/free-solid-svg-icons';
6 
```

Ahora agregamos

```
alert.service.ts      A app.module.ts      A shared.module.ts ●      S
15   ],
16   exports: [AlertModalComponent, FontAwesomeModule]
17 }
```

En app module ts agregamos en imports ReactiveFormsModule, (PARA HACER LOS FORMULARIOS MAS ESTRUCTURADOS)

En singup component ts agregamos form:FormGroup bajo exports y private fb: FormBuilder

En el constructor

En el import de apps module.ts agregamos FormsModule,

Ahora realizaremos las validaciones en signup componentes ts

```
30     buildForm() {
31         this.signupForm = this.fb.group({
32             name:[ '', [Validators.required]], //campo será requerido
33             lastname:[ '', [Validators.required]],
34             lastname2:[ '', [Validators.required]],
35             email:[ '', [Validators.required,Validators.email]],
36             username:[ '', [Validators.required]]]
37         });
38     };

```

En el input de cada uno en el html colocamos `FormControlName="firstName"` por ejemplo dentro de la etiqueta y tenemos que revisar que concuerden con los escritos en `signupcomponent`

Pegamos el anuncio amarillo de <https://getbootstrap.com/docs/4.3/components/alerts/>

Quedando de la siguiente forma

```
<input type="email" class="form-control" id="signupEmail" placeholder="Enter email" formControlName="email">
<div class="alert alert-warning" role="alert" *ngIf="signupForm.get('email').hasError('email')">
    A simple warning alert—check it out!
</div>
</div>
```

Validación en cada una de los campos con un *

Ahora entramos a

<https://stackoverflow.com/questions/46155/how-to-validate-an-email-address-in-javascript>

Y copiamos `/^(([^<>()\\[\\]\\.,;:\\s@"]+\\(.[^<>()\\[\\]\\.,;:\\s@"]+\\))*|(".+"))@((\\[[0-9]{1,3}\\.[0-9]{1,3}\\.[0-9]{1,3}\\.[0-9]{1,3}\\])|(([a-zA-Z\\-0-9]+\\.)+[a-zA-Z]{2,}))$/`

Y lo pegamos aqui

The screenshot shows a code editor with three tabs at the top: 'app.module.ts', 'signup.component.ts' (which is the active tab), and 'signup.component.html'. The code in 'signup.component.ts' is as follows:

```
32 buildForm() {  
33     this.signupForm = this.fb.group({  
34         firstName: ['', [Validators.required]], //campo será requerido  
35         lastName: ['', [Validators.required]],  
36         secondLastName: ['', [Validators.required]],  
37         email: ['', [Validators.required, Validators.email, Validators.pattern(/^((([^<>()\\[]\\.,;:\\s@"])+(\.["^<>()\\[]\\.,;:\\s@"])+)/)]],  
38         username: ['', [Validators.required]]  
39     });  
40 }
```

Y ahora tendremos:

```
app.module.ts  ✘ signup.component.ts  ●  ✎ signup.component.html  ●  
16 |     <div class="alert alert-warning" role="alert" *ngIf="signupForm.get('email').hasError('pattern')">  
17 |       Please insert a valid email!  
18 |     </div>
```

Esto para validar el correo con un pattern de mejor forma

Const let y var para hacer variables

Ng serve —host 0.0.0.0

TODOLIST 4/4

Programación asincrona

Procesos que se ejecutan simultáneamente pero con un final a diferencia de el hilo que le tienes que poner el final puesto que continuara

Func 1 y Func 2

Atender las peticiones simultáneamente o en paralelo

Promesas o funciones Callback

- Para acceder a las propiedades de un objeto
ObjVar['propiedad1'] o ObjVar.propiedad1

POST https://apichidalida.tk/api/v1/auth/signup

Body (JSON application/json)

```
{
  "message": "Missing email field",
  "code": 1000,
  "status": 400
}
```

AuthController

- POST** /api/v1/auth/login
- POST** /api/v1/auth/signup
- GET** /api/v1/auth/user
- GET** /api/v1/auth/user/username
- GET** /api/v1/auth/user/email

<https://apichidalida.tk/api-docs/#/AuthController/AuthController.findEmail>

Screenshot of the Postman application interface showing a request to https://apichidalida.tk/api/v1/auth/user/email?valoremail.

The request method is GET, and the URL is https://apichidalida.tk/api/v1/auth/user/email?valoremail. The Body tab is selected, showing a JSON payload:

```
1 {  
2   "firstName": "Alejandro",  
3   "lastName": "Villarroel",  
4   "secondLastName": "Colderon",  
5   "username": "alexvanhell",  
6   "email": "e@mail.com",  
7   "password": "contraseña"  
8 }
```

The response status is 400 Bad Request, with a message: "Missing email field", code: 1000, and status: 400.