1.- ng new hacienda\_buzon\_tributario\_web

2.- cd hacienda\_buzon\_tributario\_web

3.- ng g c auth/layout --skip-tests

4.- ng g c auth/pages/login --skip-tests

5.- ng g c auth/pages/register --skip-tests

6.- Controlados que va a fungir con enlace entre las diferentes paginas

ng g c dashboard/layout/layoutDash --skip-tests –flat

import { RouterModule } from '@angular/router'; -> (manejo de rutas)

\*\*\*.html -> agregamos el router-outlet “Seguirá creciendo”

<router-outlet>

7,. Manejo de estilos

ng add @angular/material

npm install primeflex

angular.json

styles: [

…

"./node\_modules/primeflex/primeflex.css"

]

8.- Manejo de errores

npm install sweetalert2

9.- Manejo de pagina no encontrada

ng g c dashboard/pages/notFoundPage --skip-tests --skip-selecto

10.- Configuracion para acortar los paths relativos en tsconfig.json

{

…

"compilerOptions": {

…

"paths": {

      "@shared/\*": ["./src/app/dashboard/\*"],

      "@auth/\*":   ["./src/app/auth/\*"]

"@environments/\*": ["./src/environments/\*"]

    }

}

}

Ejemplo de aplicación import { SidemenuComponent } from '@shared/sidemenu/sidemenu.component';

11.- app.router.ts -> genweramos las rutas

export const routes*: Routes* = [

  {

    path: 'auth',

    //canActivate: [isNotAuthenticatedGuard],

    loadComponent: *()* => import('./auth/layout/layout.component').then(c => c.LayoutComponent),

    children: [

      {

        path: 'login',

        title: 'Login',

        loadComponent: *()* => import('./auth/pages/login/login.component').then(c=> c.LoginComponent)

      },

      {

        path: '',

        redirectTo: 'login',

        pathMatch: 'full'

      }

    ]

  },

  {

    path: 'dashboard',

    //canActivate: [isAuthenticatedGuard],

    loadComponent: *()* => import('./dashboard/layout/layout-dash.component').then(c => c.LayoutDashComponent),

    children: [

      /\*{

        path: 'refrendo/:idConcepto/:formulario',

        title: 'Refrendo Vehicular',

        loadComponent: () => import('./portal-servicios/smyt/pages/refrendo-page/refrendo-page.component'),

      },\*/

…

    ]

  },

  {

    path: '',

    redirectTo: 'auth',

    pathMatch: 'full'

  },

  {

    path: '\*\*',

    loadComponent: *()* => import('./dashboard/pages/not-found-page/not-found-page.component').then(c => c.NotFoundPageComponent)

  }

];

12.- App.component.htm -> borramos todo solo dejamos el <router-outlet>

-> 4.- Agregamos código html, librerías necesarios y creamos el formulario

-> 3.- Agregamos código html, RouterModule y Css

13.- Crear la carpeta en src/environments, manejo de variables globales. Creamos dos archivos para manejo de variables en pruebas y en producción

environments.ts y environments.prod.ts

Editamos angular.json

“projects”[

“hacienda\_buzon…”{

“architect”: {

build: {

…

"configurations": {

            "production": {

              "fileReplacements": [{"replace": "src/environments/environments.ts", "with": "src/environments/environments.prod.ts"}],

…

}

}

}

}

]

14.- app.config.ts -> inyectar HttpClientModule e importarlo -> manejo global de peticiones

15.- ng g s auth/services/authService ->

>ng g s auth/services/authService --skip-tests

inyectar HttpClient

16.- Crear las interfaces auth/interfaces/login-respose-struct.interface.ts y auth/interfaces/user-struct.interface.ts

>ng g i auth/interfaces/loginResponseStruct

>ng g i auth/interfaces/userStruct

>ng g i auth/interfaces/authStatusStruct -> maneja la estructura de logeado, no logeado, etc. Es un ENUM

17.- En el paso 14. Agregamos el método de login y agregamos toda la funcionalidad del servico para manejar el registro

Login recibirá parámetros de tipo LoginRequestStruct

Generamos la interfaz auth/interfaces/login-request-struct

> ng g i auth/interfaces/loginRequestStruct

18.- Paso 4.- inyectamos AuthService implementamos el método login y hacemos el llamado al servico de logeo

Inyectamos Routes para el manejo de rutas

Inyectamos Swal para las notificaciones a los usuarios

19.- Cremos un servicio de validaciones /auth/services/validationService -> Su función es validar dos campos y eliminar la validación del vacío

>ng g s auth/services/validation --skip-tests

En el paso 18.- inyectomos validationService