Modifiquemos el proyecto

La actividad que has subido trata sobre la modificación de un proyecto llamado SkeletonAPP utilizando el framework Ionic para crear una aplicación móvil híbrida. A continuación, te proporciono un paso a paso detallado junto con el código correspondiente, explicando cada línea de código con comentarios.

Paso 1: Configuración del Proyecto

1. **Ruta y Nombre del Proyecto:**
   - Crea una carpeta para tu proyecto, por ejemplo: `C:\\IonicProjects\\SkeletonAPP`.
   - Navega a esa carpeta en tu terminal.
2. **Crear un Nuevo Proyecto en Ionic:**

   ```
   ionic start SkeletonAPP blank --type=angular
   ```

   - Este comando crea un nuevo proyecto de Ionic con Angular en una plantilla en blanco.
3. **Navegar al Proyecto:**

   ```
   cd SkeletonAPP
   ```

Paso 2: Crear y Modificar las Páginas `Login` y Inicio

1. **Crear la Página `Login`:**

   ```
   ionic generate page login
   ```

   - Este comando crea una nueva página llamada `login`.
2. **Crear la Página `Home`:**

```
ionic generate page inicio
```

- Actualizar el archivo app-routing.module.ts

```
import { NgModule } from '@angular/core';
import { PreloadAllModules, RouterModule, Routes } from '@ang

const routes: Routes = [
  {
    path: '',//######
    redirectTo: 'login',//####
    pathMatch: 'full'//#####
  },
  {
    path: 'login',
    loadChildren: () => import('./login/login.module').then(
  },
  {
    path: 'inicio',
    loadChildren: () => import('./inicio/inicio.module').then
  },
];

@NgModule({
  imports: [
    RouterModule.forRoot(routes, { preloadingStrategy: Preloa
  ],
  exports: [RouterModule]
})
export class AppRoutingModule { }
```

```
import { NgModule } from '@angular/core';
import { PreloadAllModules, RouterModule, Routes } from '@ang
```

```
/*
  Aquí se importan los módulos necesarios para configurar el
  - `NgModule` es el decorador que define un módulo de Angula
  - `PreloadAllModules` es una estrategia de precarga que car
  - `RouterModule` es el módulo que contiene las directivas d
  - `Routes` es una interfaz que define cómo se deben configu
*/

const routes: Routes = [
  {
    path: '',
    // Cuando el camino (path) es una cadena vacía, significa
    redirectTo: 'login',
    // Se redirige automáticamente a la ruta 'login'.
    pathMatch: 'full'
    // `pathMatch: 'full'` indica que la redirección debe ocu
  },
  {
    path: 'login',
    // Esta es la ruta para la página de login. Cuando la URL
    loadChildren: () => import('./login/login.module').then(m
    // `loadChildren` es utilizado para cargar perezosamente
    // `import('./login/login.module')` importa dinámicamente
    // `.then(m => m.LoginPageModule)` asegura que se cargue
  },
  {
    path: 'inicio',
    // Esta es la ruta para la página de inicio. Cuando la UR
    loadChildren: () => import('./inicio/inicio.module').then
    // Similar a la ruta anterior, esta configuración carga p
  },
];

/*
  Estas son las rutas definidas para la aplicación.
  - La primera ruta redirige a la página de login si la URL e
```

```
  - Las otras rutas cargan perezosamente los módulos de logi
*/

@NgModule({
  imports: [
    RouterModule.forRoot(routes, { preloadingStrategy: Preloa
    // `RouterModule.forRoot(routes)` configura las rutas pri
    // `{ preloadingStrategy: PreloadAllModules }` indica que
  ],
  exports: [RouterModule]
  // `exports: [RouterModule]` permite que el módulo de enrut
})
export class AppRoutingModule { }
// Esta es la clase del módulo de enrutamiento principal de
```

3. **Modificar la Página `Login` (src/app/login/login.page.html):**
   - **Archivo:** `src/app/login/login.page.html`
   - **Código:**

```
<ion-header [translucent]="true">
  <ion-toolbar>
    <ion-title>Login</ion-title>
  </ion-toolbar>
</ion-header>
<ion-content [fullscreen]="true">
  <ion-item>
    <!--"fixed" | "floating" | "stacked"-->
    <ion-label color="success" position="floating">Usua
rio</ion-label>
      <!--ngModel enlaza la propiedad 'username' del co
mponente TypeScript con el valor del campo de entrada.
es un enlace de tipo bidireccional -->
    <ion-input [(ngModel)]="username" maxlength="8" min
```

```
length="3" required></ion-input>
 </ion-item>
 <ion-item>
  <ion-label position="floating">Contraseña</ion-label>
Vinculación bidireccional entre el input y la propiedad
de password del componente de angular
  <ion-input [(ngModel)]="password" type="password" max
length="4" required></ion-input>
 </ion-item>
 <ion-button expand="full" (click)="login()">ingresar</
ion-button>
</ion-content>
```

-

4. **Modificar el Archivo de la Página `Login` (src/app/login/login.page.ts):**
   - **Archivo:** `src/app/login/login.page.ts`
   - **Código:**

```
import { Component, OnInit } from '@angular/core';
import { Router } from '@angular/router';//##########
//definimos el componente de Angular
@Component({
  selector: 'app-login',//nombre
  templateUrl: './login.page.html',//La url del templat
e HTML
  styleUrls: ['./login.page.scss'],//La URL de los esti
los de la página
})
export class LoginPage implements OnInit {

  //Almacenamos el nombre del usuario y el password en
una variable
  username: string = '';
  password: string = '';
```

```
  constructor(private router:Router) { }
  /*
    El constructor de la clase recibe una instancia de `Router` como un parámetro inyectado.
    - `private router: Router` define una propiedad privada `router` que se inicializa con la instancia del `Router` inyectado.
    - Esta inyección permite a la clase utilizar el `Router` para realizar la navegación programática dentro de la aplicación.
  */
  login(){
    /*== (igualdad débil): Compara los valores después de convertirlos al mismo tipo si es necesario (coerción de tipo). Por ejemplo, 5 == "5" es true porque el string "5" se convierte en número antes de comparar.
=== (igualdad estricta): Compara tanto el valor como el tipo sin realizar ninguna conversión. Por ejemplo, 5 === "5" es false porque un número y un string no son del mismo tipo.*/
    //Verificamos que el usuario y la contraseña cumplan los requisitos
    if (this.username.length >=3 && this.username.length <=8 && this.password.length ===4){
      //Navega a la página de inicio si es necesario
      this.router.navigate(['/inicio'],{
        queryParams: {username: this.username}
      });
      /*
        Si las validaciones son exitosas, la aplicación navega a la página de inicio (`/inicio`).
        - `this.router.navigate(['/inicio'], { queryParams: { username: this.username } });` usa el `Router` para navegar a la ruta `/inicio`.
        - `queryParams: { username: this.username }` pasa el nombre de usuario como un parámetro de consulta e
```

```
n la URL. Esto permite que la siguiente página acceda a
l nombre de usuario si lo necesita.
      */
    }else{
      //Muestra un mensaje de error si las condiciones
no se cumplen
      alert ('Usuario o contraseña invalidos')
    }
  }
  ngOnInit() {
  }

}
```

5. **Modificar la Página `Inicio` (src/app/inicio/inicio.page.html):**
   - **Archivo:** `src/app/inicio/inicio.page.html`
   - **Código:**

```
<ion-header>
  <ion-toolbar>
    <ion-title>Inicio</ion-title>
  </ion-toolbar>
</ion-header>

<ion-content>
  <ion-card>
    <ion-card-header>
      <ion-card-title>Bienvenido, {{ username }}</ion-c
ard-title>
    </ion-card-header>
    <ion-card-content>
      <ion-item>
        <ion-label position="floating">Nombre</ion-labe
```

```
l>
        <ion-input [(ngModel)]="firstName"></ion-input>
      </ion-item>
      <ion-item>
        <ion-label position="floating">Apellido</ion-la
bel>
        <ion-input [(ngModel)]="lastName"></ion-input>
      </ion-item>
      <ion-item>
        <ion-label position="floating">Nivel de Educaci
ón</ion-label>
        <ion-input [(ngModel)]="educationLevel"></ion-i
nput>
      </ion-item>
      <ion-item>
        <ion-label position="floating">Fecha de Nacimie
nto</ion-label>
        <ion-input [(ngModel)]="birthDate" type="date">
</ion-input>
      </ion-item>
    </ion-card-content>
  </ion-card>

  <ion-button expand="full" (click)="clear()">Limpiar</
ion-button>
  <ion-button expand="full" (click)="show()">Mostrar</i
on-button>
</ion-content>
```

- **Explicación:**
  - `{{ username }}` muestra el nombre de usuario ingresado en el login.
  - Los campos de entrada permiten al usuario ingresar su información adicional.

6. **Modificar el Archivo de la Página `Inicio` (src/app/inicio/inicio.page.ts):**

- **Archivo:** `src/app/inicio/inicio.page.ts`
- **Código:**

```
import { Component, OnInit } from '@angular/core';
import { ActivatedRoute } from '@angular/router';

@Component({
  selector: 'app-inicio
   ',
  templateUrl: './inicio.page.html',
  styleUrls: ['./inicio.page.scss'],
})
export class HomePage implements OnInit {

  username: string = '';
  firstName: string = '';
  lastName: string = '';
  educationLevel: string = '';
  birthDate: string = '';

  constructor(private route: ActivatedRoute) {}

  ngOnInit() {
    // Recupera el nombre de usuario desde los parámetr
os de la URL
    this.route.queryParams.subscribe(params => {
      if (params['username']) {
        this.username = params['username'];
      }
    });
  }

  clear() {
    // Limpia todos los campos del formulario
    this.firstName = '';
```

```
    this.lastName = '';
    this.educationLevel = '';
    this.birthDate = '';
  }

  show() {
    // Muestra el nombre y apellido en un mensaje emerg
ente
    alert(`Nombre: ${this.firstName} \n Apellido: ${thi
s.lastName}`);
  }
}
```

- `ngOnInit()` recupera el nombre de usuario desde la página `Login`.
- `clear()` limpia los campos del formulario.
- `show()` muestra un mensaje con el nombre y apellido ingresados.