



UNIVERSIDAD
NACIONAL
AUTÓNOMA DE
NICARAGUA,
MANAGUA

UNAN - MANAGUA

Guía de Aprendizaje

Desarrollo de Aplicaciones Móviles

Msc. Luis Manuel Espinoza Estrada

Introducción

El proyecto consiste en desarrollar una agenda telefonica que permita crear, mostrar, eliminar y listar a los usuarios de la agenda.

A continuación se describen los requerimientos de la aplicación.

1. Mostrar la lista con los usuarios agregados en la agenda.
2. Buscar un usuario dentro de la lista de la agenda.
3. Agregar usuarios a la agenda.
4. Modificar los usuarios de la agenda
5. Eliminar los usuarios de la agenda.

La pantalla de inicio, deberá mostrar la lista con los usuarios agregados a la agenda, la segunda pantalla deberá agregar un usuario con su número telefonico.

Cabe destacar que en este proyecto solamente se podrán agregar los datos a un arreglo, existen otros medios de almacenamientos que permiten almacenar los datos de forma persistente, pero por el tiempo del taller y el nivel que exige dicho forma de trabajo, se utilizará el método antes mencionado.

Creando un proyecto en Ionic

Para crear un proyecto en Ionic es necesario utilizar el siguiente comando

```
ionic start Congreso blank
```

Donde **Congreso** será el nombre de la carpeta que tendrá todos los archivos necesarios para crear la aplicación móvil.

Nota: Es necesario conocer la ubicación de la carpeta donde se esta creando el proyecto.

Para validar la correcta descarga de los paquetes, es necesario ejecutar el proyecto utilizando el comando

```
ionic serve
```

Se deberá mostrar el proyecto en un navegador web.

Agregar Bootstrap a la aplicación

Entre a la siguiente URL:

<https://getbootstrap.com/>

copie el CSS only

BootstrapCDN

When you only need to include Bootstrap's compiled CSS or JS, you can use [BootstrapCDN](#).

CSS only

```
<link rel="stylesheet" href="https://
```

Pegue el link dentro del head del archivo index ubicado en el src.

Ejercicio 1

Agregar usuarios

Para agregar un usuario, se creará una ventana modal que permita almacenar los datos en un arreglo. Para crear un componente que funcionará como ventana modal es necesario utilizar el siguiente comando. Es necesario estar ubicado dentro de la carpeta del proyecto para que se pueda crear el componente.

ionic generate component **agregar**

Para utilizar el componente **agregar** es necesario importar el componente en el archivo app.module.ts

```
import { AgregarComponent } from './agregar/agregar.component';
```

En el @NgModule, agregar en las declarations el **AgregarComponent** y entryComponents AgregarComponent. Deberá tener el siguiente aspecto

```
@NgModule({  
  declarations: [AppComponent, AgregarComponent],  
  entryComponents: [AgregarComponent],  
})
```

Componente como ventana modal

El siguiente enlace muestra como crear una venta modal.

<https://ionicframework.com/docs/api/modal>

En el archivo que se encuentra ubicado dentro de la carpeta src/app/home/home.page.ts deberá importar el componente que funcionará como ventana modal

```
import { AgregarComponent } from '../agregar/agregar.component';
```

Además, es necesario importar el la clase ModalController

```
import { ModalController } from '@ionic/angular';
```

Luego, es necesario crear un objeto de la clase en el constructor de la clase HomePage

```
constructor(private Modal : ModalController) {}
```

Después, es necesario crear un método que permita mostrar el modal.

```
async AgregarModal() {  
  const modal = await this.Modal.create({  
    component: AgregarComponent  
  });  
  return await modal.present();  
}
```

En el archivo ubicado en src/app/home/home.page.html es necesario agregar un mecanismo que permita llamar al método **AgregarModal** creado anteriormente. En este caso se creará un botón que permita llamar al método, este botón estará ubicado en la parte derecha del **toolbar**. Puede ubicar el siguiente código debajo del **ion-title**

```
<ion-buttons slot="end">  
  <ion-button color="primary" (click) = "AgregarModal()" >Agregar</ion-button>  
</ion-buttons>
```

En el botón, es necesario que agregue el evento click y se llame el método AgregarModal() que permite mostrar el componente creado anteriormente.

A continuación se muestra como debería ir quedando la aplicación.

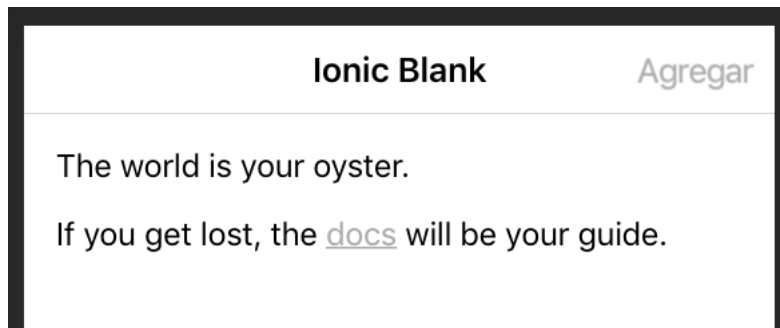


Ilustración 1 – Inicio



Ilustración 2 - Ventana Modal

Modificando el componente Agregar

1. Agregar un header, toolbar y un botón para cerrar el modal

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

    <ion-buttons slot="end">
      <ion-button color="primary">x</ion-button>
    </ion-buttons>
  </ion-toolbar>
</ion-header>
```

2. Agregar un formulario

```

<ion-content>
  <form>
    <div class="form-group">
      <label for="nombre">Nombre: </label>
      <input type="text" class="form-control" id="nombre" placeholder="Ingrese el nombre">
    </div>
    <div class="form-group">
      <label for="numero">Número</label>
      <input type="number" class="form-control" id="numero" placeholder="Ingrese el número">
    </div>

    <ion-button color="primary">Agregar</ion-button>
  </form>
</ion-content>

```

Agregar el FormsModule al proyecto.

Esto se deberá hacer en el archivo src/app/app.module.ts

```
import { FormsModule } from '@angular/forms';
```

Luego se agrega la clase dentro del ngModel, específicamente en el arreglo correspondiente a los imports. Observe la siguiente imagen e identifique donde está ubicado el **FormsModule**

```

@NgModule({
  declarations: [AppComponent, AgregarComponent],
  entryComponents: [AgregarComponent],
  imports: [BrowserModule, IonicModule.forRoot(), AppRoutingModule, FormsModule],
})

```

Se modifica el formulario antes creado y se agrega el ngModel para conectar las variables de la clase AgregarComponent con los inputs del formulario.

```
<input type="text" [(ngModel)] = "nombre" class="form-control" id="nombre" name="nombre" placeholder="Ingrese el nombre">
```

Realice el mismo procedimiento al input del número

Agregar los atributos a la clase AgregarComponent

```

nombre: string;
numero: number;

```

Crear un servicio dentro de la carpeta servicios.

Para crear un servicio en Ionic es necesario utilizar el siguiente comando, además es necesario ubicar el servicio dentro de la carpeta servicio (puede variar el nombre), con el fin de tener un orden en nuestra estructura de carpetas.

```
ionic generate service servicios/agenda
```

Dentro del servicio, específicamente debajo de los imports, es necesario crear una interfaz que permita mantener una estructura de los datos almacenados en el arreglo.

```
interface Datos {  
  nombre: string;  
  numero: number;  
}
```

Luego, cree un atributo dentro de la clase **AgendaService** llamado **arreglo** donde deberá cumplir con la siguientes propiedades.

1. Que sea de tipo arreglo
2. Que defina una estructura de objetos (nombre, numero)

```
arreglo: Array<{nombre: string, numero: number}> = [];
```

Ahora, es necesario crear un método que permita agregar los datos al arreglo. El método recibirá dos parámetros, el nombre y el numero. Para agregar valores al arreglo, es necesario utilizar el método push del objeto Array.

```
agregar(nombre: string, numero: number)  
{  
  this.arreglo.push({nombre, numero});  
}
```

Para terminar la clase **AgendaService** (por el momento) se crea un método que permita obtener el arreglo. Ese método se llamará **getArreglo**

```
getArreglo()  
{  
  return this.arreglo;  
}
```

En la clase **AgregarComponent** específicamente en el constructor será necesario crear un objeto del servicio

```
constructor(public agendaService: AgendaService) {}
```

Es de mucha importancia estar consciente de que si se utilizará una clase fuera del archivo, es necesario importar dicha clase

```
import { AgendaService } from '../servicios/agenda.service';
```

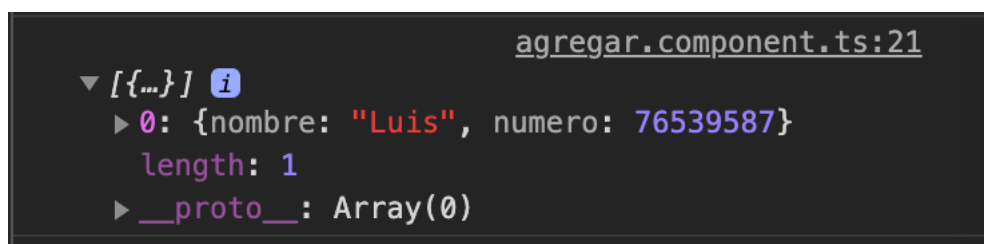
Luego, se deberá crear un método llamado **agregarDatos** y se deberá mandar a llamar al método **agregar** del servicio. Puede utilizar el método **log** de la clase **console** y el método **getArreglo** del servicio para observar los cambios en el navegador.

```
agregarDatos()  
{  
  this.agendaService.agregar(this.nombre, this.numero);  
  console.log(this.agendaService.getArreglo());  
}
```



A screenshot of a web form titled "Agregar" with a close button "x" in the top right corner. The form has two input fields: "Nombre:" with the value "Luis" and "Número" with the value "76539587". Below the fields is a button labeled "Agregar".

Ilustración 3 - Ventana de agregar



```
agregar.component.ts:21  
▼ [{...}] ⓘ  
  ► 0: {nombre: "Luis", numero: 76539587}  
    length: 1  
  ► __proto__: Array(0)
```

Ilustración 4 - Consola del navegador

Ejercicio 2

Lista de contactos

En el archivo [home.page.html](#) ubicado en el directorio `src/app/home` se deberá crear una estructura que permita mostrar una lista de elementos.

Dentro del `ion-content` crea una lista con la etiqueta `ion-list` y `ion-item`. (Ejemplo de lista en Ionic: <https://ionicframework.com/docs/api/list>)

```
<ion-list>
  <ion-item>
    Nombre
    <div class="item-note" slot="end"> Numero </div>
  </ion-item>
</ion-list>
```

Ilustración 5 - Lista de elementos

En la aplicación deberán tener algo parecido a la siguiente imagen.

Ionic Blank		Agregar
Nombre	Numero	



Reto: Ya es momento de cambiar el título de la aplicación, identifique en el archivo [home.page.html](#) donde se encuentra ubicado el título y cambielo por “Agenda de contacto”

Mostrando datos

En el archivo `home.page.ts`

1. Se importa la el servicio

```
import { AgendaService } from '../servicios/agenda.service';
```

2. Se crea un objeto de la clase `AgendaService` en el constructor

```
constructor(private Modal : ModalController, private agendaService: AgendaService) {}
```

3. Antes del constructor se crea un atributo llamado contactos y se asigna un arreglo en vacío.

```
contactos = [];
```

4. En el constructor se asigna el arreglo que retorna el método getArreglo del servicio al arreglo contactos.

```
constructor(private Modal : ModalController, private agendaService: AgendaService) {  
    this.contactos = agendaService.getArreglo();  
}
```

5. En el archivo [home.page.html](#) se deberá modificar la lista antes creada.
 - a. Es necesario utilizar la directiva *ngFor para recorrer el arreglo creado en el archivo [home.page.ts](#)

```
<ion-item *ngFor="let item of contactos" >
```

- b. Luego es necesario mostrar interpolar los valores obtenidos al recorrer el arreglo

```
{{item.nombre}}
```

- c. Al final deberá tener algo similar a la siguiente imagen

```
<ion-list>  
    <ion-item *ngFor="let item of contactos" >  
        {{item.nombre}}  
        <div class="item-note" slot="end"> {{item.numero}} </div>  
    </ion-item>  
</ion-list>
```



Es momento de probar la aplicación

Agregar

X

Nombre:

Luis

Número

76539587

AGREGAR

Ilustración 6 - Ventana de agregar

Agenda de contactos		AGREGAR
Luis	76539587	

Ilustración 7 - Lista de contactos

Ejercicio 3

Eliminar contactos

Antes de eliminar los contactos, es necesario cambiar la lista creada anteriormente, se deberá ver como la siguiente

```

<ion-list>
  <ion-item-sliding *ngFor="let item of contactos; let i = index">
    <ion-item>
      <ion-label>{{item.nombre}}</ion-label>
      <div class="item-note" slot="end"> {{item.numero}} </div>
    </ion-item>

    <ion-item-options side="end">
      <ion-item-option color="danger" (click) = "eliminar[{{item.numero}}]">Eliminar</ion-item-option>
    </ion-item-options>
  </ion-item-sliding>
</ion-list>

```

Se agregan nuevos elementos: ion-item-sliding, ion-item-options y ion-item-option. Estos elementos permitirá desplazar la lista para mostrar un botón que de forma visible se ve un poco mas estetico.

La siguiente imagen muestra el resultado

Agenda de contactos		Agregar
76529587		Eliminar

Luego en el servicio: AgendaService se deberá crear un método que permita eliminar los elementos del arreglo. Este método recibe un parámetro, ese parámetro será el número de teléfono del contacto. Se utiliza el método filter que permite filtrar los elementos dependiendo de la condición. En este caso devuelve los elementos que sean diferentes al parámetro.

```
eliminar(item: number)
{
  this.arreglo = this.arreglo.filter(
    (i) => i.numero !== item );
}
```

Para finalizar es necesario llamar a ese método en el archivo [home.page.ts](#) y asignar nuevamente el arreglo a la variable **contactos**

```
eliminar(item: number)
{
  this.agendaService.eliminar(item);
  this.contactos = this.agendaService.getArreglo();
}
```



Prueba la aplicación

A continuación dejo el repositorio donde se puede encontrar el proyecto completo

<https://github.com/lespinozae1989/Congreso-Agenda>