Ejemplo práctico de Ionic
Aplicación para visualizar recetas

Repositorio: https://github.com/ManuCastrillonM/Ionic-example/
*Consejo: Cada paso del desarrollo está especificado en una rama del proyecto, puedes navegar al paso que desees utilizando: git checkout [número-del-paso]

Pre-requisitos:

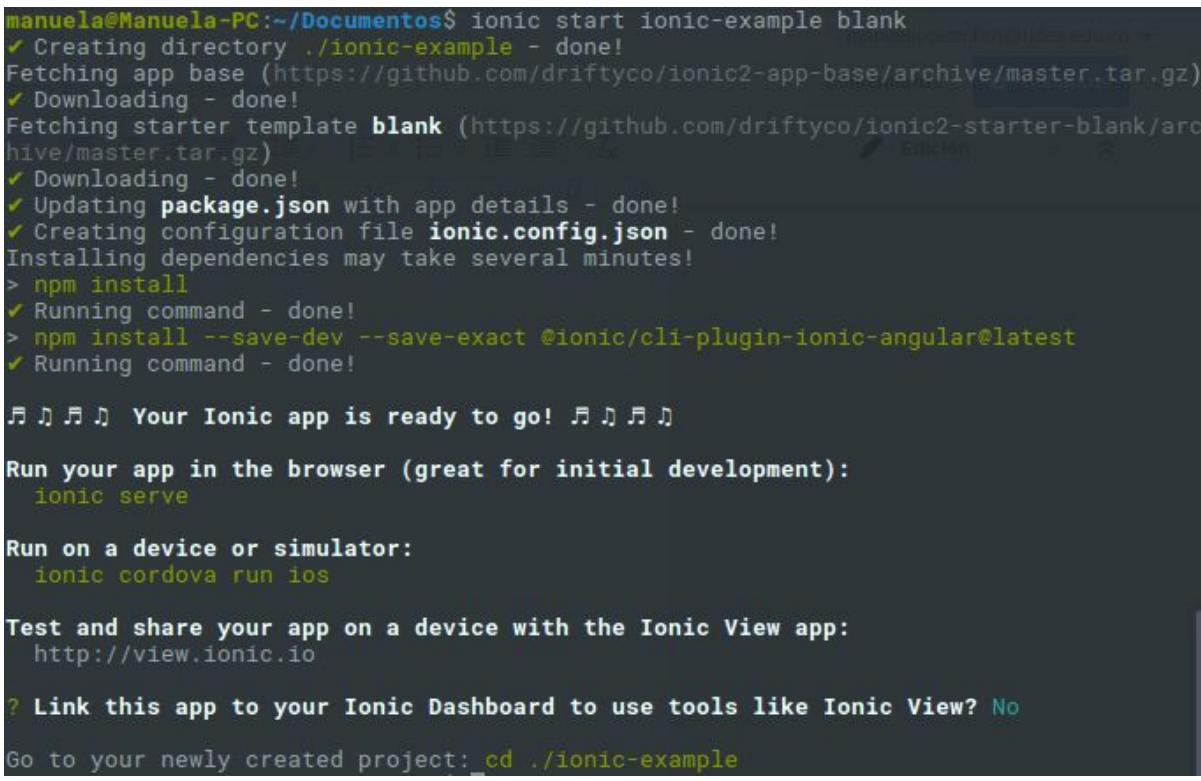
- NodeJS
- Npm

Instalación:

> npm install -g cordova ionic

Desarrollo:

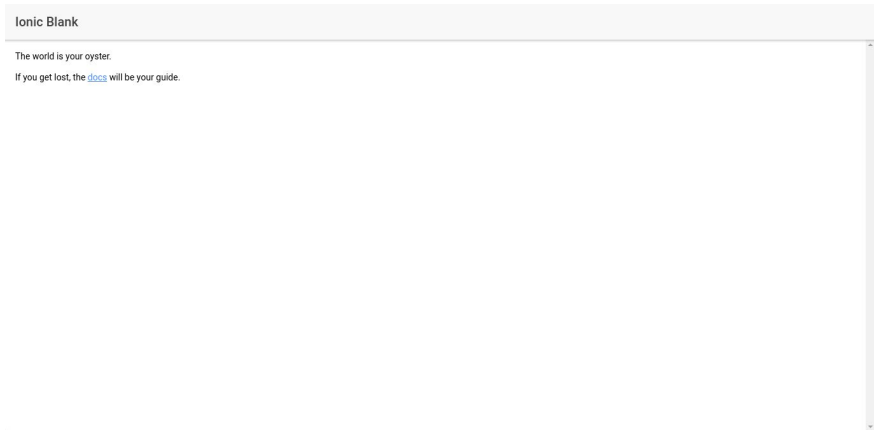
1. **Crear el proyecto**
   a. Creación del proyecto
      > ionic start ionic-example blank



   b. Ingresar al directorio del proyecto
      > cd ionic-example

   c. Ejecutar por primer vez el proyecto

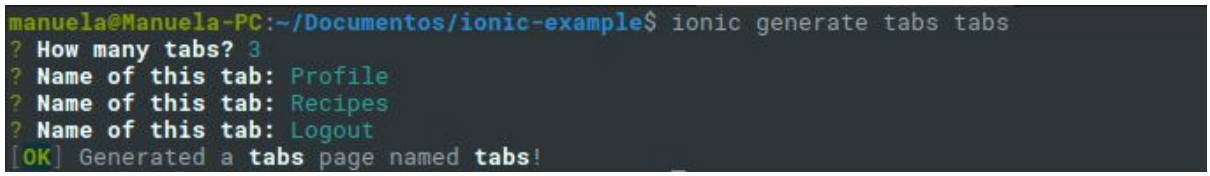
> ionic serve



2. Crear vista principal

a. Crear tabs

> ionic generate tabs



b. Importar la página “TabsPage” en el módulo de la aplicación, agregandola a “declarations” y “entryComponents”

app.module.ts

```
import { IonicApp, IonicErrorHandler, IonicModule } from 'ionic-angular';
import { SplashScreen } from '@ionic-native/splash-screen';
import { StatusBar } from '@ionic-native/status-bar';

import { MyApp } from './app.component';
import { HomePage } from '../pages/home/home';
import { TabsPage } from '../pages/tabs/tabs';

@NgModule({
  declarations: [
    MyApp,
    HomePage,
    TabsPage
  ],
  imports: [
    BrowserModule,
    IonicModule.forRoot(MyApp)
  ],
  bootstrap: [IonicApp],
  entryComponents: [
    MyApp,
    HomePage,
    TabsPage
  ],
  providers: [
```

c. Cambiar el rootPage por “TabsPage” en el archivo “src/app/app.component.ts”

app.component.ts

```
import { Component } from '@angular/core';
import { Platform } from 'ionic-angular';
import { StatusBar } from '@ionic-native/status-bar';
import { SplashScreen } from '@ionic-native/splash-screen';
import { TabsPage } from '../pages/tabs/tabs';
@Component({
  templateUrl: 'app.html'
})
export class MyApp {
  rootPage:any = TabsPage;

  constructor(platform: Platform, statusBar: StatusBar, splashScreen: SplashScreen) {
    platform.ready().then(() => {
      // Okay, so the platform is ready and our plugins are available.
      // Here you can do any higher level native things you might need.
      statusBar.styleDefault();
      splashScreen.hide();
    });
  }
}
```

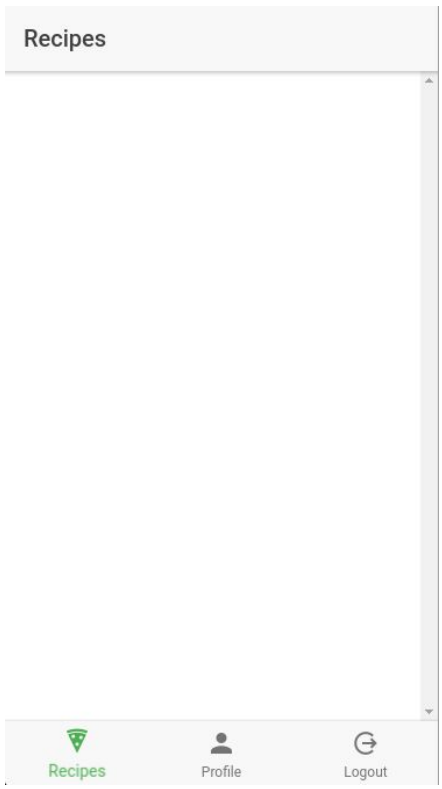
d. Personalizar los iconos de las tabs [Documentación sobre ionicons]

```
tabs.html
1  <ion-tabs>
2      <ion-tab [root]="profileRoot" tabTitle="Profile" tabIcon="person"></ion-tab>
3      <ion-tab [root]="recipesRoot" tabTitle="Recipes" tabIcon="pizza"></ion-tab>
4      <ion-tab [root]="logoutRoot" tabTitle="Logout" tabIcon="log-out"></ion-tab>
5  </ion-tabs>
```

e. Personalizar colores de la aplicación en el archivo "src/theme/variables.scss"

```
variables.scss
25
26  $colors: (
27    primary:    #4caf50,
28    secondary:  #ff9800,
29    danger:     #ff5722,
30    light:      #f4f4f4,
31    dark:       #222
32  );
```

Aplicación actual:



3. Creación de provider para la página "Recipes"

a. Generar Provider para obtener las recetas
> ionic generate provider recipes

```
manuela@Manuela-PC:~/Documentos/ionic-example$ ionic generate provider recipes
[OK] Generated a provider named recipes!
```

b. Crear la clase “recipe” en el provider y un arreglo de las mismas

recipes.ts

```ts
import { Injectable } from '@angular/core';
import { Http } from '@angular/http';
import 'rxjs/add/operator/map';

export class Recipe {
  constructor(
    public id?: number,
    public name?: string,
    public ingredients?: string,
    public instructions?: string
  ) { }
}

const RECIPES: Recipe[] = [];

@Injectable()
export class RecipesProvider {

  constructor(public http: Http) {
    console.log('Hello RecipesProvider Provider');
  }

}
```

c. Métodos para agregar, listar y eliminar recetas.

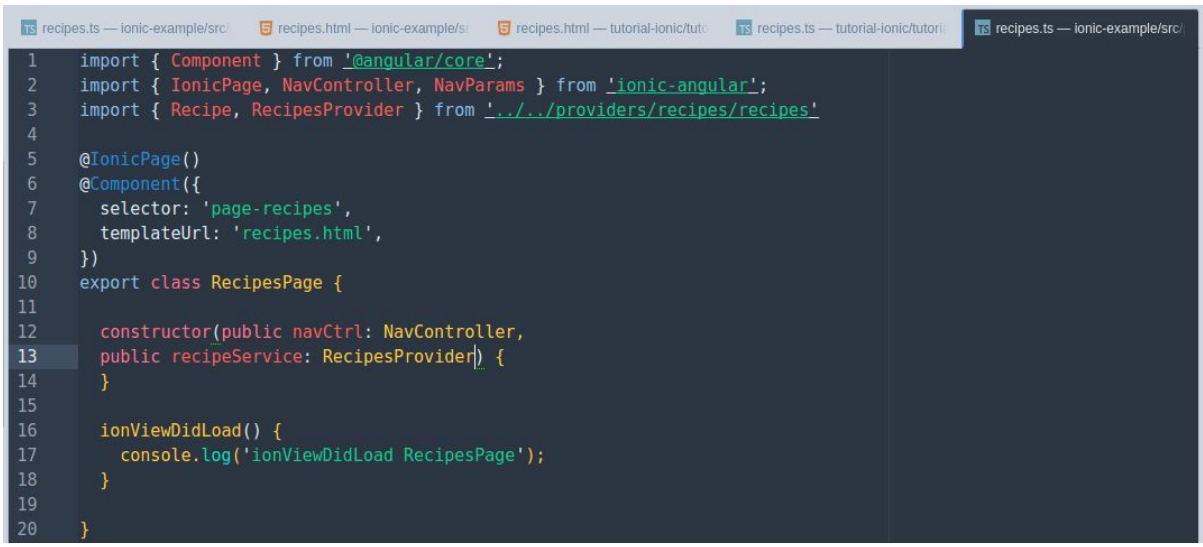
recipes.ts

```ts
addRecipe(recipe): void {
  let recipeId = RECIPES.reduce((max, recipe) => Math.max(recipe.id, max), -1) + 1;
  recipe.id = recipeId;
  RECIPES.push(recipe);
}

getRecipes(): Promise<Recipe[]> {
  return new Promise<Recipe[]>(resolve => {
    setTimeout(() => {
      resolve(RECIPES);
    }, FETCH_LATENCY);
  })
}

getRecipe(id: number | string): Promise<Recipe> {
  return new Promise<Recipe>(resolve => {
    setTimeout(() => {
      resolve(RECIPES.filter(recipe => recipe.id === +id)[0]);
    }, FETCH_LATENCY);
  })
}

removeRecipe(id: number | string): void {
  let recipeIdx = +id - 1;
  RECIPES.splice(recipeIdx, 1);
}
```

Documentación:

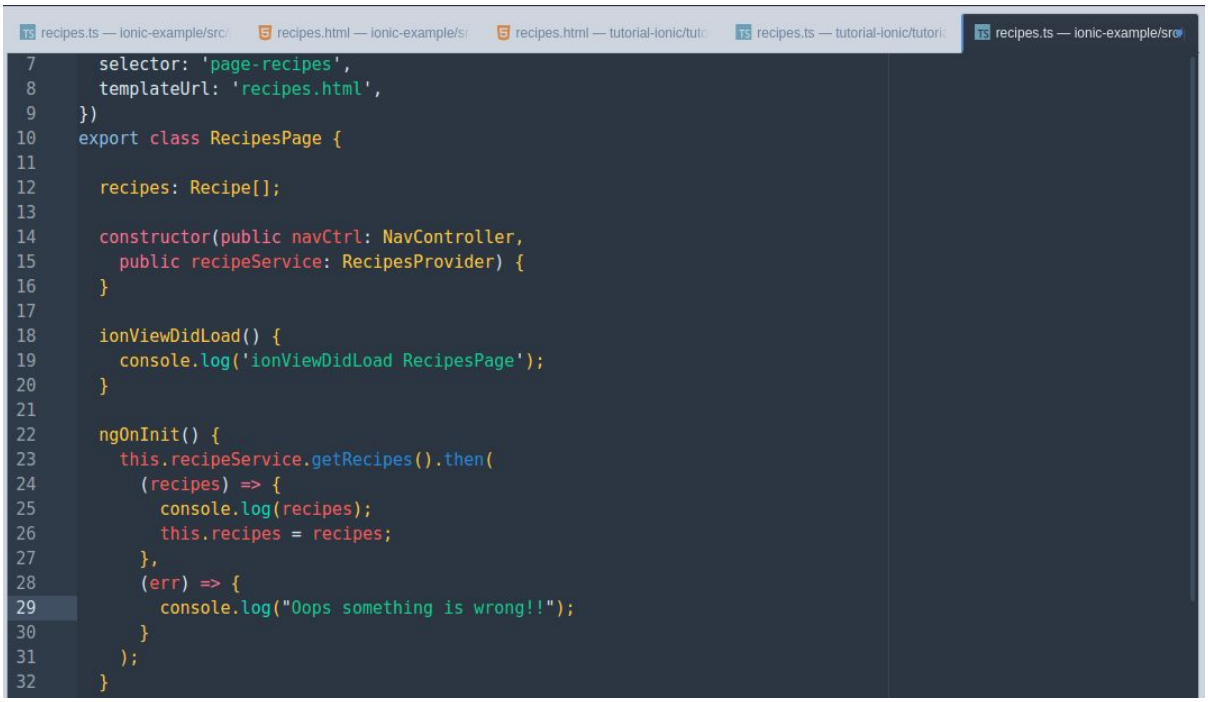
- Método Reduce()
- Promesas en Javascript
- Método Splice()

d. Importar servicio de recetas en el controlador de la página "recipes"



```
import { Component } from '@angular/core';
import { IonicPage, NavController, NavParams } from 'ionic-angular';
import { Recipe, RecipesProvider } from '../../providers/recipes/recipes'

@IonicPage()
@Component({
  selector: 'page-recipes',
  templateUrl: 'recipes.html',
})
export class RecipesPage {

  constructor(public navCtrl: NavController,
  public recipeService: RecipesProvider) {
  }

  ionViewDidLoad() {
    console.log('ionViewDidLoad RecipesPage');
  }

}
```

e. Cargar la lista de recetas desde el provider en el método "ngOnInit()" y guardarlas en un arreglo.



```
  selector: 'page-recipes',
  templateUrl: 'recipes.html',
})
export class RecipesPage {

  recipes: Recipe[];

  constructor(public navCtrl: NavController,
    public recipeService: RecipesProvider) {
  }

  ionViewDidLoad() {
    console.log('ionViewDidLoad RecipesPage');
  }

  ngOnInit() {
    this.recipeService.getRecipes().then(
      (recipes) => {
        console.log(recipes);
        this.recipes = recipes;
      },
      (err) => {
        console.log("Oops something is wrong!!");
      }
    );
  }
```

Documentación:

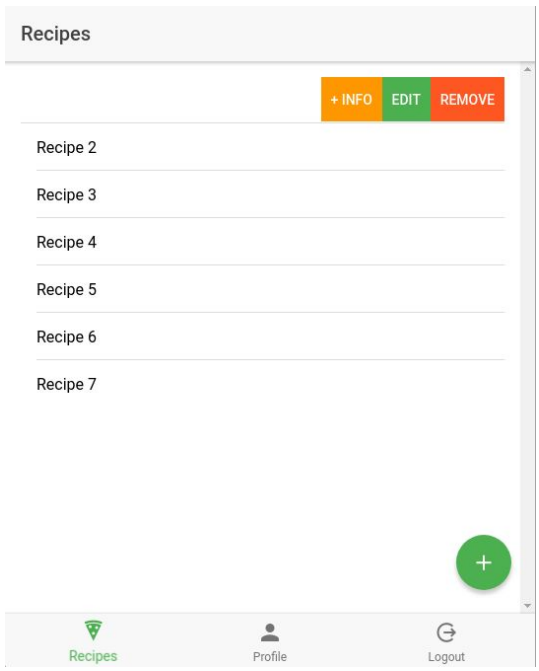
- Ciclo de vida de una vista

4. Creación de vista “Recipes”

a. Se crea la lista de elementos que va a mostrar las recetas.

recipes.html

```
</ion-header>

<ion-content padding>
  <ion-list *ngIf="recipes && recipes.length > 0">
    <ion-item-sliding *ngFor="let recipe of recipes">
      <ion-item>
        {{recipe.name}}
      </ion-item>

      <ion-item-options side="right">
        <button ion-button color="secondary" (click)="presentRecipeModal(recipe)">+ Info</button>
        <button ion-button (click)="edit(recipe)">Edit</button>
        <button ion-button color="danger" (click)="remove(recipe)">Remove</button>
      </ion-item-options>
    </ion-item-sliding>
  </ion-list>

  <ion-fab right bottom>
    <button ion-fab (click)="add()">
     <ion-icon name="add"></ion-icon>
   </button>
  </ion-fab>

  <p *ngIf="recipes && recipes.length == 0">
    No recipes available
  </p>

</ion-content>
```



5. Crear formulario para agregar y editar recetas.

Ya que el formulario para agregar y editar recetas es similar, este se puede crear como componente para ser reutilizado posteriormente.

a. Crear componente “recipe-form”

```
manuela@Manuela-PC:~/Documentos/ionic-example$ ionic generate component recipe-form
[OK] Generated a component named recipe-form!
```

b. Editar vista del formulario

recipe-form.html

```
<form #recipeForm="ngForm" name="recipeForm" novalidate>
  <ion-item>
    <ion-label floating color="primary">Name</ion-label>
    <ion-input [(ngModel)]="recipe.name" name="name" type="text" #name="ngModel" required></ion-input>
  </ion-item>

  <p [hidden]="name.valid" color="danger" padding-left>
    The field is required
  </p>

  <ion-item>
    <ion-label floating color="primary">Ingredients</ion-label>
    <ion-textarea [(ngModel)]="recipe.ingredients" name="ingredients" type="text" #ingredients="ngModel" requ
  </ion-item>

  <p [hidden]="ingredients.valid" color="danger" padding-left>
    The field is required
  </p>

  <ion-item>
    <ion-label floating color="primary">Instructions</ion-label>
    <ion-textarea [(ngModel)]="recipe.instructions" name="instructions" type="text" #instructions="ngModel" r
  </ion-item>

  <p [hidden]="instructions.valid" color="danger" padding-left>
    The field is required
  </p>

  <ion-row responsive-sm>
    <ion-col>
      <button ion-button [disabled]="!recipeForm.valid" (click)="processRecipe()" type="submit" color="primar
          {{submitBtn}}
      </button>
    </ion-col>
  </ion-row>
</form>
```

Documentación:

- NgModel

c. Crear instrucciones para el texto del botón “submit”

recipe-form.html | recipe-form.ts

```ts
import { Component, Input, Output, EventEmitter } from '@angular/core';
import { Recipe, RecipesProvider } from '../../providers/recipes/recipes'

@Component({
  selector: 'recipe-form',
  templateUrl: 'recipe-form.html'
})
export class RecipeFormComponent {

  @Input() recipe: Recipe;
  @Input() isUpdate: boolean;
  @Output() fireAction: EventEmitter<Recipe> = new EventEmitter<Recipe>();
  submitBtn: string;

  constructor(private recipeService: RecipesProvider) {
    console.log('Hello RecipeFormComponent Component');
  }

  ngOnInit() {
    this.submitBtn = this.isUpdate ? "Update" : "Save";
  }

  processRecipe(): void {
    this.fireAction.emit(this.recipe);
  }

}
```

Documentación:

- @Input y @Output
- EventEmitter

6. **Agregar recetas!**
    a. Generar página para agregar receta
    > ionic generate page recipe-add

```
manuela@Manuela-PC:~/Documentos/ionic-example$ ionic generate page recipe-add
[OK] Generated a page named recipe-add!
```

    b. Importar página en el módulo de la aplicación

recipe-add.html | recipe-add.ts | recipes.ts | app.module.ts

```
import { MyApp } from './app.component';
import { HomePage } from '../pages/home/home';
import { TabsPage } from '../pages/tabs/tabs';
import { RecipeAddPage } from '../pages/recipe-add/recipe-add';
import { RecipesProvider } from '../providers/recipes/recipes';
import { RecipeFormComponent } from '../components/recipe-form/recipe-form';

@NgModule({
  declarations: [
    MyApp,
    HomePage,
    TabsPage,
    RecipeFormComponent,
    RecipeAddPage
  ],
  imports: [
    BrowserModule,
    IonicModule.forRoot(MyApp)
  ],
  bootstrap: [IonicApp],
  entryComponents: [
    MyApp,
    HomePage,
    TabsPage,
    RecipeAddPage
```

c. Agregar componente “recipe-form” a la vista “recipe-add”

recipe-add.html

```
<ion-header>

  <ion-navbar>
    <ion-title>Add recipe</ion-title>
  </ion-navbar>

</ion-header>


<ion-content padding>
  <recipe-form (fireAction)="save($event)" [isUpdate]="false" [recipe]="recipe"></recipe-form>
</ion-content>
```

d. Crear método para agregar receta a través del “RecipesProvider”

recipe-add.html | recipe-add.ts

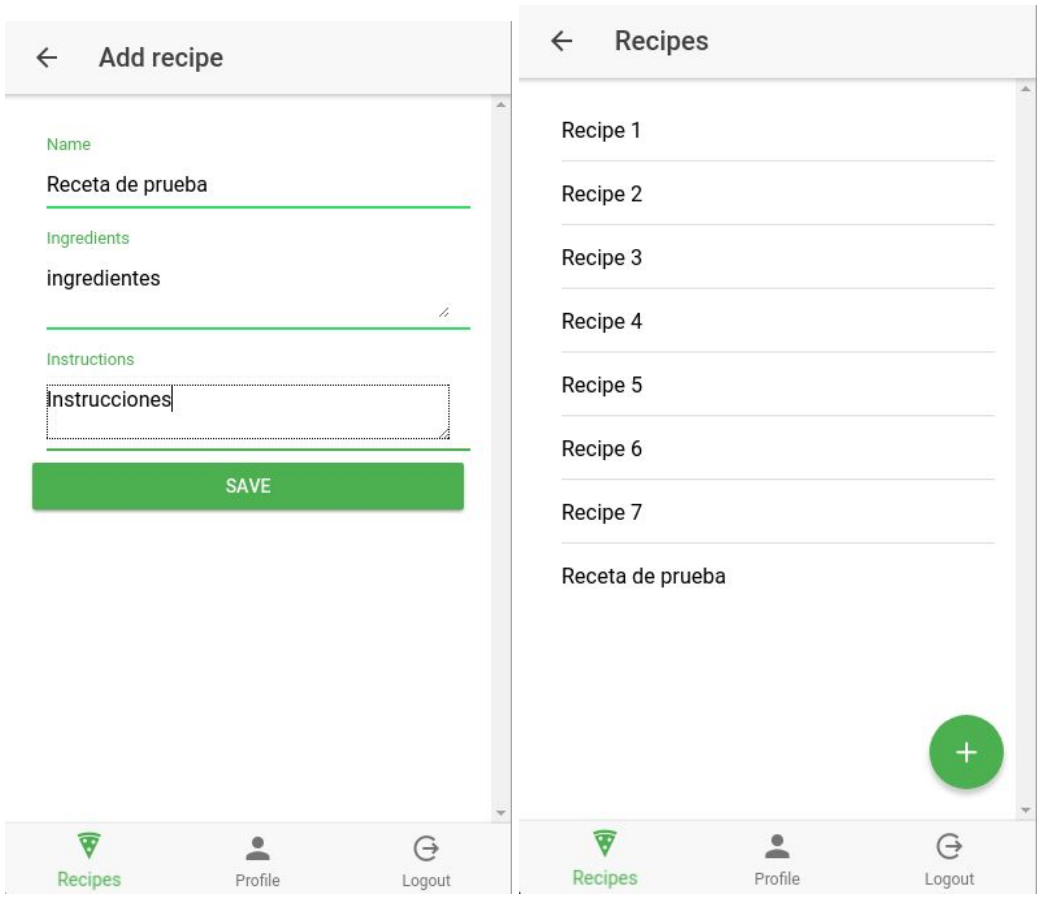
```ts
import { IonicPage, NavController } from 'ionic-angular';
import { Recipe, RecipesProvider } from '../../providers/recipes/recipes';
import { RecipesPage } from '../recipes/recipes';

@IonicPage()
@Component({
  selector: 'page-recipe-add',
  templateUrl: 'recipe-add.html',
})
export class RecipeAddPage {

  recipe: Recipe = new Recipe();
  constructor(
    public navCtrl: NavController,
    private recipeService: RecipesProvider
  ) { }

  save(recipe) {
    this.recipeService.addRecipe(recipe);
    this.navCtrl.push(RecipesPage);
  }

  ionViewDidLoad() {
    console.log('ionViewDidLoad RecipeAddPage');
  }

}
```

e. Implementar método “add()” en el controlador de la página “recipes”, que se ejecutará al pulsar el botón con el símbolo “+”

recipe-add.html | recipe-add.ts | recipes.ts

```ts
import { Component } from '@angular/core';
import { IonicPage, NavController } from 'ionic-angular';
import { Recipe, RecipesProvider } from '../../providers/recipes/recipes';
import { RecipeAddPage } from '../recipe-add/recipe-add';
```

```ts
  add() {
    this.navCtrl.push(RecipeAddPage);
  }
```



7. **Mostrar detalles de una receta**
    a. Crear página “recipe-info”
    > ionic generate page recipe-info

```
manuela@Manuela-PC:~/Documentos/ionic-example$ ionic generate page recipe-info
[OK] Generated a page named recipe-info!
```

    b. Importar página al módulo de la aplicación

app.module.ts

```
import { RecipeFormComponent } from '../components/recipe-form/recipe-form';
import { RecipeInfoPage } from '../pages/recipe-info/recipe-info';

@NgModule({
  declarations: [
    MyApp,
    HomePage,
    TabsPage,
    RecipeFormComponent,
    RecipeAddPage,
    RecipeInfoPage
  ],
  imports: [
    BrowserModule,
    IonicModule.forRoot(MyApp)
  ],
  bootstrap: [IonicApp],
  entryComponents: [
    MyApp,
    HomePage,
    TabsPage,
    RecipeAddPage,
    RecipeInfoPage
  ],
```

c. Importar “ModalController” en el controlador de la página “recipes”

app.module.ts | recipe-info.html | recipe-info.ts | recipes.ts | recipes.html

```
import { Component } from '@angular/core';
import { ModalController, IonicPage, NavController } from 'ionic-angular';
import { Recipe, RecipesProvider } from '../../providers/recipes/recipes';
import { RecipeAddPage } from '../recipe-add/recipe-add';
import { RecipeInfoPage } from '../recipe-info/recipe-info';

@IonicPage()
@Component({
  selector: 'page-recipes',
  templateUrl: 'recipes.html',
})
export class RecipesPage {

  recipes: Recipe[];

  constructor(public navCtrl: NavController,
    public modalCtrl: ModalController,
    public recipeService: RecipesProvider) {
  }
```

d. Implementar método “PresentRecipeModal()” en el controlador de la página “recipes” el cual se ejecuta al seleccionar la opción “+ info” en cada una de las recetas.

```
presentRecipeModal(recipe) {
  let recipeModal = this.modalCtrl.create(RecipeInfoPage, {
    recipe: recipe
  });
  recipeModal.present();
}
```

e. Importar “ViewController” en el constructor del controlador de la página “recipe-info”

app.module.ts | recipes.ts | recipe-info.ts | recipe-info.html

```ts
import { Component } from '@angular/core';
import { IonicPage, NavController, ViewController, NavParams } from 'ionic-angular';

@IonicPage()
@Component({
  selector: 'page-recipe-info',
  templateUrl: 'recipe-info.html',
})
export class RecipeInfoPage {

  constructor(public navCtrl: NavController,
    public navParams: NavParams,
    public viewCtrl: ViewController) { }
```

f. En el método “ngOnInit” recuperar la receta enviada como parámetro y asignarla a una variable tipo “Recipe” en el controlador asegurándonos de importar la clase “Recipe”.

```ts
  ngOnInit() {
    this.recipe = this.navParams.get("recipe");
  }
```

g. Crear vista HTML con una card en la cual se va a mostrar la información detallada de la receta.

app.module.ts | recipes.ts | recipe-info.ts | recipe-info.html

```html
<ion-header>

  <ion-toolbar>
    <ion-title>Recipe detail</ion-title>
    <ion-buttons start>
      <button ion-button (click)="closeModal()">
    <ion-icon name="close"></ion-icon>
  </button>
    </ion-buttons>
  </ion-toolbar>

</ion-header>

<ion-content padding>
  <ion-card>
    <img [src]="'assets/img/recipe.jpg'" />
    <ion-card-content>
      <ion-card-title color="secondary">{{recipe.name}}</ion-card-title>
      <h2>Ingredients:</h2>
      <p> {{recipe.ingredients}} </p>
      <h2>Instructions:</h2>
      <p> {{recipe.instructions}} </p>
    </ion-card-content>
  </ion-card>
</ion-content>
```

h. Implementar método “closeModal()” para cerrar el modal

```ts
  closeModal(){
    this.viewCtrl.dismiss();
  }
```

Generar APK:

> cordova platform add android

> cordova build --release android

Ejercicio propuesto:

- Implementar la funcionalidad para editar y eliminar recetas.