INSTALATION LOCAL ENV

----------------------

node -v

npm install ionic cordova -g

ionic -v

cordova -v

START

----------------------

ionic start nameApp blank / tabs / sidemenu

ionic lab

ionic serve

ionic cordova run android

ionic cordova run android --prod –release

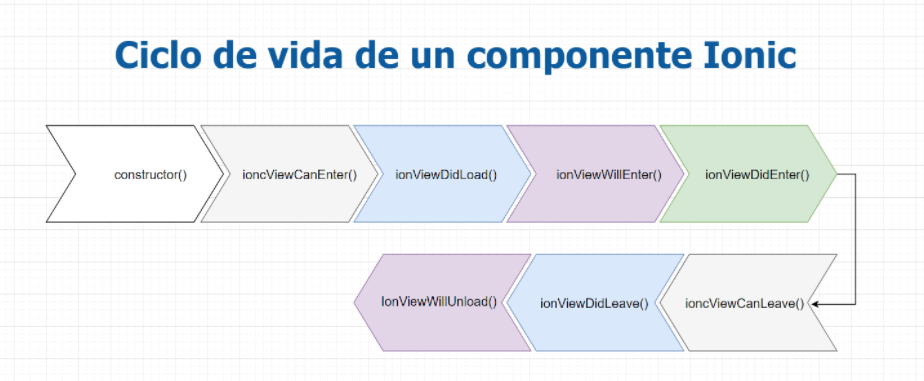
ionic build

ionic cordova prepare android

generar splash / icon

ionic cordova resources

"<preference name="AndroidPersistentFileLocation" value="Compatibility" />"



#### constructor

Debe de mantenerse lo más simple posible y solo inicializar las variables de su clase. Este evento se dispara cuando se está creando la página y solo lo hace una vez.

#### ionViewCanEnter

Este sería nuestro guardia de navegación y es muy útil si queremos chequear algo antes de entrar a la página. Es el evento ideal para comprobar si cuentas con los permisos para poder ver el contenido de la página. Debes de asegurarte que la lógica que utilices acá devuelva un booleano.

#### ionViewDidLoad

Se ejecuta cuando la página se ha cargado. Este evento sólo se activa una vez y sólo una vez como la función constructor(), pero la diferencia es que en este punto, puede estar seguro de que todas sus variables y dependencias inyectadas están disponibles para su uso. Esta es la función donde usted puede hacer todas sus llamadas HTTP iniciales para obtener sus datos y hacer el levantamiento pesado principal de su aplicación. El código que se ejecuta aquí solo se llama una vez, no por tanto una vez es cacheada la página.

#### ionViewWillEnter

Se ejecuta cuando la página está a punto de entrar y convertirse en la página activa. A pesar de que la página está completamente cargada desde el punto de vista del código, todavía hay mucha magia que se está realizando detrás de la escena para hacer que tu página sea totalmente activa y visible para el usuario. Ionic todavía tiene una cierta lógica interna para manejar cosas como la animación de las transiciones etc. Este es un gran lugar para tareas que se deben de realizar siempre que entras en la vista, como activar listener de eventos, actualizar una tabla, ocultar y mostrar cosas en su página antes de que sea visible para el usuario etc.

#### ionViewDidEnter

Se ejecuta cuando la página ha entrado completamente y ahora es la página activa. Este evento se disparará, ya sea la primera carga o al cargar la página en caché. Es la función final que se activa al navegar en una página como parte del ciclo de vida de ella. Acá ya puedes poner toda la lógica que desees aunque se utiliza fundamentalmente para alguna característica que necesitas que el usuario vea muy pronto o la primera cosa que desees que el usuario interactúe cuando entre a la página.

#### ionViewCanLeave

Corre antes de que la vista pueda salir. Esto puede usarse como una especie de “guardia” en las vistas autenticadas donde necesita comprobar los permisos antes de que la vista pueda salir. En esencia es lo mismo a ionViewCanEnter() pero en este caso te deja salir de la página si la lógica que utilizaste dentro de ella retorna true.

#### ionViewWillLeave

Se ejecuta cuando la página está a punto de salir. En el punto en que se desencadena este evento, sigue siendo la página activa, pero se ha puesto en cola para que se elimine y ya no se puede evitar salir de la página. Acá puede ser un buen lugar para preparar los datos que podrías utilizar en la siguiente página o realizar una llamada asíncrona alguna API que se necesitará en la próxima vista.

#### ionViewDidLeave

Se ejecuta cuando la página ha terminado de salir y ya no es la página activa. Esta función es un gran lugar para guardar datos o estados de la página que está dejando, así como activar algunas operaciones de fondo que no requieren que la vista sea visible

#### ionViewWillUnload

Es el último evento que se dispara en el ciclo de vida de una página típica de Ionic. Este es el lugar donde se liberan los recursos que ya no son necesarios y todo tipo de limpieza para evitar posibles pérdidas de memoria. Aquí haces el “unsubscribe” a los eventos y observables que te hayas suscrito. Esta función sólo se ejecuta una vez y es la última parada antes de que la página se destruya y se desvincula de la vista con todos sus elementos eliminados.

# PAGES & NAVIGATION

ionic generate page namePage

Para la navegación necesitamos importar NavController.

## Ir a una página pasándole parámetros

this.navCtrl.push(RecipePage, {idRecipe: id});

Ir a la página RecipePage, y le pasamos un objeto con key idRecipe = al valor de la variable id.

En la página RecipePage necesitamos importar NavParams:

import { NavController, NavParams } from 'ionic-angular';

para poder recoger los parámetros pasados con el push.

ngOnInit(){

this.idRecipe = this.navParams.get('idRecipe');

}

## Back buttons

this.nav.setRoot(HomePage);

this.nav.push(page.component);

this.navCtrl.pop(), // removing a view -> as back button

this.navCtrl.popToRoot(); // go to root page

# Alerts

onAlertDelete(idRecipe) {

const alert = this.alertCtrl.create({

title: 'Borrar receta',

message: '¿Estás seguro de que quieres borrar esta receta?',

buttons: [

{

text: 'Ok',

handler: () => {

this.deleteRecipe(idRecipe);

}

},

{

text: 'Cancelar',

role: 'cancel'

}

]

});

alert.present();

}

# Modals

modalRecipe(recipe) {

const modal = this.modalCtrl.create(RecipePage, recipe);

modal.present();

}

# Gestures

* (click)
* (tap)
* (press)