

MANUAL TÉCNICO PARA APLICACIÓN DE GESTION DE INVENTARIO (ANDROID)

Juan Francisco Galván De Santiago
CA AUTOMOTIVE DURANGO

APÉNDICE A

A.1. INTRODUCCIÓN

En este capítulo se repasará la estructura general del programa, para lograr que el programador se familiarice con ella, y que sea capaz de realizar modificaciones y ampliaciones a la herramienta.

A.2. ESTRUCTURA GENERAL DE LA APLICACIÓN

La estructura general de la aplicación consta de dos partes, la primera desarrollada en *Ionic* con ayuda de *AngularJS* que se encuentra en el dispositivo Android, y la segunda, que es un conjunto de distintos archivos escritos en *PHP* que permiten que la aplicación se comuniquen a la misma base de datos que hace uso la aplicación de escritorio.

A partir de ahora las dos partes de la aplicación serán llamadas “*Principal*” para la aplicación Android y “*Secundaria*” para los archivos escritos en *PHP*.

A.3. ESTRUCTURA DE LA HERRAMIENTA PRINCIPAL

La estructura de la aplicación en *Ionic* es la estándar para cualquier aplicación desarrollada usando *Ionic 3.9.2* por lo que podemos encontrar una estructura como la siguiente:

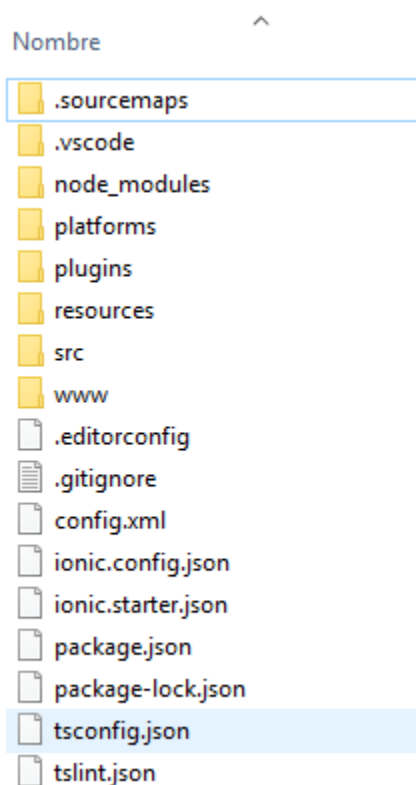


Ilustración 1. Estructura estándar de IONIC 3.9.2

Las vistas de la app, las podemos encontrar en la ruta: `/NombreApp/src/pages` estas vistas se componen de tres elementos principales, un archivo `.html` que contiene el maquetado de la vista, un archivo `.ts` que contiene todos los scripts de la vista en específico y un archivo `.scss` que puede contener los elementos visuales de la vista en cuestión.

La parte central de la aplicación se puede encontrar en la ruta: `/NombreApp/src/app` para esta aplicación en cuestión contiene un archivo de menú lateral llamado `app.html`, un archivo llamado `app.components.ts` que es el primer archivo `.ts` cargado durante la ejecución de la aplicación y en el podemos encontrar las funciones:

- `InitializeApp()`:
 - Se realizan acciones de manera nativa para Android, en este caso aquí se establece el color de la barra de estado y se oculta la pantalla splash, una vez que la app se ha inicializado.
- `openPage()`:
 - Se establece la pagina de inicio por defecto de la aplicación, en este caso es la pagina de *Login* para poder iniciar sesión.

Actualmente la aplicación cuenta con 22 vistas distintas las cuales son:

- `add-item`
 - Encargada de añadir productos nuevos a la base de datos
- `Addprovider`
 - Encargada de añadir un proveedor nuevo a la base de datos
- `Admin`
 - Contiene la vista de administrador
- `Dashboard`
 - Contiene la vista de operador
- `Deleteitem`
 - Contiene la vista para borrar ítems de la base de datos
- `Delete-provider`
 - Encargada de borrar un proveedor de la base de datos
- `Delete-user`
 - (Puedes cambiar el nombre si lo deseas) Contiene la vista para la gestión de usuarios
- `Delete-user-action`
 - Contiene una vista que contiene los datos de un usuario seleccionado para ser borrado
- `Departures`
 - Contiene la vista para la función de “salida individual”
- `Edit-provider`
 - Contiene la vista que obtiene datos de un proveedor en específico y permite cambiar el nombre
- `Edit-user-action`
 - Contiene una vista que obtiene los datos de un usuario en específico, y permite cambiar los valores de “nombre completo” “email” y “cargo” del usuario.

- Generateqr
 - Vista encargada de generar un Código QR de un ítem del stock en específico
- Guest-view
 - Contiene la vista de invitado
- Home
 - (Puedes cambiar el nombre si lo deseas) Contiene la vista de la función “Gestión Inventario” de la aplicación
- Login
 - Contiene la vista para el inicio de sesión, es la pagina de inicio de la aplicación
- Modify
 - (Puedes cambiar el nombre si lo deseas) Contiene la vista que permite obtener los datos de un elemento en específico del inventario para editarlos
- Multiple-departure
 - Contiene la vista para poder realizar una salida múltiple, que es la función mas “compleja” de la aplicación, envía los parámetros de “Codigo”, “nombre” y “cantidad” a una tabla llamada “Volatile” en la base de datos, para después ser recuperados en formato JSON y poder realizar el reporte
- Product-entry
 - Contiene la vista que permite ingresar una cantidad a un elemento en específico del inventario, además de cambiar su FIFO
- Providers
 - Vista que se encarga de desplegar todos los proveedores registrados en la base de datos, para después eliminar, agregar o editarlos.
- Register
 - Permite registrar un usuario nuevo en la base de datos
- Reports
 - Vista para generar un reporte general de stock en la aplicación
- Searchphysical
 - Vista de la función de Escáner de códigos QR de la aplicación.

A.4. ESTRUCTURA DE LA HERRAMIENTA SECUNDARIA

La estructura de la herramienta secundaria consta de una serie de archivos escritos en *PHP*, que se encargan de realizar peticiones a la base de datos *MySQL*.

Los archivos que lo conforman son 24 archivos, que realizan funciones distintas, como borrar elementos, añadir elementos nuevos, comprobar el inicio de sesión o simplemente obtener campos de la base de datos.

23 archivos se encuentran del lado del servidor de la app en la ruta: `/www/IonicApp/`,

Mientras que uno se encuentra en la ruta: `/www/` debido a que contiene los datos de conexión a la base de datos.

Los archivos principales son los siguientes:

- dbConnect.php
 - Contiene la conexión a la base de datos de la aplicación
- Delete_provider.php
 - Se encarga de borrar un proveedor de la base de datos
- Delete_stock.php
 - Se encarga de eliminar un ítem de la base de datos
- Delete_user.php
 - Se encarga de eliminar el registro de un usuario de la base de datos
- Departure_stock.php
 - Se encarga de realizar una resta de una cantidad definida por el usuario de un elemento de la base de datos
- Edit_provider.php
 - Se encarga de editar un proveedor en la base de datos
- Edit_user.php
 - Se encarga de editar un registro de usuario de la base de dato
- Json_departure_reciever.php
 - Se encarga de recibir los parámetros de la Función “Enviar a Salida múltiple” de la aplicación y enviarlos a la tabla “volatile” de la base de datos
- Json_departure_checker.php
 - Se encarga de recibir los elementos de la tabla “volatile” de la base de datos
- Json_departure_decrypter.php
 - Se encarga de realizar la resta de las cantidades de los elementos que se envían a la vista de “Salida Múltiple” de la ampliación
- Json_entry_stock.php
 - Se encarga de realizar la adición a una cantidad de un producto dado en la base de datos, además de permitir cambiar el FIFO del mismo
- Json_fetch_charge.php
 - Se encarga traer los datos del cargo de usuario de la base de datos de la aplicación
- Json_fetch_name.php
 - Se encarga de traer los datos de nombre de usuario de la base de datos de la aplicación
- Json_fetch_providers.php
 - Se encarga de traer los datos de proveedores de la base de datos de la aplicación
- Json_fetch_user.php
 - Se encarga de traer los datos de usuarios de la base de datos de la aplicación
- Json_read.php
 - Se encarga de traer los datos de todos los elementos que estén en el stock
- Json_table_clearer.php
 - Se encarga de limpiar la tabla “volatile” de la base de datos para poder usarla después.

- Login.php
 - Se encarga de realizar la comprobación de la contraseña introducida por el usuario al momento de iniciar sesión con el Hash que se genera al momento de registrar al usuario.
- Post_one.php
 - Se encarga de dos cosas:
 - Insertar un nuevo ítem en la base de datos
 - Insertar un nuevo proveedor en la base de datos
- Qrcode_check.php
 - Se encarga de realizar una consulta con la cadena de texto obtenida después de escanear un código.
- Register.php
 - Se encarga de registrar los datos de un usuario nuevo, y de encriptar la contraseña que se elija al momento del registro.

APÉNDICE B

B.1. INTRODUCCIÓN

En este apéndice se explica como añadir una nueva característica a la aplicación.

B.2. AÑADIR UNA NUEVA VISTA A LA APLICACIÓN

Para añadir una nueva vista a la aplicación es necesario abrir una ventana de símbolo de sistema en el directorio raíz de la aplicación e introducir el comando:

```
ionic generate page nombrePagina
```

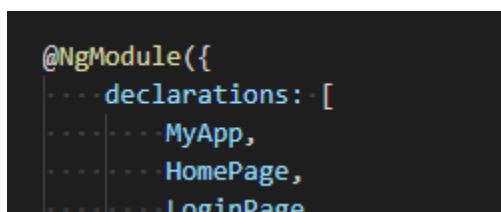
Y se creará una nueva carpeta con el nombre previamente introducido, con 4 archivos, un *.html*, un *.ts*, un *.scss* y un archivo *.module.ts*.

Sin embargo *ionic* no reconocerá la nueva pagina de manera inmediata, para esto, hay que agregarla al archivo *app.modules.ts* realizando de la siguiente manera:

Se hace la importación de la página en la parte superior del documento:

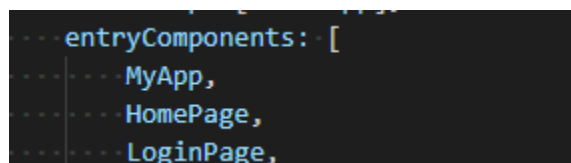
```
import { PaginaNombre } from '../pages/pagina-nombre/pagina-nombre';
```

Después dentro de ese mismo archivo es necesario añadir el nombre que se importa de la página al método “@NgModule({})” dentro de las secciones “declarations:” y “entrycomponents:”



```
@NgModule({  
  declarations: [  
    MyApp,  
    HomePage,  
    LoginPage,  
  ],  
})
```

Ilustración 2 Método @NgModule



```
entryComponents: [  
  MyApp,  
  HomePage,  
  LoginPage,  
]
```

Ilustración 3 Método "entryComponents"

