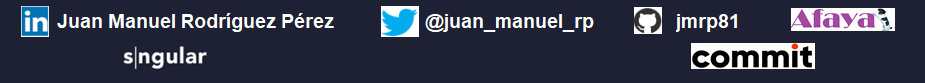
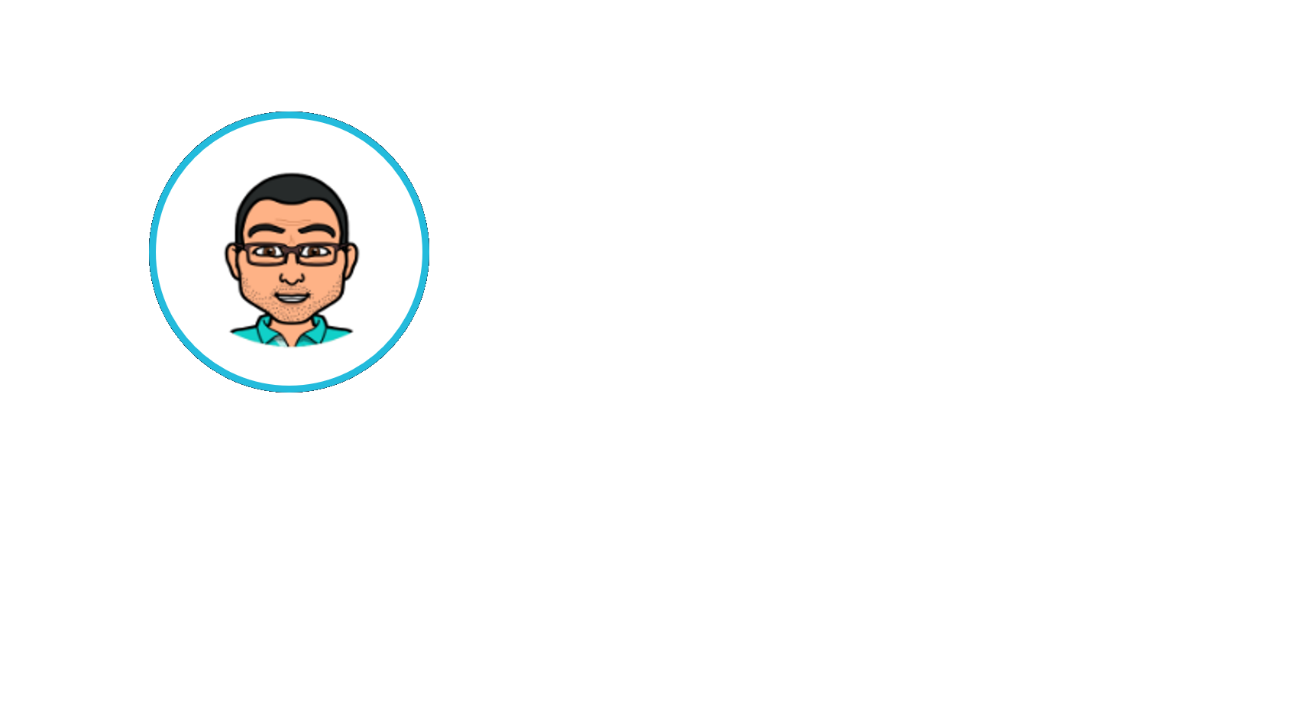
Introducción al Front mediante "Holas Mundos"



Contenidos

[1. Introducción 4](#_Toc529810103)

[2. Descripción Hardware & Software 5](#_Toc529810104)

[3. Pequeña Descripción de los Frameworks / Librerías 6](#_Toc529810105)

[3.1 Angular 6](#_Toc529810106)

[3.2 Aurelia 6](#_Toc529810107)

[3.3 React 6](#_Toc529810108)

[3.4 Vue 6](#_Toc529810109)

[3.5 Ionic 6](#_Toc529810110)

[3.6 NativeScript 7](#_Toc529810111)

[3.7 ReactNative 7](#_Toc529810112)

[4. Datos Comparativos 8](#_Toc529810113)

[4.1 Encuesta Twitter 8](#_Toc529810114)

[4.2 StackOverflow Etiquetas 8](#_Toc529810115)

[4.3 Cursos Udemy 9](#_Toc529810116)

[4.4 Github REPOSITORIOS 9](#_Toc529810117)

[4.5 Github Commits 10](#_Toc529810118)

[4.6 Github Issues 10](#_Toc529810119)

[5. Recursos 11](#_Toc529810120)

[5.1 Angular 11](#_Toc529810121)

[5.2 Aurelia 11](#_Toc529810122)

[5.3 React 11](#_Toc529810123)

[5.4 Vue 11](#_Toc529810124)

[5.5 Ionic 11](#_Toc529810125)

[5.6 NativeScript 11](#_Toc529810126)

[5.7 ReactNative 11](#_Toc529810127)

[6. Instalación 12](#_Toc529810128)

[6.1 General 12](#_Toc529810129)

[6.1.1 Instalación de Visual Studio Code 12](#_Toc529810130)

[6.1.2 Instalación de Node 12](#_Toc529810131)

[6.1.3 Uso de NPM 12](#_Toc529810132)

[6.1.4 Instalación de TypeScript 12](#_Toc529810133)

[6.2 Angular (Material) 13](#_Toc529810134)

[6.2.1 Elementos que instalar 13](#_Toc529810135)

[6.2.2 Creando nuestra aplicación 13](#_Toc529810136)

[6.2.3 Alguna configuración necesaria 14](#_Toc529810137)

[6.2.4 Ejemplo de las principales clases. 15](#_Toc529810138)

[6.3 Aurelia 19](#_Toc529810139)

[6.3.1 Instalación de Aurelia-Cli 19](#_Toc529810140)

[6.3.2 Creación del Proyecto 19](#_Toc529810141)

[6.4 React 20](#_Toc529810142)

[6.4.1 Crear proyecto 20](#_Toc529810143)

[6.4.2 Estructura del proyecto 20](#_Toc529810144)

[6.4.3 Ejecución de la aplicación 21](#_Toc529810145)

[6.5 Vue 22](#_Toc529810146)

[6.5.1 Instalación de Vue 22](#_Toc529810147)

[6.5.2 Instalación de vue-cli 22](#_Toc529810148)

[6.5.3 Herramientas para el navegador 22](#_Toc529810149)

[6.5.4 Creando nuestro proyecto. 22](#_Toc529810150)

[6.5.5 Ya creado el proyecto 25](#_Toc529810151)

[6.5.6 Ejemplo de archivos 26](#_Toc529810152)

[6.5.7 Ejecutando la app 30](#_Toc529810153)

[6.6 Ionic 31](#_Toc529810154)

[6.6.1 Preparación Entornos desarrollo Android e IOS 31](#_Toc529810155)

[6.6.2 Instalación de Ionic 31](#_Toc529810156)

[6.6.3 Creando nuestra aplicación 31](#_Toc529810157)

[6.6.4 Ejecutar la aplicación en navegador: 32](#_Toc529810158)

[6.6.5 Añadir platform Android al proyecto: 32](#_Toc529810159)

[6.6.6 Añadir platform iOS al proyecto: 32](#_Toc529810160)

[6.6.7 Ejecutar la aplicación en emulador: 32](#_Toc529810161)

[6.6.8 Ejecutar o generar la aplicación para producción: 32](#_Toc529810162)

[6.7 NativeScript 33](#_Toc529810163)

[6.7.1 Preparación Entornos desarrollo Android e IOS 33](#_Toc529810164)

[6.7.2 Instalación de NativeScript 33](#_Toc529810165)

[6.7.3 Creando nuestra Aplicación 33](#_Toc529810166)

[6.7.4 Preparar las platform: 34](#_Toc529810167)

[6.7.5 Ejecución de la aplicación: 34](#_Toc529810168)

[6.7.6 Dispositivos móviles disponibles: 34](#_Toc529810169)

[6.7.7 Algunos errores al ejecutar en emulador: 34](#_Toc529810170)

[6.8 ReactNative 35](#_Toc529810171)

[6.8.1 Preparación Entornos desarrollo Android e Ios 35](#_Toc529810172)

[6.8.2 Instalación de ReactNative 35](#_Toc529810173)

[6.8.3 Creación de un proyecto 35](#_Toc529810174)

[6.8.4 Ejecución del proyecto 35](#_Toc529810175)

[6.8.5 Ejecución del proyecto en simulador o dispositivo virtual 35](#_Toc529810176)

[7. Preparación Entorno de Desarrollo con Android 36](#_Toc529810177)

[7.1.1 Introducción 36](#_Toc529810178)

[7.1.2 Preparación de Android con Chocolatey 36](#_Toc529810179)

[7.1.3 Preparación de Android con Android Studio 38](#_Toc529810180)

[7.1.4 Emulador de Windows 38](#_Toc529810181)

[8. Bibliografía 39](#_Toc529810182)

[9. Control de versiones 40](#_Toc529810183)

1. Introducción

Este documento es la versión extendida de la presentación usada para la charla en la CommitConf 2018 en Madrid:

“Introducción al Front mediante ‘Holas Mundos’”

Lo que se pretende con este documento como con la presentación en la charla, **NO** eshacer una comparativa sobre los diferentes frameworks y/o librerías existentes para saber cuál es mejor o cual usar.

La intención es acercar la situación actual del ecosistema “Front End” como de las diferentes herramientas existentes para afrontar las diferentes necesidades en el desarrollo o a la hora de afrontar nuevos proyectos.

La temática surge a raíz de tener un perfil Full Stack e intentar orientarme más hacía el lado Front End y de que [Azahara Fernández Guizán](https://www.linkedin.com/in/azahara-fernandez-guizan-41713033/) estuviera haciendo pequeños proyectos con el Framework ‘Aurelia’.

Al investigar sobre los diferentes frameworks e intentar trabajar por primera vez con ellos a parte de la curva de aprendizaje de cada uno y hablando con gente que empieza en el sector me di cuenta de que a veces lo que más cuesta es hacer el primero y conseguir ejecutarlo.

Por ello es simplemente la realización de proyectos “Holas Mundos”, por lo que queda claro que tampoco se profundiza lo suficiente en cada framework y/o librería a la hora del desarrollo.

1. Descripción Hardware & Software

|  |  |
| --- | --- |
| **Hardware** | **Software** |
| * Marca y Modelo:   MSI – GE72 2QL(Apache)-252XES   * Procesador:   Intel Core i7-5700HQ 2.70 Ghz   * RAM:   8 GB DDR3L SODIMM   * Controlador Gráfico:   nVidia Geforce GTX950M, 2GB GDDR5   * Pantalla:   17.3" LED Full HD (1920x1080) 16:9 Anti-Glare   * Disco Duro:   Samsung 840 Evo 250 GB | * Windows 10 Pro * Visual Studio Code versión 1.28 * Node 8.12.0 LTS |

1. Pequeña Descripción de los Frameworks / Librerías
   1. Angular

Angular es un Framework multiplataforma desarrollado en TypeScript para usar el patrón MVC o MVVM en aplicaciones web basadas en SPA (Single Page Application).

Tiene licencia MIT y es mantenido por Google su última versión estable es la 7 y está fechada del 11 de octubre de 2018.

* 1. Aurelia

Aurelia es un Framework escrito con ecmaScript para usar en aplicaciones web, móvil o de escritorio usando los patrones MVC o MVVM.

Esta bajo licencia MIT y el equipo de desarrollo lo lidera Rob Eisenberg.

* 1. React

React es una librería multiplataforma desarrollado en JavaScript, uno de sus objetivos es ayudar a crear páginas web SPA manejando únicamente la Interfaz de usuario. Es decir, React sería la View en el modelo MVC o MVVM.

Se puede usar junto React-based usado para las partes no-UI.

Es mantenido por Facebook bajo licencia MIT y su última versión estable es la 16.4.2 fechada del 1 de agosto de 2018.

* 1. Vue

Vue o Vue.js es un framework multiplataforma desarrollado en JavaScript para la construcción de interfaces de usuario y páginas web basada en SPA.

Su autor original fue Evan You, tiene licencia MIT y su última versión es la 2.5.17 con fecha del 1 de agosto del 2018.

* 1. Ionic

Ionic es un framework para desarrollar aplicaciones hibridas para Android e IOS utilizando tecnología web como HTML, CSS, SASS, TypeScript; en las últimas versiones está construido sobre Angular.

Esta mantenido por la empresa Drift Co (creado por Max Lynch, Ben Sperry and Adam Bradley) bajo licencia MIT su última versión estable es la 3.9.2 con fecha 8 de noviembre de 2017.

* 1. NativeScript

NativeScript es un framework multiplataforma construido sobre JavaScript y TypeScript para construir aplicaciones para Android e IOS, soporta el uso de Angular y Vue.js.

Es mantenido por la empresa Telerik bajo una licencia Apache su última versión la 4.2.3 data del 28 de agosto de 2018.

* 1. ReactNative

ReactNative es un framework desarrollado a partir de React para desarrollar aplicaciones con arquitectura nativa de Android, IOS y UWP usando JavaScript y React, aunque te permite el uso de código nativo.

1. Datos Comparativos

Como se comenta en la introducción **NO** se pretende hacer una comparación para elegir uno u otro framework sino para hacernos una idea de sus usos por diferentes personas.

Los datos son obtenidos del mes de octubre de 2018.

* 1. **Encuesta Twitter**



* 1. **StackOverflow Etiquetas**
  2. **Cursos Udemy**
  3. **Github REPOSITORIOS**
  4. **Github Commits**
  5. **Github Issues**

1. Recursos
   1. Angular

* <https://angular.io/>
* https://github.com/angular/angular
* <https://material.angularjs.org/>
* A quien seguir en Twitter: **@carlosazaustre**
  1. Aurelia
* <https://aurelia.io>
* https://github.com/aurelia
* https://www.uno-de-piera.com/aurelia-un-gran-framework-javascript
* A quien seguir en Twitter: **@azahara\_fergui**
  1. React
* <https://reactjs.org>
* https://github.com/facebook/react
* https://www.udemy.com/aprendiendo-react
* A quien seguir en Twitter: **@midudev**
  1. Vue
* <https://github.com/jdonsan/estudiando-vue-js>
* https://carlosazaustre.es/que-es-lo-que-me-gusta-de-vue-js/
* https://vuejs.org/
* A quien seguir en Twitter: **@jdonsan**
  1. Ionic
* <https://ionicframework.com/>
* <https://github.com/ionic-team/ionic>
  1. NativeScript
* <https://docs.nativescript.org/tutorial/chapter-0>
* <https://www.progress.com/nativescript>
* https://nativescript-vue.org/en/docs/introduction/
  1. ReactNative
* <https://facebook.github.io/react-native/>
* <http://www.reactnative.com/>

1. Instalación

En este punto por cada framework/librería explicare los pasos para hacer la instalación del entorno y crear en cada uno de ellos un ‘Hola Mundo’.

* 1. General

En este apartado y sus subapartados describiré las instalaciones comunes a todos los frameworks/librerías.

* + 1. Instalación de Visual Studio Code

Para el desarrollo he usado el editor de texto Visual Studio Code de Microsoft, lo podéis descargar desde el enlace:

* <https://code.visualstudio.com/download>
  + 1. Instalación de Node

El primer paso de todo es instalar en nuestro equipo Node, para ello podéis acceder a su zona de descargas ( <https://nodejs.org/es/download/> ).

* + 1. Uso de NPM

Una vez instalado Node podemos buscar su consola de comandos para seguir instalando diferentes componentes usando NPM ( <https://www.npmjs.com/> ).

* + 1. Instalación de TypeScript

En todos los proyectos he usado el lenguaje TypeScript, que no deja de ser un superconjunto de JavaScript que básicamente añade tipado y objetos basados en clases. Su código será traducido a JavaScript a través de un compilador.

Para su instalación usando la línea de comandos de Node y NPM ejecutamos:

* npm install -g typescript
  1. Angular (Material)
     1. Elementos que instalar

Desde la consola de Node

1. Angular: **npm install -g angular**
2. Angular/cli: **npm install -g @angular/cli**
3. El uso de Bootstrap es opcional→ **npm install bootstrap --save**, en el documento style de nuestro proyecto ponemos:

**@import url('../node\_modules/bootstrap/dist/css/bootstrap.min.css');**

* + 1. Creando nuestra aplicación

Antes de empezar podemos ver más información sobre material en su web oficial

( <https://material.angular.io/> ) también tienen una guía de inicio

( <https://material.angular.io/guide/getting-started> )

1. Creamos proyecto de Angular desde la consola de comandos y situados en la carpeta raíz que queramos usar escribimos el comando:

**ng new HelloWorldMaterialAngular**

1. Instalamos Material en el proyecto anteriormente creado usando el siguiente comando en la terminal de comandos:

**npm install --save @angular/material @angular/cdk**

1. Además de lo anterior si quisiéramos tener animaciones y algunos componentes como (mat-slide-toggle, mat-slider, matTooltip) de HammerJS deberíamos instalarlos usando para animaciones el comando:

**npm install --save @angular/animations**

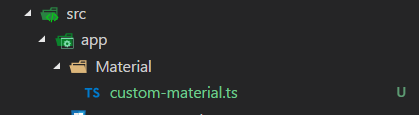
Y para HammerJS el comando:

**npm install -g --save hammerjs**

(Debemos tener en cuenta no confundir los comandos npm y ng, mientras uno es de angula (ng) el otro es de NPM por lo cual las ‘g’ tienen diferente significado. Mientras en el entorno NPM significa global, por lo que si la indicamos estaremos haciendo la instalación global para todos los proyectos que realicemos, en Angular es la abreviatura de generate.)

* + 1. Alguna configuración necesaria

Para el uso de Material recomiendo crear en nuestro proyecto un módulo que en mi caso llamo CustomMaterial donde tendremos todo el contenido a usar de material.



* Importamos dicho modulo en el root **app.module.ts:**

import { CustomMaterial } from './Material/custom-material';

import { BrowserAnimationsModule } from '@angular/platform-browser/animations';

* Ejemplo de la anterior clase custom citada para albergar material:

import { NgModule } from '@angular/core';

// imports material

import {

MatToolbarModule

} from '@angular/material';

@NgModule({

imports: [

MatToolbarModule

],

exports: [

MatToolbarModule

]

})

export class CustomMaterial {}

En ella iremos importando los módulos de Material a usar y añadiéndolos a import y export del módulo.

* Importación de un estilo predeterminado de Material, para ello usaremos la hoja general de estilos style.css del proyecto:

@import '~@angular/material/prebuilt-themes/purple-green.css';

* + 1. Ejemplo de las principales clases.

1. App.module.ts

import { BrowserModule } from '@angular/platform-browser';

import { NgModule } from '@angular/core';

import { HttpClientModule } from '@angular/common/http';

import { CustomMaterial } from './Material/custom-material';

import { BrowserAnimationsModule } from '@angular/platform-browser/animations';

import { AppComponent } from './app.component';

@NgModule({

declarations: [

AppComponent

],

imports: [

BrowserModule,

HttpClientModule,

CustomMaterial,

BrowserAnimationsModule

],

providers: [],

bootstrap: [AppComponent]

})

export class AppModule { }

1. App.component.ts

import { Component } from '@angular/core';

@Component({

selector: 'app-root',

templateUrl: './app.component.html',

styleUrls: ['./app.component.css']

})

export class AppComponent {

title = 'Angular Material';

}

1. App.component.html

<section>

<header>

<mat-toolbar color="primary">

<mat-toolbar-row>

<mat-icon svgIcon="helloIcon"></mat-icon>

<span>Welcome to {{ title }}!</span>

</mat-toolbar-row>

</mat-toolbar>

</header>

<article>

<mat-grid-list cols="2" color="primary">

<mat-grid-tile>

<mat-list role="list">

<mat-list-item role="listitem">

<img width="75" alt="Angular Logo" src="data:image/svg+xml;base64,PHN2ZyB4bWxucz0iaHR0cDovL3d3dy53My5vcmcvMjAwMC9zdmciIHZpZXdCb3g9IjAgMCAyNTAgMjUwIj4KICAgIDxwYXRoIGZpbGw9IiNERDAwMzEiIGQ9Ik0xMjUgMzBMMzEuOSA2My4ybDE0LjIgMTIzLjFMMTI1IDIzMGw3OC45LTQzLjcgMTQuMi0xMjMuMXoiIC8+CiAgICA8cGF0aCBmaWxsPSIjQzMwMDJGIiBkPSJNMTI1IDMwdjIyLjItLjFWMjMwbDc4LjktNDMuNyAxNC4yLTEyMy4xTDEyNSAzMHoiIC8+CiAgICA8cGF0aCAgZmlsbD0iI0ZGRkZGRiIgZD0iTTEyNSA1Mi4xTDY2LjggMTgyLjZoMjEuN2wxMS43LTI5LjJoNDkuNGwxMS43IDI5LjJIMTgzTDEyNSA1Mi4xem0xNyA4My4zaC0zNGwxNy00MC45IDE3IDQwLjl6IiAvPgogIDwvc3ZnPg=="> Some links to help you:

</mat-list-item>

<mat-divider></mat-divider>

<mat-list-item role="listitem">

<a class="urlToOtherWeb" target="\_blank" rel="noopener" href="https://angular.io/tutorial">

<button mat-raised-button>

<mat-icon svgIcon="earthIcon"></mat-icon>

Tour of Heroes

</button>

</a>

</mat-list-item>

<mat-list-item role="listitem">

<a class="urlToOtherWeb" target="\_blank" rel="noopener" href="https://github.com/angular/angular-cli/wiki">

<button mat-raised-button>

<mat-icon svgIcon="earthIcon"></mat-icon>

CLI Documentation

</button>

</a>

</mat-list-item>

<mat-list-item role="listitem">

<a class="urlToOtherWeb" target="\_blank" rel="noopener" href="https://blog.angular.io/">

<button mat-raised-button>

<mat-icon svgIcon="earthIcon"></mat-icon>

Angular blog

</button>

</a>

</mat-list-item>

</mat-list>

</mat-grid-tile>

<mat-grid-tile>

<mat-list role="list">

<mat-list-item role="listitem">

<h2>Material samples</h2>

</mat-list-item>

<mat-divider></mat-divider>

<mat-list-item role="listitem">

<a class="urlToOtherWeb" target="\_blank" rel="noopener" href="https://material.angular.io/">

<button mat-raised-button>

<mat-icon svgIcon="earthIcon"></mat-icon>

Angular Material

</button>

</a>

</mat-list-item>

<mat-list-item role="listitem">

<a class="urlToOtherWeb" target="\_blank" rel="noopener" href="https://material.angular.io/components/categories">

<button mat-raised-button>

<mat-icon svgIcon="earthIcon"></mat-icon>

Components

</button>

</a>

</mat-list-item>

<mat-list-item role="listitem">

<a class="urlToOtherWeb" target="\_blank" rel="noopener" href="https://github.com/angular/material2">

<button mat-raised-button>

<mat-icon svgIcon="earthIcon"></mat-icon>

Gihub

</button>

</a>

</mat-list-item>

</mat-list>

</mat-grid-tile>

</mat-grid-list>

</article>

</section>

1. Custom-material.ts

import { NgModule } from '@angular/core';

import {DomSanitizer} from '@angular/platform-browser';

// imports material

import {

MatToolbarModule, MatSidenavModule, MatGridListModule, MatIconRegistry

} from '@angular/material';

import {MatIconModule} from '@angular/material/icon';

import {MatDividerModule} from '@angular/material/divider';

import {MatListModule} from '@angular/material/list';

@NgModule({

imports: [

MatToolbarModule,

MatIconModule,

MatSidenavModule,

MatGridListModule,

MatDividerModule,

MatListModule

],

exports: [

MatToolbarModule,

MatIconModule,

MatSidenavModule,

MatGridListModule,

MatDividerModule,

MatListModule

]

})

export class CustomMaterial {

constructor(iconRegistry: MatIconRegistry, sanitizer: DomSanitizer) {

iconRegistry.addSvgIcon(

'earthIcon',

sanitizer.bypassSecurityTrustResourceUrl('../assets/images/earth.svg'));

iconRegistry.addSvgIcon(

'helloIcon',

sanitizer.bypassSecurityTrustResourceUrl('../assets/images/hello.svg'));

}

}

1. Styles.css

@import '~@angular/material/prebuilt-themes/purple-green.css';

* 1. Aurelia
     1. Instalación de Aurelia-Cli

Instalamos Aurelia-Cli desde línea de comandos usando el comando:

* **npm install -g aurelia-cli**
  + 1. Creación del Proyecto

Para crear el proyecto desde línea de comandos usamos el comando:

* au new

Esto nos mostrara un asistente de instalación donde mediante preguntas nos dejara elegir ciertas características.

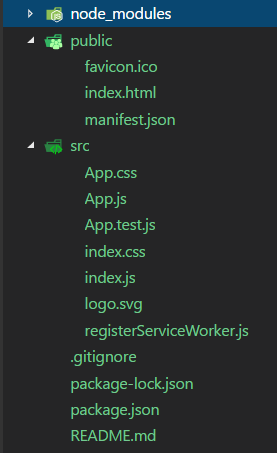
Te mostrara la información del proyecto a partir de las opciones elegidas anteriormente y te preguntara si realmente quieres crear el proyecto.

Y también te preguntara si quieres instalar las dependencias del proyecto.

* 1. React
     1. Crear proyecto

Para crear el proyecto usamos el comando desde la consola de comandos:

* **create-react-app [nombreApp]**
  + 1. Estructura del proyecto

Al crear el proyecto podemos ver como se nos genera la estructura de la izquierda.

**node\_modules:** Es la carpeta donde se genera las diferentes librerías de node y react necesarias para el desarrollo del proyecto.

**public**: En esta vemos como se generan los archivos de carácter público, como puede ser el **favicon** a usar por la aplicación como el archivo html **index.html** que será el usado por los navegadores para mostrar nuestra página web desarrollada.

El archivo **manifest.json** será un archivo json donde se especificara diferente información sobre la aplicación por ejemplo el nombre, nombre corto, iconos a usar, página de inicio, color de tema, color del fondo, etc.

**src**: En esta carpeta será donde realizaremos gran parte del desarrollo.

De inicio podemos ver un bloque formado por diferentes archivos con la palabra **App**, donde **App.js** sería el archivo JavaScript que contendrá nuestro componente principal, **App.css** es la hoja de estilos que contendrá los estilos generales orientados a los componentes y **App.test.js** será el fichero JavaScript de test.

Podemos observar el archivo **index.js** que será un JavaScript que contendrá la información para renderizar nuestra aplicación, desde él se llamará al componente principal; en este caso existente en **App.js** para cargarlo en el fichero **index.html** en el tag:

<div id="root"></div>

También tenemos el **index.css** hoja de estilos que contendrá los estilos generales a nivel de la aplicación general, en mi opinión mientras en App.css podemos tener los estilos generales para los componentes en este se podrían tener los relativos a los elementos generales de HTML como el body, footer, header, etc.

Una buena recomendación es generar una buena estructura de carpetas dentro de src para tener más ordenado y limpio nuestro proyecto.

* + 1. Ejecución de la aplicación

Una buena recomendación es generar una buena estructura de carpetas dentro de src para tener más ordenado y limpio nuestro proyecto.

Para ejecutar la aplicación podemos usar el comando:

**npm start**

Para facilitar el teste podéis usar la extensión para los navegadores como Chrome, firefox:

* **react developer tools**

Otros comandos:

Construir la aplicación: npm run build

Iniciar Test: npm test

* 1. Vue
     1. Instalación de Vue

Desde la línea de comandos de Node instalamos Vue mediante el comando:

* **npm -g install vue**
  + 1. Instalación de vue-cli

Desde la línea de comandos de Node instalamos vue-cli mediante el comando:

* **npm -g install vue-cli**
  + 1. Herramientas para el navegador

Una buena recomendación es usar la extensión vue-devtools en tu navegador.

* + 1. Creando nuestro proyecto.

Para crear nuestra aplicación una vez lo tengamos todo instalado es muy sencillo sirve con ejecutar el siguiente comando:

* **vue init webpack my-app**

Una vez ejecutado comenzara la instalación, con diferencia con otros frameworks como puede ser Angular durante la instalación te preguntara ciertas opciones de instalación.

**Nombre del proyecto:**



Si no introducimos un nombre por defecto dejara el marcado entre paréntesis que será el que coincida con el indicado en el comando usado para crear la app con vue-cli.

Por ejemplo, en mi caso, debo introducir otro nombre al usar mayúsculas, como bien podemos observar en la notificación que obtenemos por consola.



**Descripción del proyecto.**

Si en el paso anterior va todo correcto nos pedirá que le demos una descripción al proyecto.

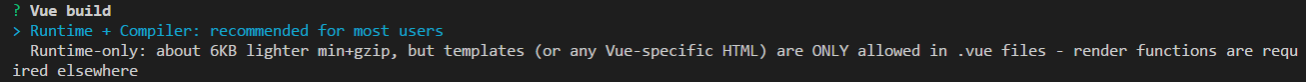


**Indicar el nombre del autor:**



**Opción de compilación:**

En este punto nos dará a elegir entre dos opciones a elegir con las flechas del teclado.



Yo he elegido la que recomienda para la mayoría de los usuarios.

**Opción de instalar router:**

En esta opción le podemos indicar si queremos que nos instale el gestor de routing de vue.



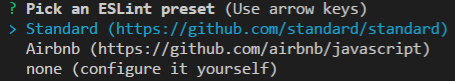
**Opción instalar ESLint:**

ESLint, para quien no lo sepa es un helper de EcmaScript para ayudar que nuestro código sea mucho más limpio ayudándonos con notificaciones mientras desarrollamos.



**Elección del tipo de ESLint a usar:**

En esta pregunta podemos elegir entre varias opciones de configuración del ESLint.

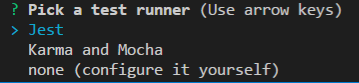


**Preparación de Unit Test**

Nos da la opción de preparar la app para el uso de Unit Test



Selección de herramienta de ejecución de los test.

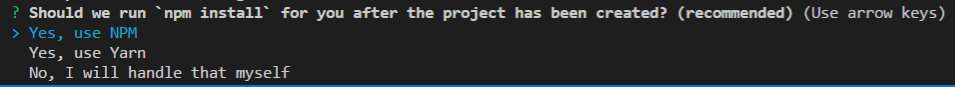


Preparar test e2e

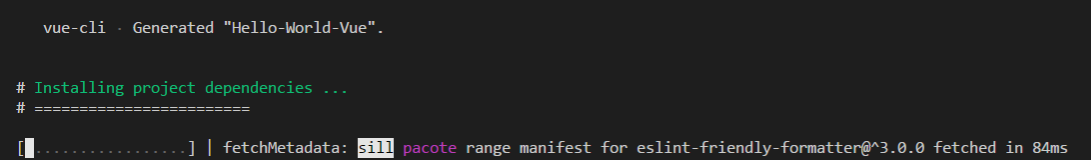


Opción de ejecutar automáticamente npm install

En este paso nos hace una pregunta interesante, ya que nos pregunta si después de la instalación queremos que nos ejecute automáticamente el comando npm install de node por el cual instalara todas las librerías necesarias teniendo en cuenta también las opciones elegidas anteriormente.

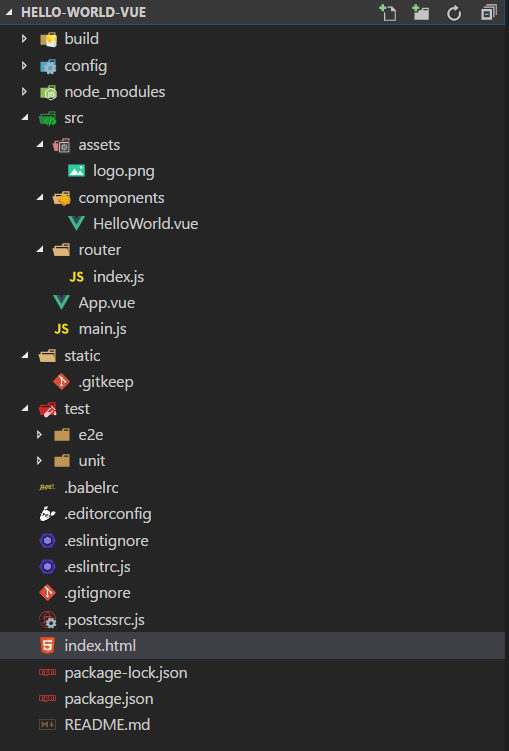


Una vez esto iniciara la creación del proyecto:



* + 1. Ya creado el proyecto

Una vez finalizado los pasos del punto 3, vue cli nos habrá creado el proyecto con la siguiente estructura (puede variar según las opciones seleccionadas).



En la carpeta src será donde vayamos realizando nuestro desarrollo de la aplicación, en ella podemos destacar las siguientes carpetas y archivos.

**Assets**: Será la carpeta para alojar recursos como imágenes, etc.

**Components**: Carpeta para ir albergando los diferentes componentes vue desarrollados.

**Router**: Tendrá el archivo js con el enrutamiento que queramos para nuestra app.

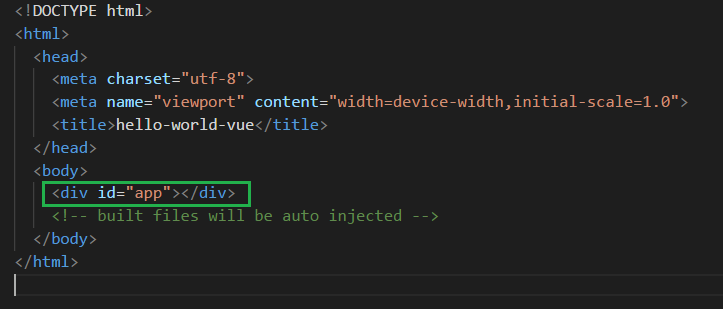
**App.vue**: Será el componente principal de la aplicación y el padre del resto de componentes será el que renderice vue y será el llamado desde el index.html

**Main.js:** Archivo js principal de configuración de la aplicación donde por ejemplo declararemos los componentes que vayamos creando entre otros.

**Index.html:** Archivo index de la aplicación punto de entrada de la aplicación y archivo principal usado por los navegadores.

* + 1. Ejemplo de archivos

**Index.html:**

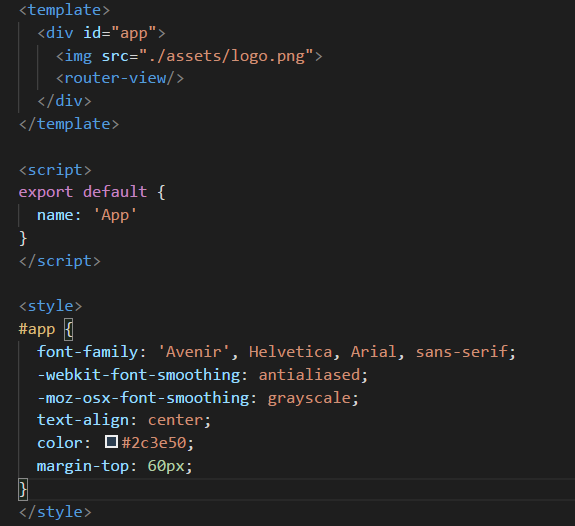


Es un archivo html común, donde nos debemos de fijar en el div con el identificador id=”app” que será usado por el archivo main.js para montar el componente principal.

**App.vue:**

Componente principal de la app, es un archivo vue, donde podemos observar y ver que llama la atención de como Vue nos aconseja que queda componente este en un único fichero conteniendo tanto el código html, javascript y css.

El código html como veis se encierra entre las etiquetas de <template>



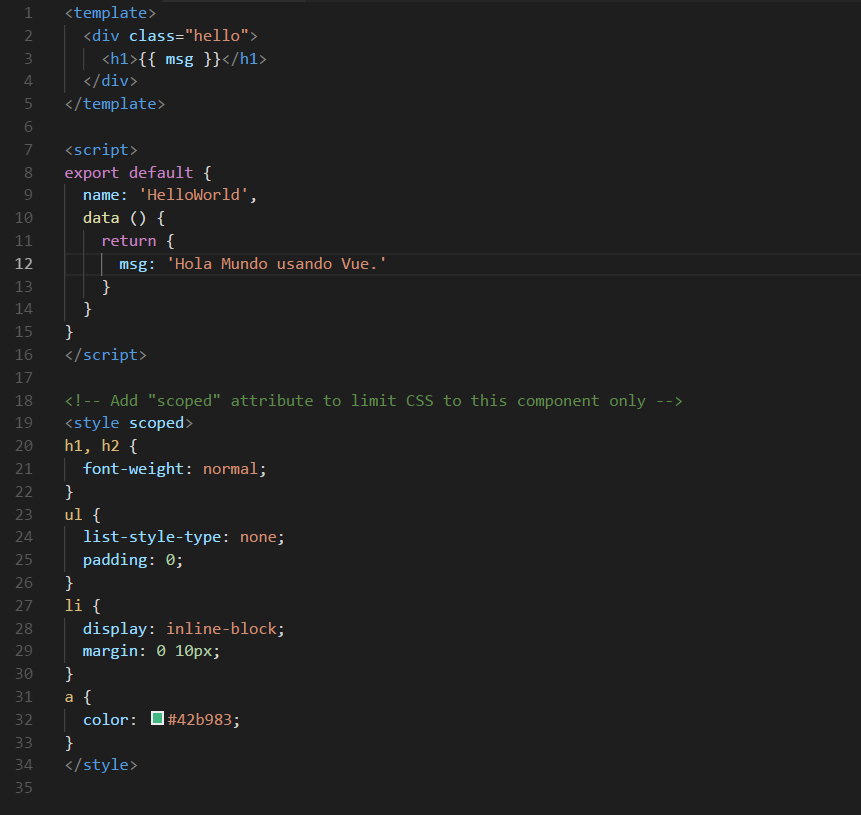
Al ser el componente principal tendrá la etiqueta:

<router-view/>

Donde se irán cargando el resto de los componentes según naveguemos por la app.

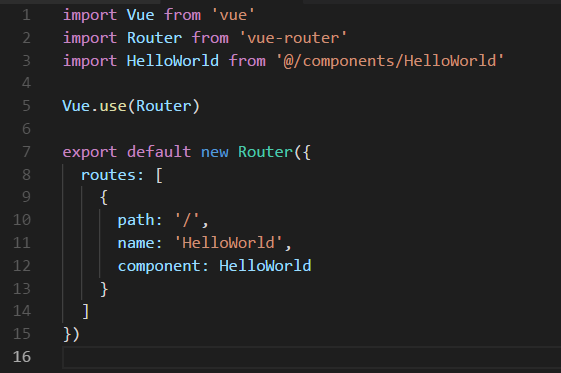
Componente secundario, **HelloWorld.vue**:

Este componente será el que contenga el contenido, en este caso el saludo típico de Hola Mundo.



Archivo **index.js** del router:

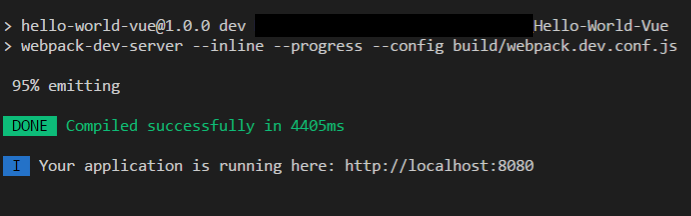
Este archivo contendrá nuestras rutas de la aplicación y en nuestro caso será el que indique que inicialmente dentro de la etiqueta router-view que está en el App.vue se cargue el anterior componente al indicarlo en la especificación como lo vemos en la siguiente imagen.



* + 1. Ejecutando la app

Para ejecutar la aplicación desde la carpeta de la misma usamos el comando:

* **npm run dev**



Abrimos el navegador y escribimos la url que nos indica en la consola de comandos.



Y podemos observar cómo nos carga nuestra app.

* 1. Ionic
     1. Preparación Entornos desarrollo Android e IOS

Posiblemente este punto puede ser el que de más “problemas” si además nunca los has preparado anteriormente.

Desde la página de Ionic Framework te recomiendan seguir las siguientes guías:

* <https://cordova.apache.org/docs/en/7.x/guide/platforms/android/>
* <https://cordova.apache.org/docs/en/7.x/guide/platforms/ios/>

Ver apartado preparación 7. Preparación Entorno de Desarrollo con Android

* + 1. Instalación de Ionic

Desde la consola de Node usamos el comando:

* **npm install -g cordova ionic**
  + 1. Creando nuestra aplicación

Una En este punto tenemos diferentes opciones:

* ionic start myApp blank 🡪 la genera vacía
* ionic start myApp tabs 🡪 genera el proyecto con el tema de tabs
* ionic start myApp sidemenu 🡪 genera el proyecto con una barra de menú

Durante la generación de la creación de la App te preguntara:

? Would you like to integrate your new app with Cordova to target native iOS and Android? (y/N)

En este punto he dicho si (opción y), con ello instalara **IONIC DEVAPP** con ello podremos testear las aplicaciones sin tener los SDKs nativos de Android e iOS (en el caso de Android recomiendan instalar AndroidStudio).

Después de esta te preguntara:

? Install the free Ionic Pro SDK and connect your app? (Y/n)

Esta opción es para instalar la versión PRO de ionic

* + 1. Ejecutar la aplicación en navegador:

Usamos el comando.

* **ionic serve** en la terminal que estemos usando.
  + 1. Añadir platform Android al proyecto:

Usamos el comando:

* **cordova platform add android –save**
  + 1. Añadir platform iOS al proyecto:

Usamos el comando:

* **cordova platform add ios –sabe**
  + 1. Ejecutar la aplicación en emulador:

Usamos el comando:

* **ionic cordova run Android --device** desde el terminal.
  + 1. Ejecutar o generar la aplicación para producción:

Usamos los comandos:

* **ionic cordova run android --prod --release**
* **ionic cordova build android --prod --release** 
  1. NativeScript
     1. Preparación Entornos desarrollo Android e IOS

Igual que en el caso de Ionic, al ser un framework para desarrollar apps móviles se deben preparar sus entornos.

Desde la página de NativeScript nos recomiendan descargar tanto para Android como para IOS desde la App Store y Google Play las siguientes aplicaciones para que se usen durante el desarrollo de la aplicación.

* **NativeScript Playground**
* **NativeScript Preview**

(Personalmente no he probado ninguna de las dos)

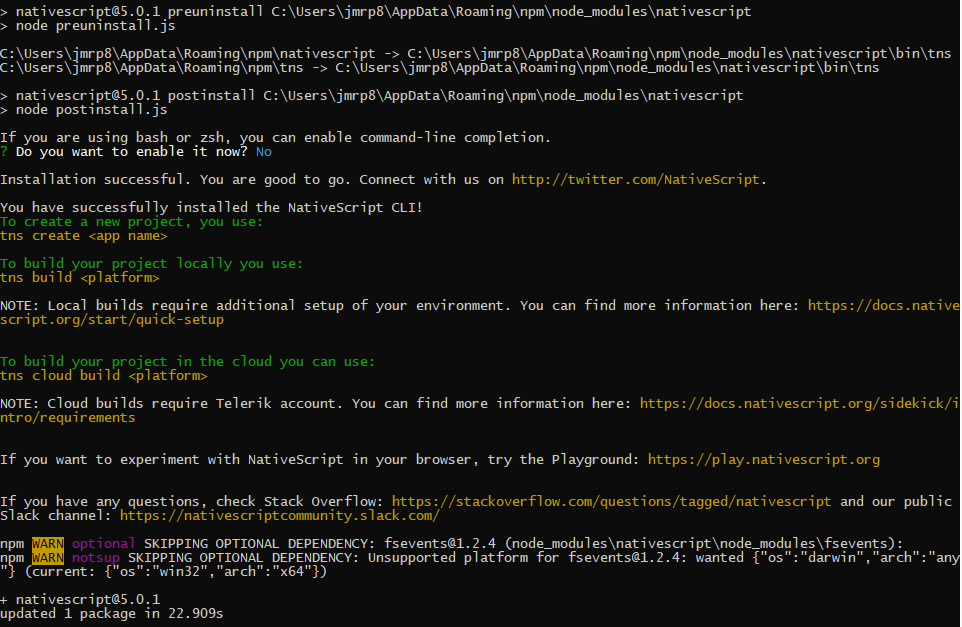
Ver apartado preparación 7. Preparación Entorno de Desarrollo con Android

* + 1. Instalación de NativeScript

Desde la línea de comandos de Node usaremos el comando:

* **npm install -g nativescript**

Resumen instalación:



Podemos comprobar el estado de la instalación mediante el comando:

* **tns doctor**
  + 1. Creando nuestra Aplicación

Desde la consola de comandos y situados en la raíz donde queramos crear el nuevo proyecto escribimos:

* **tns create HelloWorld --ng** (opción con template de angular)

Otras opciones:

* **tns create MyApp**: Crea la aplicación con el tema básico.
* **tns create MyApp --tsc**: Crea la aplicación usando el template de TypeScript
* **tns create MyApp --template nativescript-template-tutorial**: Crea la aplicación con el template del tutorial oficial.

Una vez creado abrimos la carpeta desde nuestro IDE, desde este momento las instrucciones serán referentes al uso del VSCode que nos incorpora la terminal en el propio IDE.

* + 1. Preparar las platform:

Usamos los comandos desde la consola:

* **tns prepare android** (tns prepare ios).
* **tns platform add <ios or Android>** (añadir platform)
  + 1. Ejecución de la aplicación:

Ejecutar la aplicación, por ejemplo, escribimos en la terminal:

* **tns run Android**
* **tns run Android --emulator**
* **tns run ios**

En el caso de Android si no se ejecuta y nos dice no encuentra el SDK suele ser problemas en el path.

* + 1. Dispositivos móviles disponibles:
* **tns device <platform> --available-devices** donde <platform> será Android o iOS.
* **tns device android –watch**
  + 1. Algunos errores al ejecutar en emulador:

Para ciertos problemas no reconoce el device, en mi caso era porque no estaba bien instalado “Intel HAXM”:

<https://software.intel.com/en-us/articles/intel-hardware-accelerated-execution-manager-intel-haxm>

Al hacer tns run Android --emulator he obtenido errores casi al final de la preparación de la aplicación para la instalación en el emulador.

Lo que he hecho es eliminar las platforms y ejecutar tns run Android. Y ha ido todo OK

* 1. ReactNative
     1. Preparación Entornos desarrollo Android e Ios

Igual que en el caso de Ionic y NativeScript al ser un framework para desarrollar apps móviles se deben preparar sus entornos.

Ver apartado preparación 7. Preparación Entorno de Desarrollo con Android

* + 1. Instalación de ReactNative

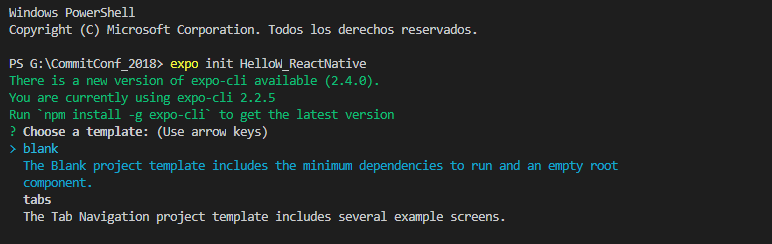
Desde la línea de comandos de Node usamos el comando que nos instalara el expo-cli usado por ReactNative:

* **npm install -g expo-cli**
  + 1. Creación de un proyecto

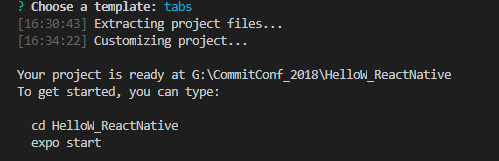
Desde la línea de comandos usamos el comando:

* **expo init NameProject** ex: expo init HelloW\_ReactNative

Nos mostrara la siguiente información para que escojamos entre el template **blank** o de **tabs**. Yo he elegido el de tabs.



Despúes de un rato (paciencia) nos mostrara la siguiente info:

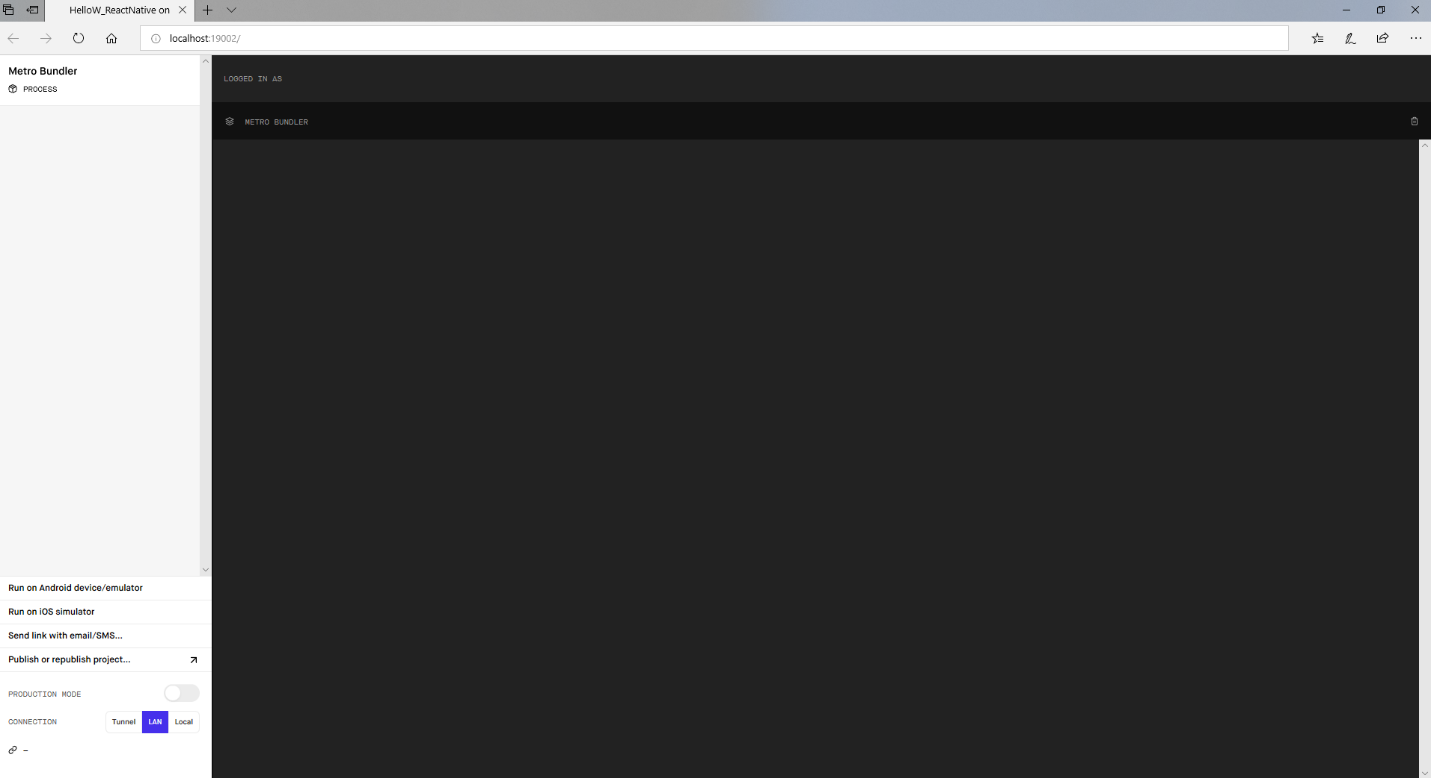


* + 1. Ejecución del proyecto

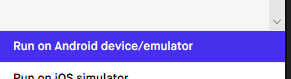
Desde la consola de comandos y la carpeta del proyecto usamos el comando:

* **npm start**

Esto nos abrirá en el navegador un asistente de expo para trabajar con la aplicación realizada. (Acordaros tener el emulador de Android ejecutado).

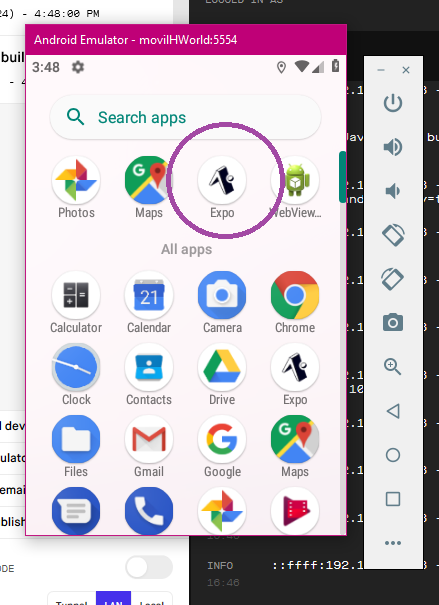


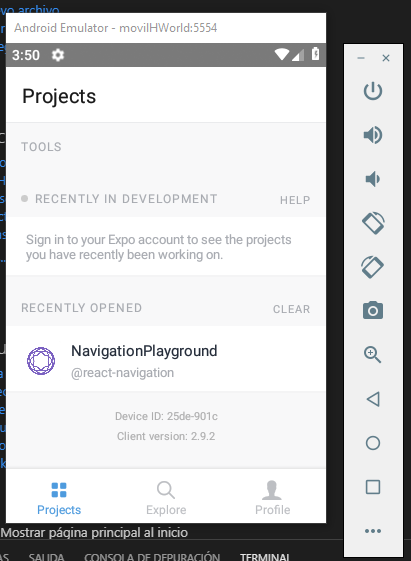
Pulsamos sobre la opción



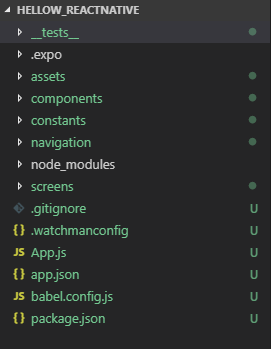
Esto nos ejecuta la aplicación sobre el emulador de Android, que aparecerá en el dispositivo bajo el símbolo y nombre de expo.

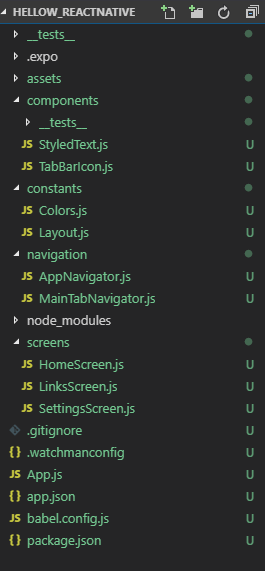
Al pulsar sobre el icono nos mostrara la aplicación con el contenido que genera por defecto el expo-cli:





* + 1. Comandos Utiles
* **npm start (inicia la web de soporte de expo)**
* **npm android (ejecutar android)**
* **npm ios (ejecutar ios)**
* **npm test (ejecutar test)**
  + 1. Arquitectura Creada





1. Preparación Entorno de Desarrollo con Android
   * 1. Introducción

Como hemos visto para los frameworks dirigidos a aplicaciones móviles es recomendable preparar el entorno para poder usar simuladores o dispositivos virtuales.

Es cierto que muchos de estos frameworks al ejecutarlos tienen simuladores vía web, vía móvil a través de apps para los dispositivos móviles.

En mi caso el entorno que he preparado ha sido solo para Android ya que como muchos podréis saber para preparar un entorno de IOS debes tener dispositivos propios de Apple.

Llegado a este punto para el entorno de Android frameworks como Ionic o NativeScript te recomiendan la instalación del IDE Android Studio.

Esto conlleva instalar un IDE que posiblemente no se vaya a usar solo para poder crear dispositivos virtuales y gestionar los SDK de Android con las herramientas **AVDManager** y **SDKManager**.

Hace unos años desde la web oficial de Android te permitían descargar por separado estas herramientas.

En mí caso he usado dos opciones usar la vía de instalar **Android Studio** y la vía del uso de **Chocolatey** (manejador de paquetes para Windows).

Debemos tener en cuenta que deberemos tener Node instalado y en el caso de Android el JDK (Java Development Kit).

* + 1. Preparación de Android con Chocolatey

**7.1.2.1 Para instalar Chocolatey pasos descritos en su web:**

Desde cmd.exe pegamos el siguiente comando:

@"%SystemRoot%\System32\WindowsPowerShell\v1.0\powershell.exe" -NoProfile -InputFormat None -ExecutionPolicy Bypass -Command "iex ((New-Object System.Net.WebClient).DownloadString('https://chocolatey.org/install.ps1'))" && SET "PATH=%PATH%;%ALLUSERSPROFILE%\chocolatey\bin"

Desde PowerShell.exe pegamos el siguiente comando:

Set-ExecutionPolicy Bypass -Scope Process -Force; iex ((New-Object System.Net.WebClient).DownloadString('https://chocolatey.org/install.ps1'))

Una vez terminada la instalación es recomendable reinicial la consola o terminal de comandos.

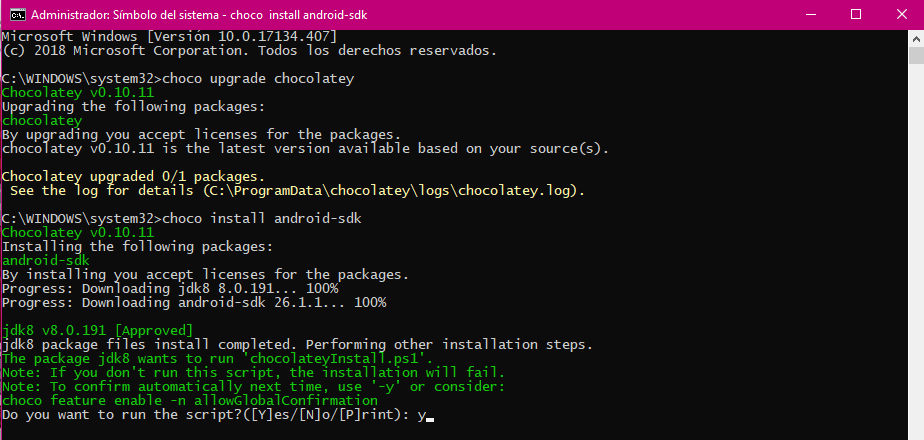
Para actualizar Chocolatey usamos el comando:

* choco upgrade chocolatey

Para instalar AndroidSDK:

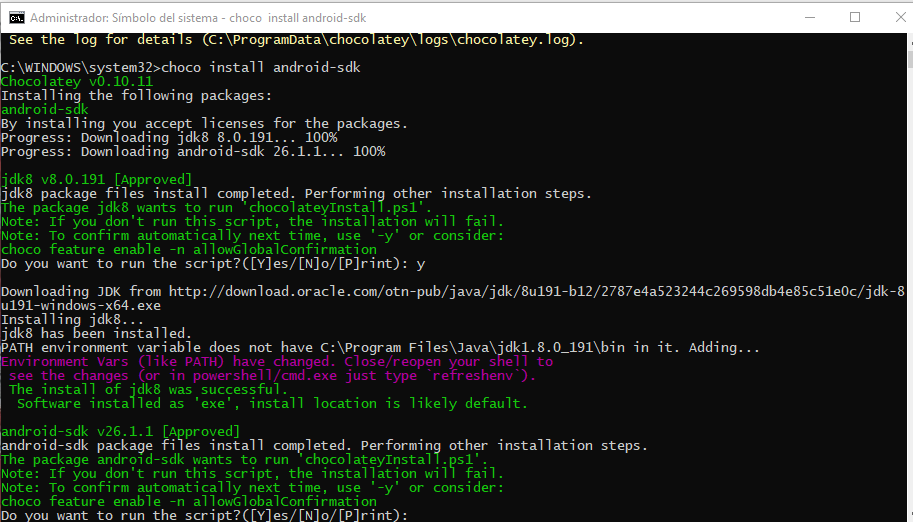
Usamos el comando:

* choco install android-sdk



Introducimos la ‘Y’ pulsamos ‘Intro’.

Comenzara verificando la instalación del JDK y luego seguirá con Android-SDK.



Volvemos a introducir la ‘y’ y pulsamos intro.

Para actualizar AndroidSDK usamos el comando:

* choco upgrade android-sdk

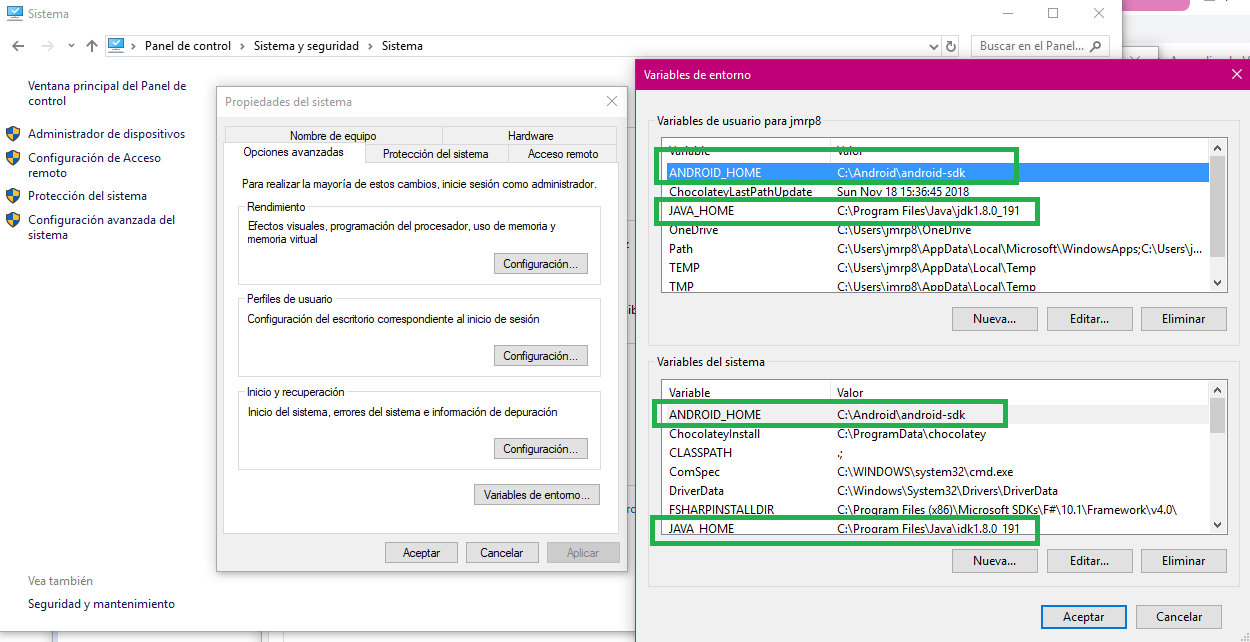
En el caso del proyecto de NativeScript, una vez seguido los anteriores pasos y habiendo reiniciado el terminal.

Instalacion de Gradle usamos el comando:

* choco install gradle

**7.1.2.2 Preparación AVDs (Dispositivos virtuales):**

Comprobar las variables del entorno.



Como crear desde línea de comandos un dispositivo virtual:

Desde la carpeta de tools/bin en Android/Android-sdk

Preparamos el sdk de la versión 25 o 28 (La versión 25 la última vez me dio algún problema)

Ejecutamos:

* sdkmanager "system-images;android-25;google\_apis;x86"
* sdkmanager "system-images;android-28;google\_apis;x86"



Si todo ha ido correctamente nos debería crear la carpetar emulator dentro de la carpeta Android-sdk.

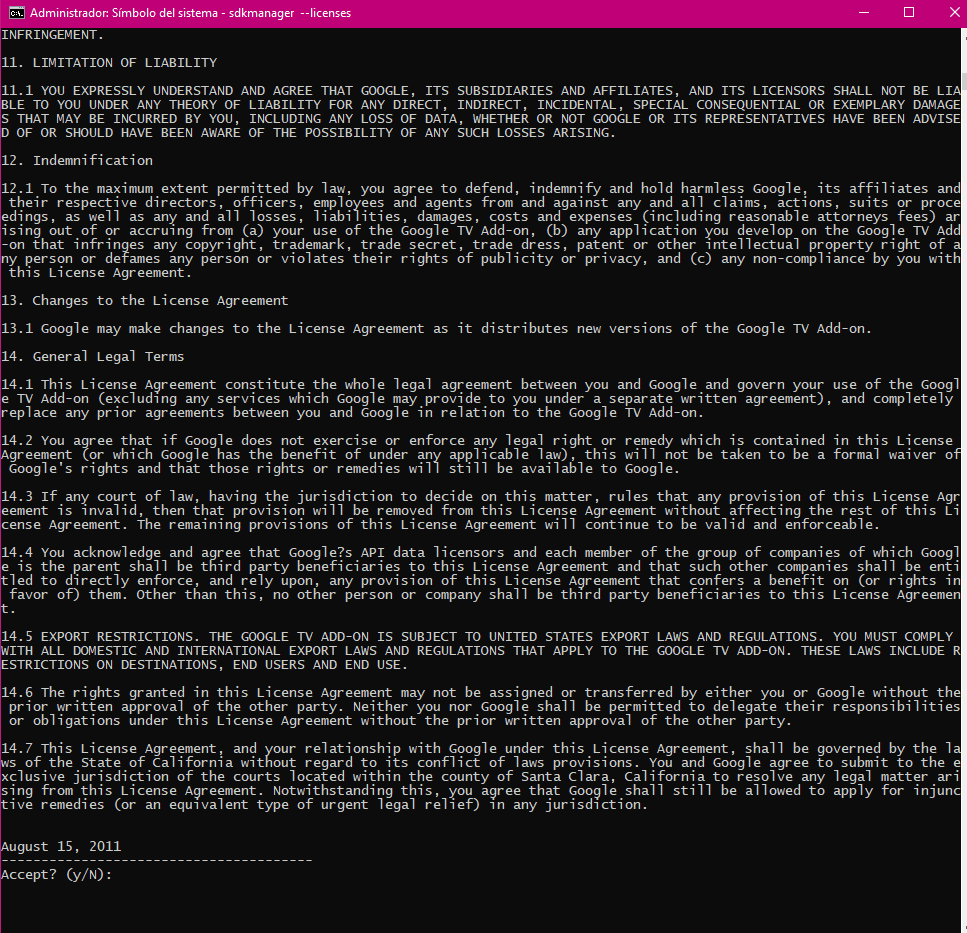
Escribimos el comando:

* sdkmanager –licenses



Escibimos ‘y’

Nos sale:



Y aceptamos escribiendo ‘Y’, y así se repite unas cinco veces.

avdmanager create avd -n name -k "sdk\_id" [-c {path|size}] [-f] [-p path]

ex: avdmanager create avd -n movilHWorld -k "system-images;android-28;google\_apis;x86" -f



Desde la carpeta android\android-sdk\emulator>

listar dispositivos: emulator -list-avds

ejecutar dispositivo: emulator @movilHWorld

* + 1. Preparación de Android con Android Studio

Esta forma puede ser la más sencilla, aunque no se use el IDE, siguiendo los pasos de la instalación desde la web:

* <https://developer.android.com/studio/>

Una vez instalado, sin necesidad de crear un proyecto desde la pantalla de inicio del IDE en la parte inferior podemos acceder a la configuración y desde ella a las opciones de SDK y tools para Android.

Pero para crear un dispositivo virtual yo he creado un proyecto vacío y una vez abierto he accedido al AVDManager para crearlo y lanzarlo.

También se puede hacer desde la terminal de Windows con los mismos comandos que los del apartado anterior sobre chocolatey.

* + 1. Emulador de Windows

Como Para Windows 8 o superior se puede usar alternativamente su emulador de Android, (no lo he probado), para más info:

<https://www.visualstudio.com/vs/msft-android-emulator/>

1. Bibliografía

Enlaces interesantes para tener en cuenta o donde se ha obtenido información.

* Visual Studio Code: <https://code.visualstudio.com/>
* Node: <https://nodejs.org/es/>
* npm: <https://www.npmjs.com/>
* Typescript: <https://www.typescriptlang.org/>
* ESLint: <https://eslint.org/>
* TSLint: <https://palantir.github.io/tslint/>
* Angular: <https://angular.io/>
* Angular – Material: <https://material.angular.io/>
* Reactjs: <https://reactjs.org/>
* NPM: <https://www.npmjs.com/>
* React Developer tools extensión:
* Chrome: <https://chrome.google.com/webstore/detail/react-developer-tools/fmkadmapgofadopljbjfkapdkoienihi>
* Mozilla: <https://addons.mozilla.org/es/firefox/addon/react-devtools/>
* VueJS: <https://vuejs.org/>
* vue-devtools: <https://github.com/vuejs/vue-devtools#vue-devtools>
* Vue.JS school: <https://vueschool.io/?utm_source=Vuejs.org&utm_medium=Banner&utm_campaign=Sponsored%20Banner&utm_content=V1>
* Presentación de jdonsan “charla-aprendiendo-vuejs”:

<https://jdonsan.github.io/charla-aprendiendo-vuejs/slides/#slide=1>

* Aurelia: <https://aurelia.io/>
* Ionic: <https://ionicframework.com/>
* Nativescript: <https://www.nativescript.org/>
* ReactNative: <https://facebook.github.io/react-native/>
* Android Developer: <https://developer.android.com/studio/#downloads>
* Apple Developer: <https://developer.apple.com/develop/>
* Chocolatey Android SDK: <https://chocolatey.org/packages/android-sdk>

1. Control de versiones

|  |  |  |
| --- | --- | --- |
| Versión | Descripción | Fecha |
| 1.0 | Versión Inicial | 10/11/2018 |
|  |  |  |