**Crear un nuevo proyecto Angular:**

**rama001**

Hay que tener instalado nodejs y angular a nivel global con:

***npm install -g @angular/cli***

A partir de esto tenemos a disposición las directivas ng de angular.

Para crear un nuevo proyecto y carpeta a partir de la carpeta actual. Nos crea todo el entorno e inicializa el git.

***> ng new angularSampleProject01***

La creación tarda unos minutos.

Después de crearla y borrar el node\_modules la hemos instalado con *npm install* es lo que habría que hacer si la bajamos del repositorio. También la instalación tarde unos minutos. Son muchos los módulos que requiere Angular, node\_modules ocupa unos 450MB.

Usaremos, aunque desde cero, creando el proyecto, el ejemplo de la página oficial de Angular:

[*https://angular.io/start*](https://angular.io/start)

Hemos creado la carpeta doc con este documento.

Crea un README.md. Escribiremos al principio de este docuemento.

Subiremos a GitHub usando distintas ramas para tener los pasos separados.

Generamos un nuevo componente:

*ng generate component top-bar*

Y sustituimos el contenido original de app.component.html por:

<app-top-bar></app-top-bar>

<div class="container">

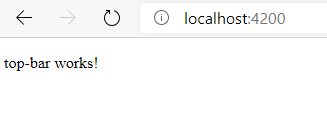
  <router-outlet></router-outlet>

</div>

Ya podemos compilar y arrancar el servidor con:

> ng serve

E invocarlo en un navegador, veremos el mensaje por defecto del contenedor creado:



Como el repositorio ya está creado lanzamos:

*git add .*

*git commit -m "first commit"*

*git branch -M rama001*

*git remote add origin https://github.com/\*\*\*\*\*\*\*\*/angularSampleProject01.git*

*git push -u origin rama001*

**rama002**

Dejamos el archivo **styles.css,** que originalmente está vacío, con el contenido que podemos ver en el propio archivo. A

//Para ver a donde apunta cada rama

git log --oneline --decorate

// Crear una nueva rama

git branch rama002

// Cambiar de rama

git checkout rama002

git add .

git commit -m "en nueva rama"

git push -u origin rama002

Añadimos al archivo index.html

  <link

  href="https://fonts.googleapis.com/icon?family=Material+Icons"

  rel="stylesheet"

/>

Para que encuentre el icono del carrito de la compra.

Cambiamos el contenido por defecto de *top-bar.component.html* y ponemos:

<a [routerLink]="['/']">

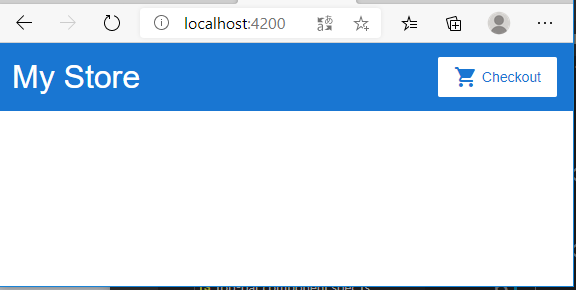
    <h1>My Store</h1>

  </a>

  <a class="button fancy-button"><i

    class="material-icons">shopping\_cart</i>Checkout</a>

Con esto, arrancamos el servidor de desarrollo con ***ng serve*** y podremos ver:



Para clonar un repositorio seleccionando una rama y seguir trabajando en otro equipo:

> *git clone --branch rama02* [*https://github.com/\*\*\*\*\*/angularSampleProject01.git*](https://github.com/*****/angularSampleProject01.git)

Con esto lo tendríamos en otro equipo y se puede seguir trabajando y subiéndolo.

* Primero instalarlo con ***npm install***

**Vamos a subir a heroku** la aplicación que tenemos en la rama002 que nos hemos bajado porque se empezó a desarrollar en otro equipo:

El método lo vemos en estas dos direcciones:

<https://medium.com/@roliver_javier/como-desplegar-tu-aplicacion-de-angular-en-heroku-7b9941b6d39>

Aquí una muy buena explicación: <https://medium.com/better-programming/how-to-deploy-your-angular-9-app-to-heroku-in-minutes-51d171c2f0d>

**Primero**: Instalar express

***npm i express –save***

**Segundo:** Crear en el directorio raiz, a nivel de tsconig.json el archivo ***server.js*** que no existía:

const express = require('express')

const app = express()

app.use(express.static('./dist/angularSampleProject01'));

app.get('/\*', function(req, res) {

    res.sendFile('index.html', {root: 'dist/angularSampleProject01/'}

  );

});

const port = 3500;

app.listen(process.env.PORT || port, () => {

    console.log(`Example app listening at http://localhost:${port}`)

})

**Tercero:** Cambiar estas dos líneas en el ***package.json***

    "start": "node server.js",

    "startServe": "ng serve",

**A partir de ahora**:

El servidor en desarrollo se sigue arrancando con

***ng serve*** (El servidor que trae angular)

Y será el que usemos porque se inicializa si hacemos cambios y todas las ventajas, pero en producción, en heroku cuando despleguemos será:

***npm start*** (El servidor express que hemos incorporado) Se puede probar que funciona en desarrollo pero no se inicializa si hacemos cambios.

**Antes de desplegar en heroku tenemos que subir a nuestro github** ya que desplegamos desde GitHub:

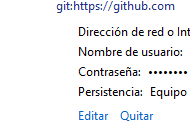
Como el repositorio ya existe basta con:

git add .

git commit -m "first commit"

git push -u origin rama002

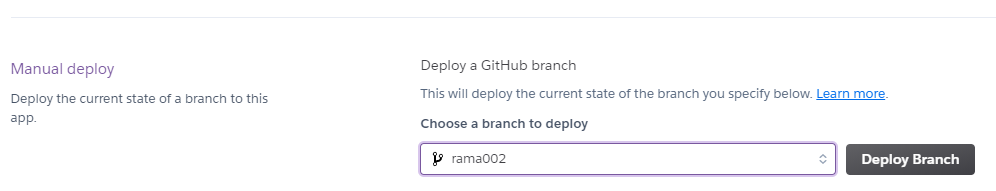
Si tuviéramos credenciales guardadas de una cuenta anterior de github hay que borrarlas previamente en Credenciales de Windows del usuario:



Si queremos ver el logs que se genera:

> ***heroku logs --app angularsampleproject01 --tail***

**Una vez creada la aplicación en heroku, si no existía, desplegamos desde la rama deseada**:



Este es el log del despliegue. Observar que hace el build por lo que no es necesario compilar en desarrollo.

-----> Node.js app detected

-----> Creating runtime environment

NPM\_CONFIG\_LOGLEVEL=error

NODE\_ENV=production

NODE\_MODULES\_CACHE=true

NODE\_VERBOSE=false

-----> Installing binaries

engines.node (package.json): unspecified

engines.npm (package.json): unspecified (use default)

Resolving node version 12.x...

Downloading and installing node 12.19.0...

Using default npm version: 6.14.8

-----> Restoring cache

- node\_modules

-----> Installing dependencies

Installing node modules

> fsevents@1.2.13 install /tmp/build\_eb9e4367\_/node\_modules/webpack-dev-server/node\_modules/fsevents

> node install.js

Skipping 'fsevents' build as platform linux is not supported

> fsevents@1.2.13 install /tmp/build\_eb9e4367\_/node\_modules/watchpack-chokidar2/node\_modules/fsevents

> node install.js

Skipping 'fsevents' build as platform linux is not supported

> core-js@3.6.4 postinstall /tmp/build\_eb9e4367\_/node\_modules/core-js

> node -e "try{require('./postinstall')}catch(e){}"

> @angular/cli@10.1.7 postinstall /tmp/build\_eb9e4367\_/node\_modules/@angular/cli

> node ./bin/postinstall/script.js

added 1504 packages in 34.737s

-----> Build

Running build

> angular-sample-project01@0.0.0 build /tmp/build\_eb9e4367\_

> ng build

Compiling @angular/core : es2015 as esm2015

Compiling @angular/common : es2015 as esm2015

Compiling @angular/platform-browser : es2015 as esm2015

Compiling @angular/platform-browser-dynamic : es2015 as esm2015

Compiling @angular/router : es2015 as esm2015

chunk {main} main.js, main.js.map (main) 13.4 kB [initial] [rendered]

chunk {polyfills} polyfills.js, polyfills.js.map (polyfills) 141 kB [initial] [rendered]

chunk {runtime} runtime.js, runtime.js.map (runtime) 6.15 kB [entry] [rendered]

chunk {styles} styles.js, styles.js.map (styles) 17.7 kB [initial] [rendered]

chunk {vendor} vendor.js, vendor.js.map (vendor) 2.29 MB [initial] [rendered]

Date: 2020-11-11T18:59:38.461Z - Hash: 84c336e9dd6119e50a14 - Time: 11423ms

-----> Caching build

- node\_modules

-----> Pruning devDependencies

removed 1423 packages and audited 81 packages in 19.289s

found 0 vulnerabilities

-----> Build succeeded!

-----> Discovering process types

Procfile declares types -> (none)

Default types for buildpack -> web

-----> Compressing...

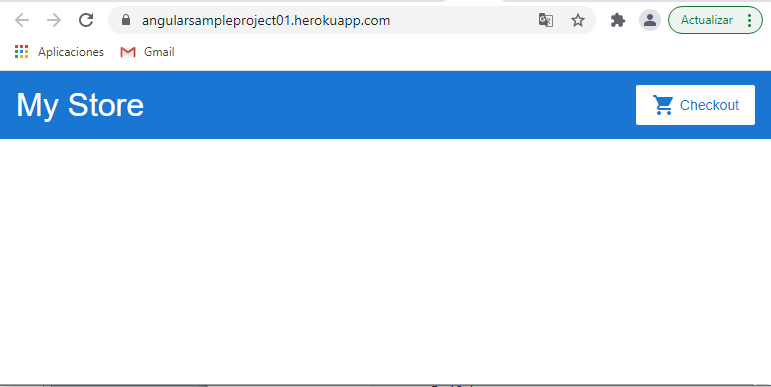
Done: 40.6M

-----> Launching...

Released v5

https://angularsampleproject01.herokuapp.com/ deployed to Heroku

Y ya podemos ver nuestra aplicación en heroku:



**Creamos la rama003 para seguir con el proyecto**

git branch rama003

// Cambiar de rama

git checkout rama003

git add .

git commit -m "en nueva rama"

git push -u origin rama003

Seguimos con nuevos componentes.

*ng generate component products-list*

Y modificar app.modules.ts con:

import { RouterModule } from '@angular/router';

y

    RouterModule.forRoot([

      { path: '', component: ProductListComponent },

    ])

import { RouterModule } from '@angular/router';

import { ReactiveFormsModule } from '@angular/forms';

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

import { TopBarComponent } from './top-bar/top-bar.component';

import { ProductListComponent } from './product-list/product-list.component';

@NgModule({

  imports: [

    BrowserModule,

    ReactiveFormsModule,

    RouterModule.forRoot([

      { path: '', component: ProductListComponent },

    ])

  ],

<!--

  Para una explicación de las rutas

  https://desarrolloweb.com/articulos/introduccion-sistema-routing-angular.html

  https://codingpotions.com/angular-componentes-routing

  router-outlet

  se sustituye por la ruta en la que estoy

  en este caso estoy en la ruta vacía, ninguna ""

  por eso se muestra el contenedor asociado a la ruta ""

  esto está definido en:

      RouterModule.forRoot([

      { path: "", component: ProductListComponent },

      { path: "products/:productId", component: ProductDetailsComponent }

    ])

  Sería como poner   <app-product-list></app-product-list>

  que son las etiquetas, selector del contenedor product-list

-->

Efectivamete funciona igual añadiendo *<app-product-list>* en *app.comonent.thml:*

<app-top-bar></app-top-bar>

<app-product-list></app-product-list>