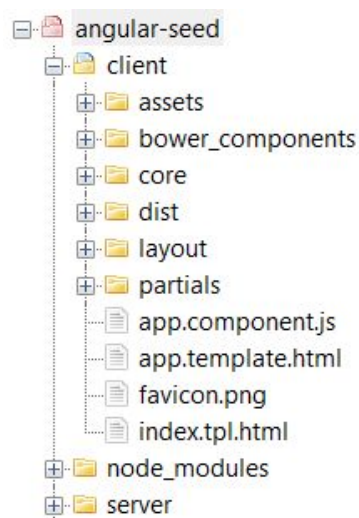
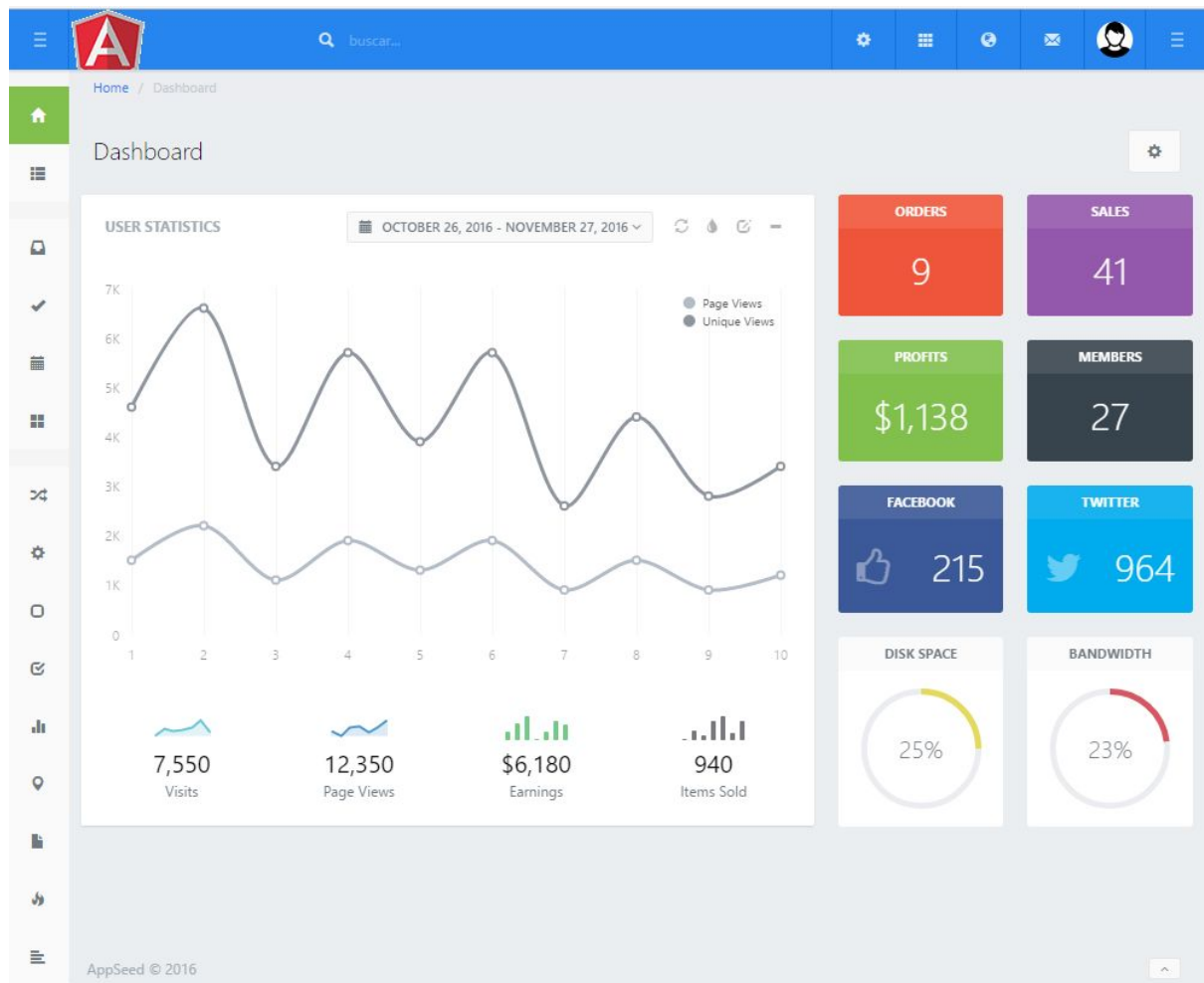


Angular-seed



Instalación

Requisitos mínimos

Para producción:

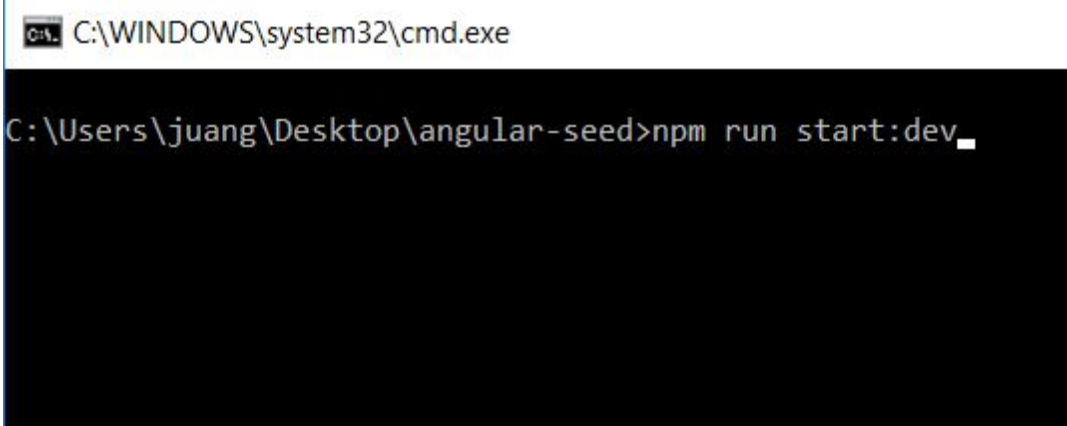
- Servidor HTTP

Para desarrollo:

- Node/NPM. Descargar e instalar Node.js: <https://nodejs.org/es/>
- Grunt. Desde una ventana de comandos ejecutar `npm install grunt-cli -g`.
- Git (opcional):
 - Descargar e instalar [git][git-setup].
 - Descargar el proyecto en una carpeta de nuestro sistema utilizando desde una consola de comandos:
 - `git clone https://github.com/coitumur/angular-seed`
- o descargar zip desde <https://github.com/coitumur/angular-seed>
- Base de Datos y Herramientas
 - Descargar e instalar el servidor [PostgreSQL][pgsql] . Cuando el asistente de instalación solicite la contraseña para el usuario 'admin' le ponemos: '1234'.
 - Descargar e instalar la herramienta [pgAdmin][pgadmin] para administración y desarrollo del servidor PostgreSQL.
 - Conectar pgAdmin al servidor PostgreSQL y ejecutar el script *esquemaDB.sql*

Desarrollo

Para empezar a desarrollar, desde una ventana de comandos nos colocamos en el directorio principal de la aplicación, en este caso el directorio **app-seed** y ejecutamos el comando: **`npm run start:dev`**



```
C:\WINDOWS\system32\cmd.exe

C:\Users\juang\Desktop\angular-seed>npm run start:dev_
```

Esta tarea **grunt**, realiza varias acciones sobre el código fuente:

- **clean**. Limpia los directorios temporales.
- **less:dev**. Procesa los archivos less y genera archivos css.
- **includeSource**. Procesa el archivo index.tpl.html en busca de comentarios especiales como:

```
<!-- include: "type": "js", "files": ["*.js",".tmp/templates.js","core/**/*.js","layout/**/*.js","partials/**/*.js"] -->
```

y a partir de la información que contienen, los reemplaza por etiquetas **<script>** para cada uno de los archivos a que dan lugar las expresiones en “files”. Como resultado genera el archivo **index.html**.

- **bowerInstall**. Procesa el archivo **index.html** en busca de los comentarios especiales:

```
<!-- bower:css --><!-- endbower -->
```

```
<!-- bower:js --><!-- endbower -->
```

reemplazandolos por etiquetas **<script>** para los archivos del tipo correspondiente (bower:tipo) que hay en la carpeta **client/bower_components**. Para saber qué archivo del

componente bower tiene que enlazar, utiliza el valor del atributo “main” del archivo bower.json que todos componentes bower tienen. El archivo bower.json de algunos componentes no tienen el atributo “main”. En este caso, es necesario incluir en el archivo bower.json que hay en el directorio principal de la aplicación, el atributo “overrides”, en el que pondremos el nombre del componente bower y el atributo “main” con el nombre del archivo que contiene el componente.

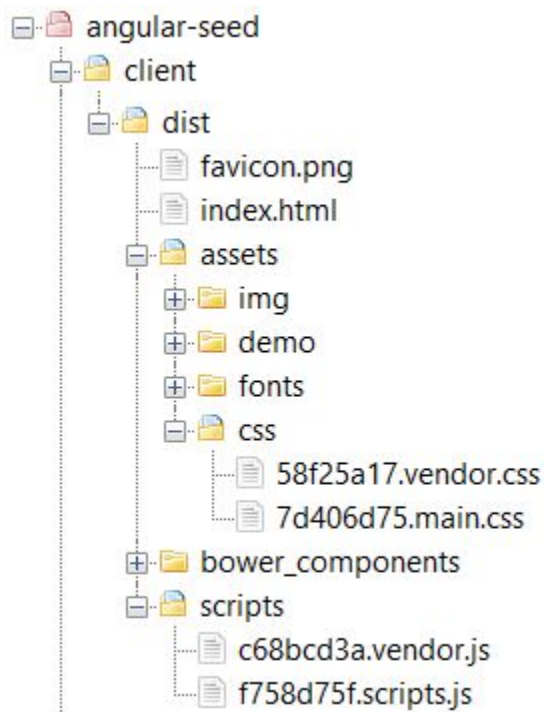
Ejemplo:

```
"overrides":{
  "jquery.easy-pie-chart": {
    "main": "dist/angular.easypiechart.js"
  }
}
```

- **concurrent:runWatchers**. Arranca el servidor HTTP y la tarea **watch**, que queda a la escucha para que cuando añadamos, modifiquemos o eliminemos algún archivo lance las tareas adecuadas.

Distribución

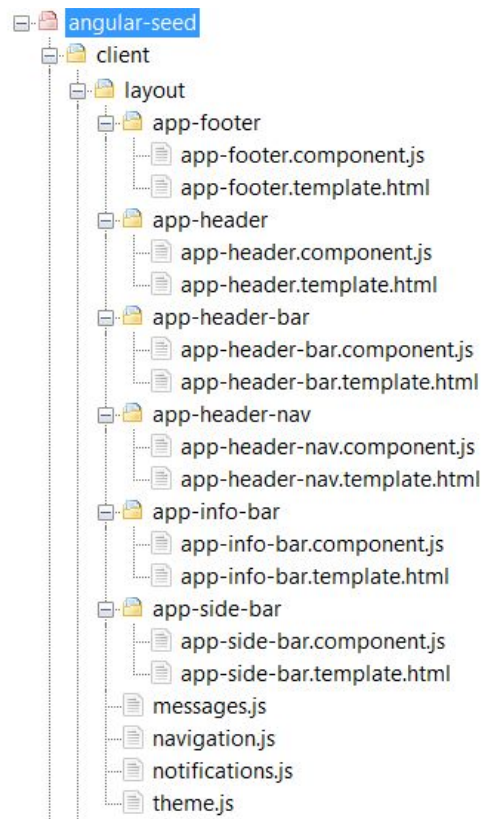
Para crear una versión de distribución, desde una ventana de comandos nos colocamos en el directorio principal de la aplicación, en este caso el directorio **app-seed** y ejecutamos el comando **npm run start:dist**, que creará una carpeta **/client/dist** con el contenido de la distribución:



Esta tarea **grunt**, realiza varias acciones sobre el código fuente:

- **clean**. Limpia los directorios temporales.
- **less:dev**. Procesa los archivos less y genera archivos css.
- **includeSource**. Procesa el archivo index.tpl.html en busca de comentarios especiales como:
`<!-- include: "type": "js", "files": ["*.js", ".tmp/templates.js", "core/**/*.js", "layout/**/*.js", "partials/**/*.js"] -->`
y a partir de la información que contienen, los reemplaza por etiquetas **<script>** para cada uno de los archivos a que dan lugar las expresiones en "files". Como resultado genera el archivo **index.html**.
- **bowerInstall**. Procesa el archivo **index.html** en busca de los comentarios especiales:
`<!-- bower:css --><!-- endbower -->`
- **useminPrepare**
- **concurrent:dist**
- **concat**. Concatena los archivos js y los archivos css.
- **ngmin**. Añade anotaciones para que la inyección de dependencias de Angular funcione con la minificación.
- **copy:dist**
- **copy:fonts**
- **cssmin**. Minifica los archivos css.
- **uglify**. Minifica los archivos js.
- **rev**. Añade resumen md5 a los nombres de los ficheros.
- **usemin**
- **clean:tmp**
- **exec:run_server**

Layout



Marco Principal

Archivo **index.tpl.html**:

```
<!DOCTYPE html>
<html lang="en">
  <head><!--insert all stylesheet links --></head>
  <body>
    <app></app>
    <script><!--insert all javascript links --></script>
  </body>
</html>
```

Componente **app**:

- client/app.component.js
- client/app.template.html:

```
<div id="app">
  <app-header></app-header>
  <app-header-bar></app-header-bar>
  <nav id="headernav">
    <app-header-nav></app-header-nav>
  </nav>
  <div id="wrapper">
    <div id="layout-static">
      <div class="static-sidebar-wrapper">
        <app-side-bar></app-side-bar>
      </div>
      <div class="static-content-wrapper">
        <div class="static-content">
          <div id="wrap" ng-view </div>
        </div>
        <footer>
          <app-footer></app-footer>
        </footer>
      </div>
    </div>
  </div>
  <app-info-bar></app-info-bar>
</div>
```

Componente **app-header**:

- client/layout/app-header.component.js
- client/layout/app-header.template.html

Componente **app-header-bar**:

- client/layout/app-header-bar.component.js
- client/layout/app-header-bar.template.html

Componente **app-header-nav**:

- client/layout/app-header-nav.component.js
- client/layout/app-header-nav.template.html

Componente **app-side-bar**:

- client/layout/app-side-bar.component.js
- client/layout/app-side-bar.template.html

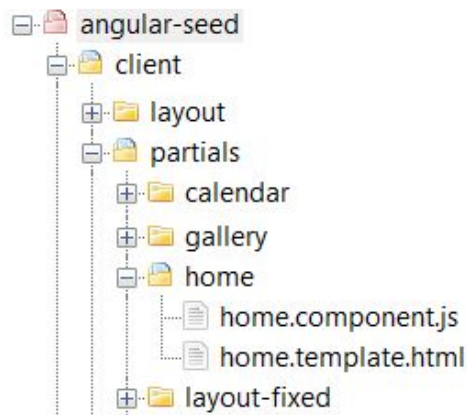
Componente **app-footer**:

- client/layout/app-footer.component.js
- client/layout/app-footer.template.html

Componente **app-info-bar**:

- client/layout/app-info-bar.component.js
- client/layout/app-info-bar.template.html

Vistas Parciales



La vista parciales están implementadas con componentes alojados en el directorio **client/partial**s.

En el archivo **component.js** de las vistas parciales, se añade la dependencia del módulo creado, al módulo principal de la aplicación:

```
angular.module('module_name', []);

angular.module('app').requires.push('module_name');

angular.module('module_name')
  .component('component_name', {
    templateUrl: 'path_to_template/template.html',
    controller: function($scope, $modal, $timeout) {

    }
  });
```

No es necesario agregar la ruta al método config del módulo principal de la aplicación, existe una ruta dinámica que intenta instanciar el componente en función del parámetro pasado en la ruta:

```
angular.module('app', []);
.config(function($provide, $routeProvider) {
  $routeProvider
    .when('/', {
      template: '<home></home>',
      resolve: {
        loadCalendar: ['$ocLazyLoad', function($ocLazyLoad) {
          return $ocLazyLoad.load([
            'bower_components/fullcalendar/fullcalendar.js',
          ]);
        }]
      }
    })
    .when('/:component', {
      template: function(param) {
        return '<' + param.component + '></'+ param.component + '>';
      }
    })
});
```

Existen vista parciales que cargan de forma lazy sus dependencias:

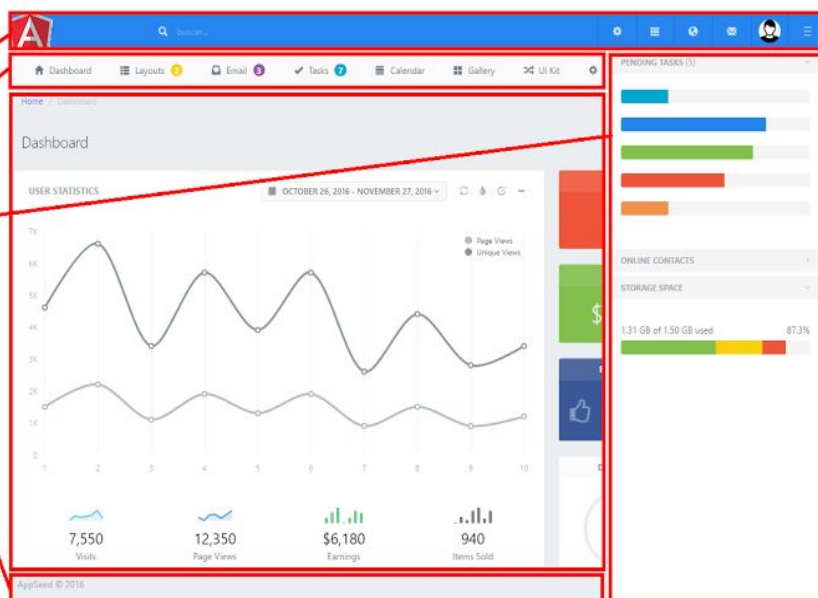
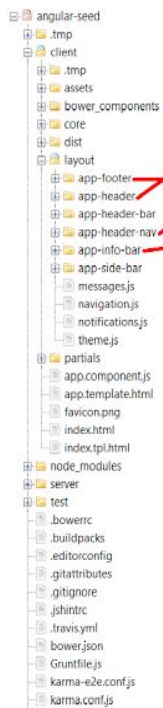
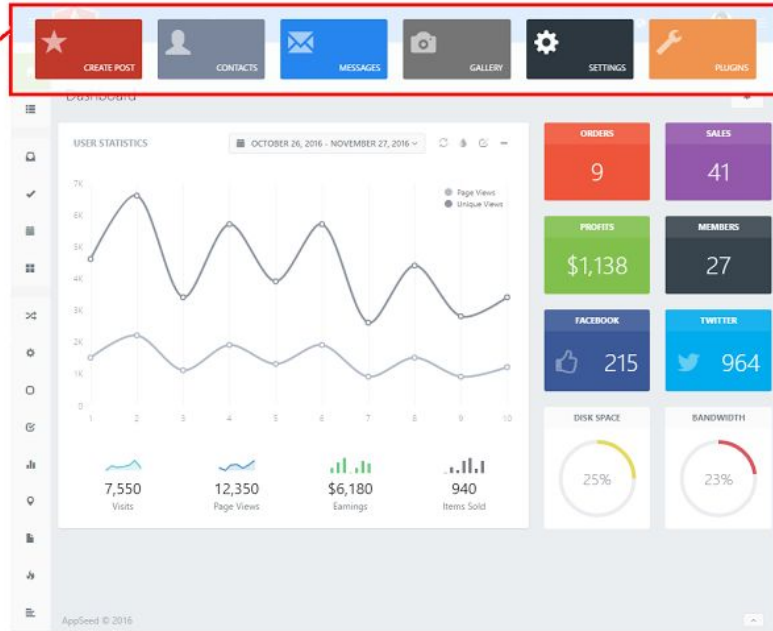
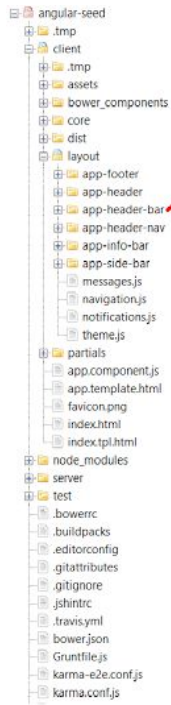
```
angular.module('module_name', []);

angular.module('app').requires.push('module_name');

angular.module('module_name')
.config(['$routeProvider', function($routeProvider) {
  $routeProvider
    .when('/component_name', {
      template: '<component_name></component_name>',
      resolve: {
        loadFile: ['$ocLazyLoad', function($ocLazyLoad) {
          return $ocLazyLoad.load([
            'path_to_file/file_name.tipe'
          ]);
        }]
      }
    })
});

.component('component_name', {
  templateUrl: 'path_to_file/template_name.template.html',
  controller: function($scope, $timeout, $theme) {

  }
});
```

Despliegue

Crear cuenta en <https://heroku.com>

Free account

Create apps, connect databases and add-on services, and collaborate on your apps, for free.

Your app platform

A platform for apps, with app management & instant scaling, for development and production.

Deploy now

Go from code to running app in minutes. Deploy, scale, and deliver your app to the world.

First name*

Last name*

Email Address*

Company name

Country*

Spain

Primary Development Language*

Node.js

CREATE FREE ACCOUNT

Signing up signifies that you have read and agree to the [Terms of Service](#) and our [Privacy Policy](#).

Cuando haya concluido el proceso de creación de la cuenta, descargar el heroku CLI :
<https://devcenter.heroku.com/articles/getting-started-with-nodejs#set-up>

Getting Started on Heroku with Node.js

Introduction
Set up
Prepare the app
Deploy the app
View logs
Define a Procfile
Scale the app
Declare app dependencies
Run the app locally
Push local changes
Provision add-ons
Start a console

Set up

In this step you will install the Heroku Command Line Interface (CLI), formerly known as the Heroku Toolbelt. You will use the CLI to manage and scale your applications, to provision add-ons, to view the logs of your application as it runs on Heroku, as well as to help run your application locally.

[Download the Heroku CLI for Windows](#)

Once installed, you can use the `heroku` command from your command shell.

On Windows, start the Command Prompt (cmd.exe) or Powershell to access the command shell.

Log in using the email address and password you used when creating your Heroku account:

```
$ heroku login
Enter your Heroku credentials.
Email: zeke@example.com
Password:
***
```

Una vez descargado e instalado, usamos el comando **heroku login** desde la línea de comandos para loguearnos usando el email y password que hemos usado para crear la cuenta en Heroku:

```
C:\WINDOWS\system32\cmd.exe
Microsoft Windows [Versión 10.0.14393]
(c) 2016 Microsoft Corporation. Todos los derechos reservados.

C:\Users\juang>heroku login
Enter your Heroku credentials.
Email: juan.guirao@hotmail.com
Password (typing will be hidden):
Logged in as juan.guirao@hotmail.com

C:\Users\juang>
```

Creamos una app sobre Heroku, con el comando **heroku create appName**. Esto prepara a Heroku para recibir nuestro código fuente. También podemos crear la app desde la web de Heroku: <https://dashboard.heroku.com/apps>.

```
C:\WINDOWS\system32\cmd.exe
Microsoft Windows [Versión 10.0.14393]
(c) 2016 Microsoft Corporation. Todos los derechos reservados.

C:\Users\juang>heroku login
Enter your Heroku credentials.
Email: juan.guirao@hotmail.com
Password (typing will be hidden):
Logged in as juan.guirao@hotmail.com

C:\Users\juang>heroku create angular-seed
Creating angular-seed... !
! Name is already taken

C:\Users\juang>heroku create app-jgp
Creating app-jgp... done
https://app-jgp.herokuapp.com/ | https://git.heroku.com/app-jgp.git

C:\Users\juang>
```

Si el nombre de la aplicación ya está tomado, no avisa para que lo cambiemos.

Inicializamos un repositorio git sobre un directorio nuevo o existente (si el proyecto nos lo hemos descargado desde algún repositorio git usando el comando **git clone**, es necesario que eliminemos el directorio **.git** que contiene, antes de ejecutar **git init**. En este caso desde dentro del directorio de nuestra aplicación **angular-seed**, ejecutamos:

git init

```
C:\WINDOWS\system32\cmd.exe

C:\Users\juang\Desktop\angular-seed>git init
Initialized empty Git repository in C:/Users/juang/Desktop/angular-seed/.git/

C:\Users\juang\Desktop\angular-seed>_
```

Configuramos un repositorio remoto sobre Heroku: **heroku git:remote -a app-jgp**

```
C:\WINDOWS\system32\cmd.exe

C:\Users\juang\Desktop\angular-seed>git init
Initialized empty Git repository in C:/Users/juang/Desktop/angular-seed/.git/

C:\Users\juang\Desktop\angular-seed>heroku git:remote -a app-jgp
set git remote heroku to https://git.heroku.com/app-jgp.git

C:\Users\juang\Desktop\angular-seed>
```

Comitear el código al repositorio:

git add .

git commit -am "firts commit"

```
C:\WINDOWS\system32\cmd.exe

C:\Users\juang\Desktop\angular-seed>git init
Initialized empty Git repository in C:/Users/juang/Desktop/angular-seed/.git/

C:\Users\juang\Desktop\angular-seed>heroku git:remote -a app-jgp
set git remote heroku to https://git.heroku.com/app-jgp.git

C:\Users\juang\Desktop\angular-seed>git add .
warning: LF will be replaced by CRLF in .bowerrc.
The file will have its original line endings in your working directory.
```

```
warning: LF will be replaced by CRLF in test/spec/controllers/main.js.  
The file will have its original line endings in your working directory.  
  
C:\Users\juang\Desktop\angular-seed>git commit -am "first commit"  
[master (root-commit) a4d1715] first commit  
412 files changed, 45351 insertions(+)  
create mode 100644 .bowerrc  
create mode 100644 .editorconfig
```

Desplegar realizar commit al repositorio remoto y para desplegar la aplicación desde el repositorio git al servidor Heroku e instalar los paquetes npm:

git push heroku master

Para asegurarnos que al menos una instancia de la aplicación está arrancada:

heroku ps:scale web=1

Ahora, visitamos la aplicación en la url generada por su nombre de aplicación

https://app-jgp.herokuapp.com.

También podemos utilizar un atajo con el comando:

heroku open

Si queremos ver el log:

heroku logs --tail

Subir cambios

El servidor está preparado para servir sólo el contenido del directorio ***client/dist***, esto quiere decir que antes de realizar el commit, debemos crear la versión de distribución con el comando ***npm run start:dist***

y después:

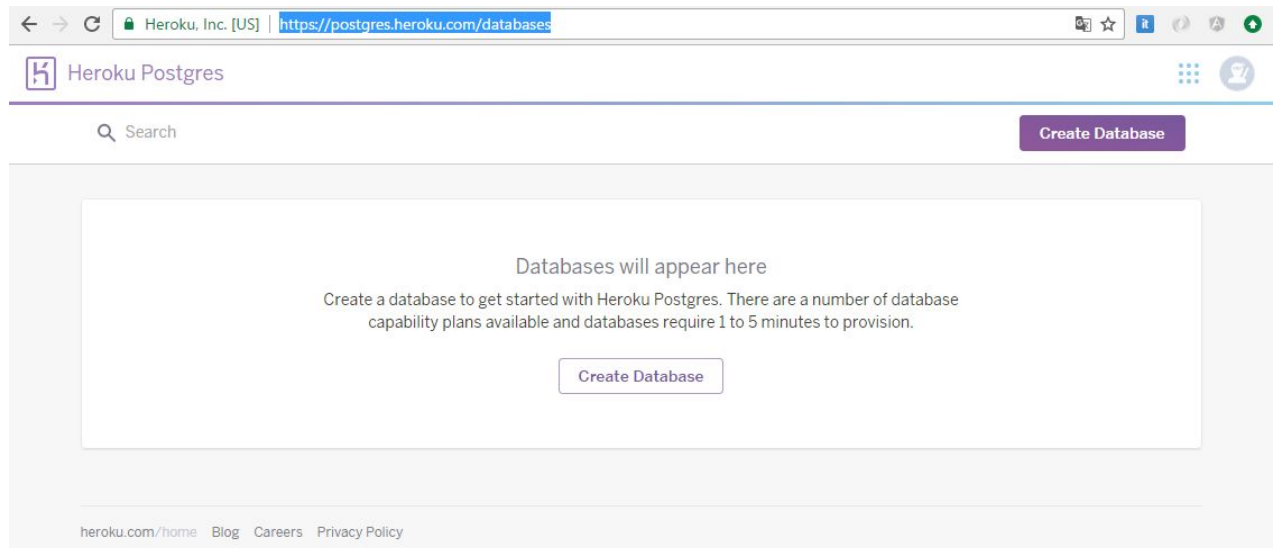
git add .

git commit -am "comment"

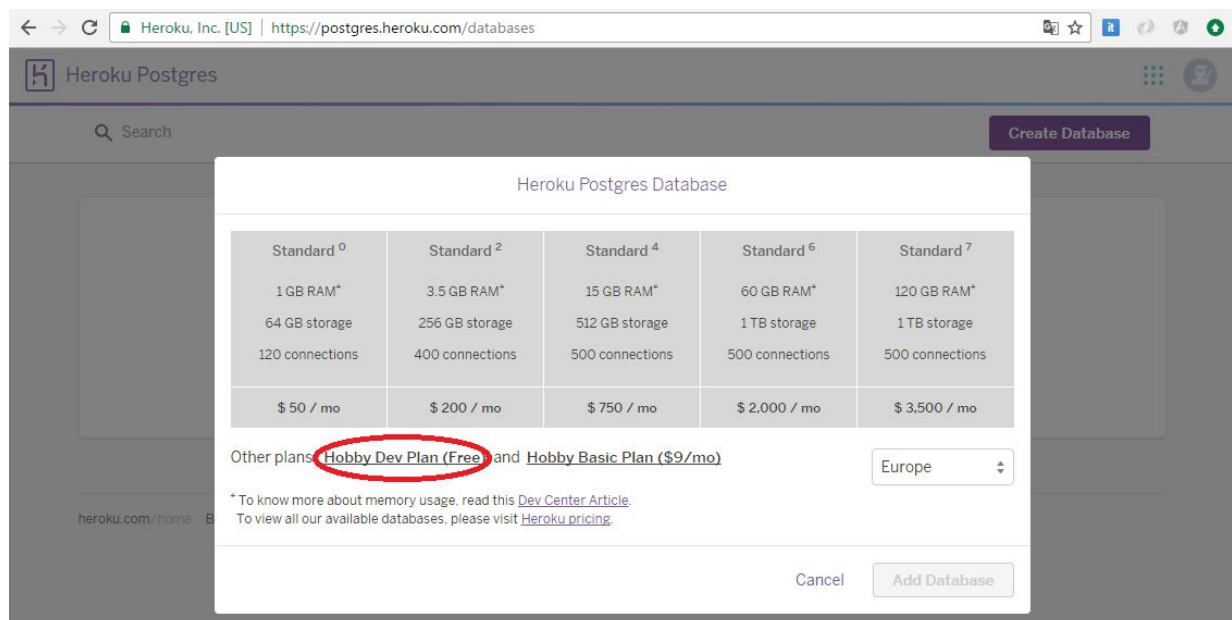
git push heroku master

Creación de la Base de Datos sobre HEROKU

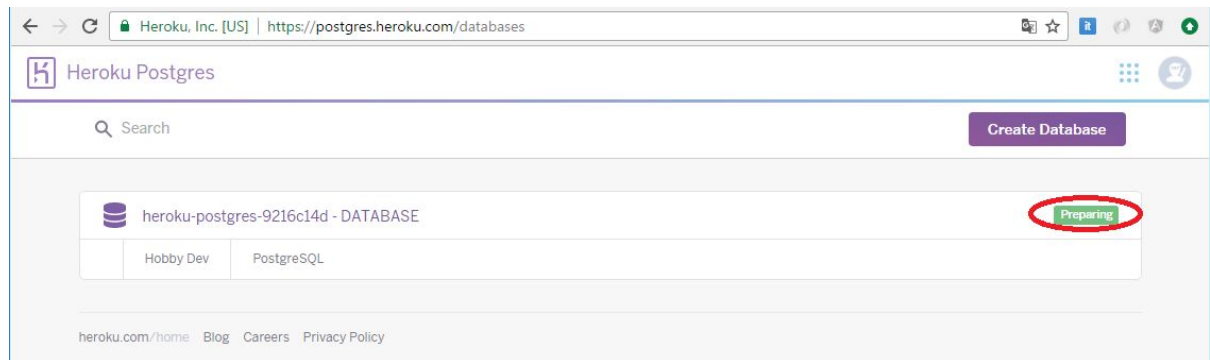
<https://postgres.heroku.com/databases>



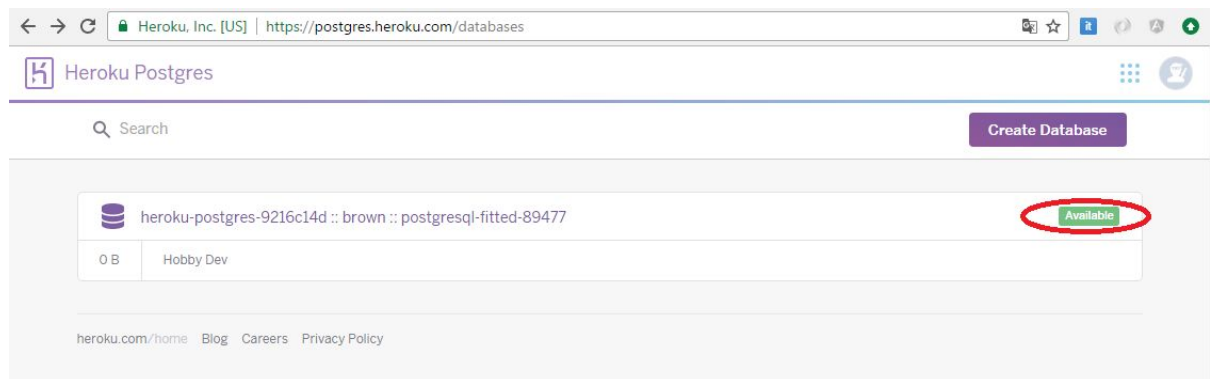
Elegimos **Hobby Dev Plan (Free)** y hacemos click en el botón **Add Database**:



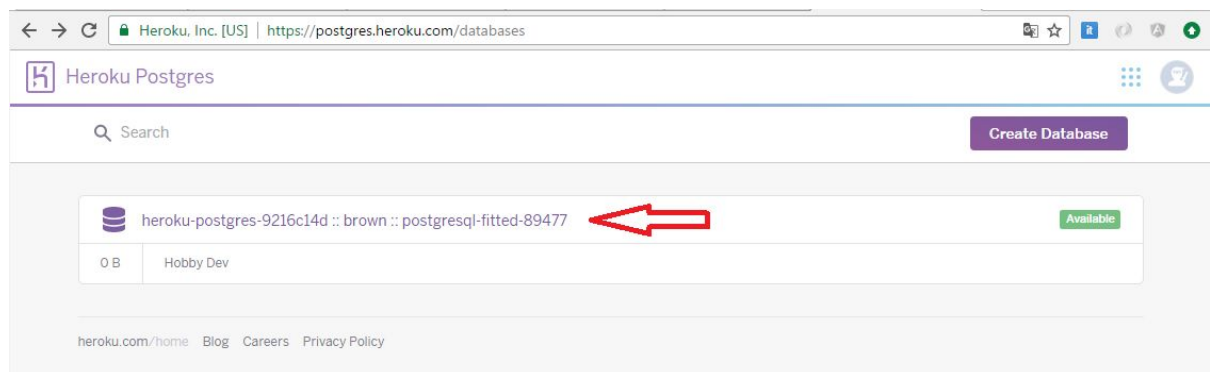
esperamos a que el estado pase de **preparing**:



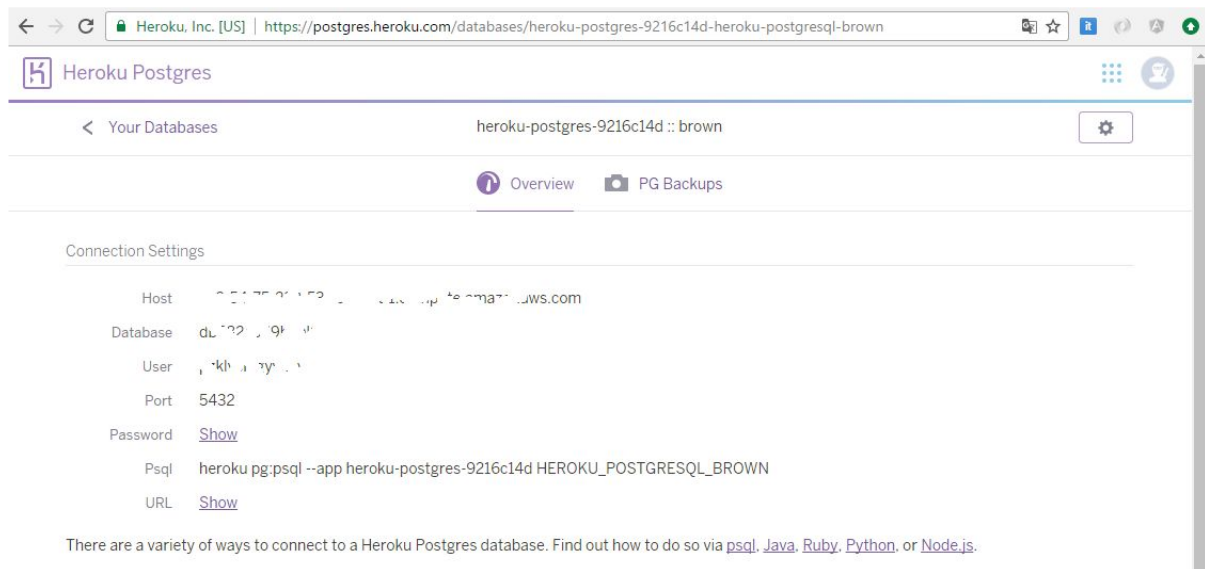
a **available**:



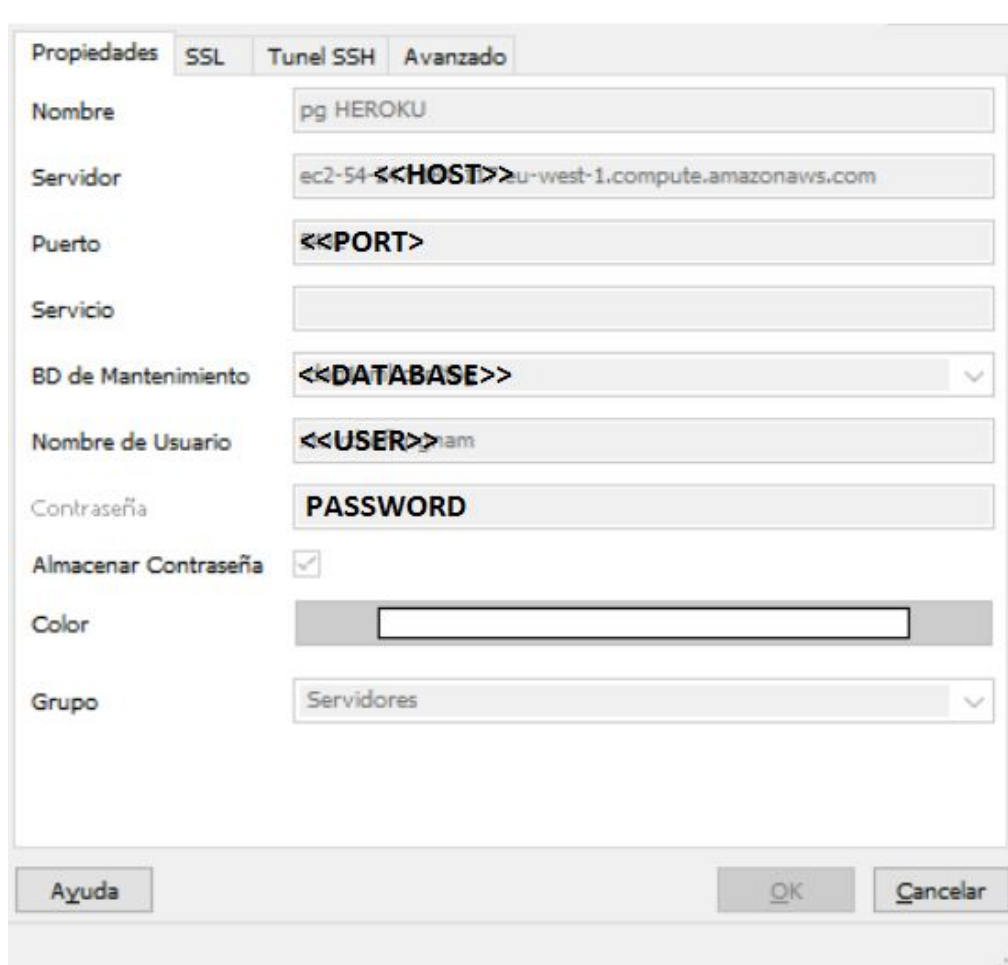
Ahora, haciendo click sobre el enlace:



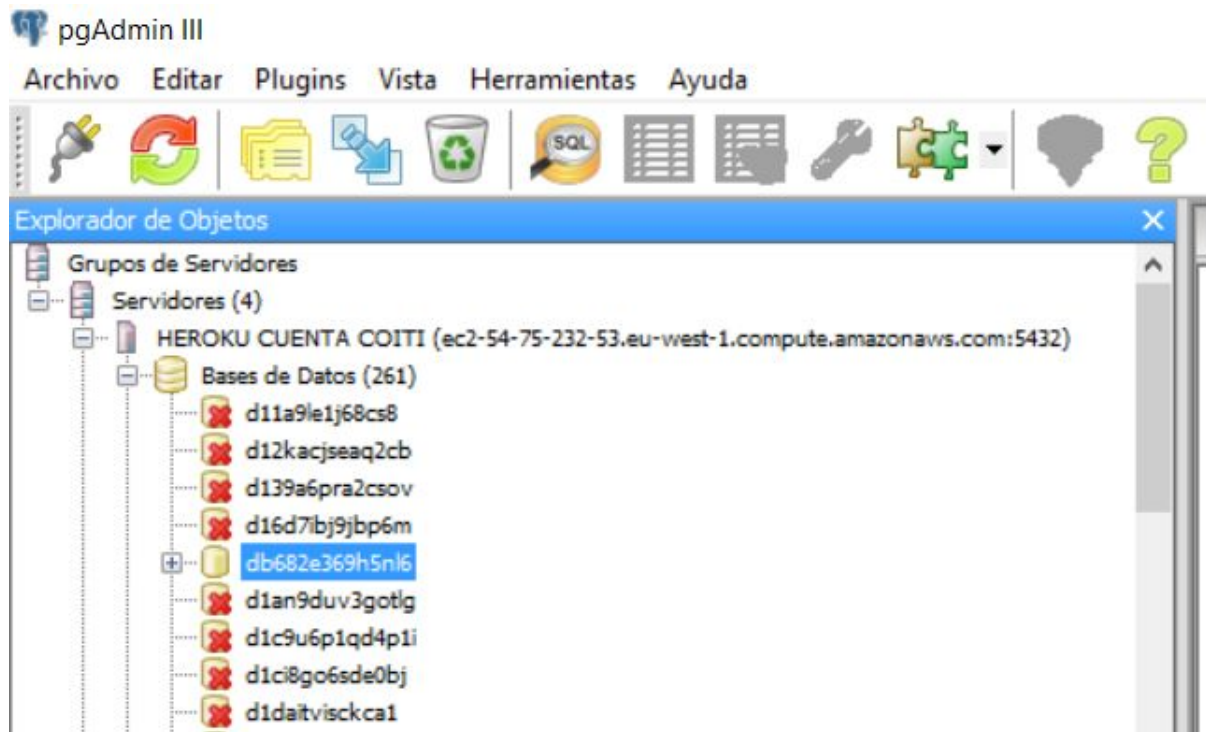
accedemos a los datos de configuración de la conexión:



con estos datos creamos una nueva conexión en **pgAdmin**:



seleccionamos la <<DATABASE>> y ejecutamos el script **esquemaDB.sql** desde la ventana SQL:



Establecimiento de Variables de Configuración en el Servidor Heroku

Desde una ventana de comandos iniciamos sesión sobre heroku (**heroku login**) para crear las variables de configuración (**heroku config:set VAR_NAME=VALUE**) de la conexión a la base de datos, para que la aplicación se conecte a la base de datos adecuada cuando esté desplegada sobre heroku.6.

```
C:\WINDOWS\system32\cmd.exe
Enter your Heroku credentials.
Email: [redacted]
Password (typing will be hidden): [redacted]
Logged in as [redacted]

C:\Users\juang\Desktop\angular-seed>heroku config:set DB_HOST=[redacted]eu-west-1.compute.amazonaws.com
Setting DB_HOST and restarting app-coi... done, v3
DB_HOST: [redacted]eu-west-1.compute.amazonaws.com

C:\Users\juang\Desktop\angular-seed>heroku config:set DB_PORT=5432
Setting DB_PORT and restarting app-coi... done, v4
DB_PORT: 5432

C:\Users\juang\Desktop\angular-seed>heroku config:set DB_DATABASE=[redacted]
Setting DB_DATABASE and restarting app-coi... done, v5
DB_DATABASE: [redacted]

C:\Users\juang\Desktop\angular-seed>heroku config:set DB_USER=[redacted]
Setting DB_USER and restarting app-coi... done, v6
DB_USER: [redacted]

C:\Users\juang\Desktop\angular-seed>heroku config:set DB_PASSWORD=[redacted]
Setting DB_PASSWORD and restarting app-coi... done, v7
DB_PASSWORD: [redacted]
```