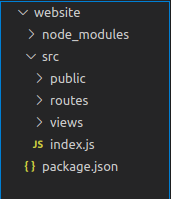
Primer sitio web con express

fuente: <https://www.youtube.com/watch?v=sh-NanMOh1Q&t=267s>

## 1 – Creación de la estructura e inicialización

Un proyecto web con express tiene la siguiente estructura básica de carpetas



Website es la carpeta contenedora del proyecto, la cual contiene las siguientes carpetas y archivos:

* **node\_modules**. Contiene las dependencias del proyecto.
* **Src**. Contiene el código propio.
* **Package.json**. Archivo de metadatos que contiene información del proyecto, como descripción, dependencias, scripts, etc.

La carpeta src tiene a su vez tres subcarpetas:

* **Public**. Contiene los archivos html, css, js, imágenes, etc. que deben ser accedidos desde el navegador.
* **Routes**. Es donde se declara cómo responde el servidor a las peticiones que recibe.
* **Views**. Contiene las vistas que serán renderizadas desde cada ruta.

\* En esta guía solo usamos esas carpetas pero es importante saber que más adelante en proyectos mas complejos y con conexión a base de datos también son necesarias las carpetas **model** y **controller** para establecer los modelos de los documentos a cargar en la bd, y desacoplar la lógica de las rutas.

El package.json se crea de manera rápida abriendo una consola en la carpeta principal del proyecto y ejecutando el siguiente comando.

npm init –yes

Luego instalamos las dependencias necesarias para el proyecto.

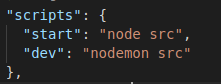
* Express. Framework de nodejs para crear páginas web.
* Pug. Motor de plantillas.
* Morgan. Logger para control de peticiones en el servidor.

npm i express pug morgan –save

Como dependencia de desarrollador instalamos nodemon, que permite refrescar el servidor cada vez que se realiza un cambio.

npm i nodemon -D

Para ejecutarlo de manera mas sencilla agregamos el siguiente script en el package.json.



Ahora podemos ejecutar el proyecto desde consola con el comando

npm run dev

## 2 – Creación del servidor

En el archivo principal index.js creamos el código del servidor.



1. Importamos los módulos necesarios, en este caso son el framework **express**, el módulo **morgan** para la utilización de su middleware y el módulo **path** que permite unir direcciones (ver cómo funciona).
2. Incializamos el módulo de express.
3. Asignamos algunos valores a las configuraciones de express. Esto equivale a declarar algunas constantes de uso general. En este caso indicamos el numero de puerto que se va a usar, el motor de plantillas (pug) y la dirección de la carpeta con las vistas.
4. Declaramos la sección de middlewares (porciones de código que deben ser ejecutadas antes de cada ruta). En este caso solo usamos el módulo morgan.
5. Indicamos la ubicación del archivo con las rutas que gestionan las peticiones del navegador. De momento se encuentran todas en un único archivo, pero en aplicaciones mas grandes es recomendable separarlas al menos por cada entidad que representa una colección en la base de datos.
6. Declaramos la ubicación de la carpeta con archivos estáticos. Estos pueden ser accedidos desde el navegador. En general son imágenes, scripts o archivos css para estilizar la página.

Por ejemplo, si en el navegador ponemos [**http://localhost:3000/css/estilos.css**](http://localhost:3000/css/estilos.css)muestra el archivo estilos.css, ya que es un archivo público del servidor.

1. La última sección es donde se asigna el puerto y se inicializa el servidor.

## 3 - Router

De la misma manera en que se relacionan la ruta principal con un archivo en la carpeta views se debe crear una ruta específica para cada petición del servidor. Tener todas las rutas en el archivo principal es un problema en una página muy grande, por lo tanto estas se deben implementar en un archivo aparte.



1. Primero se crea en src una nueva carpeta llamada **routes**, con su propio **index.js**.
2. En este archivo se vuelve a importar e inicializar express, pero ya no con el nombre app, sino llamado **router**. Con esta constante se van a ir creando todas las rutas y al finalizar es exportada para ser requerida desde la aplicación.

\*A diferencia del video, para que funcione tuve que configurar dentro de este archivo el path de las vistas.



1. Especificamos como se va a responder a cada ruta
2. exportamos el objeto router para ser utilizado desde el archivo principal de la aplicación.



### Rutas

Las rutas tienen una sintaxis específica. Primero se llama a algún método que resuelva una petición http, de momento solo usamos **get**, pero en general según la acción que se realice los mas comunes son:

* **Get**. Cuando se solicita información de la página.
* **Post**. Al agregar datos en una bd.
* **Put**. Al modificar o actualizar algún dato.
* **Delete**. Al eliminar algo.

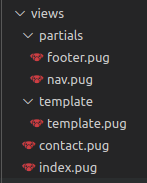
La función **get** recibe dos parámetros, el primero es la **url** desde donde se hizo la peticion, y el segundo una función de **callback** donde se implementa lo que debe hacer el servidor al recibir la petición. Esta función tiene a su vez tres parámetros:

* Request (**req**). Contiene información enviada desde el cliente.
* Response (**res**). Almacena información que debe ser devuelta al navegador. Acepta metodos como send (respuesta en formato texto, json o xml), sendFile(respuesta en formato html), render, etc.
* **next**. Permite usar la ruta a modo de middleware, ejecutando una porción de código pero sin devolver una respuesta al navegador y pasando el control a otra ruta.

La función usada en este caso para la respuesta es **render.** Esta recibe el nombre de la platilla que se debe renderizar y puede recibir un objeto json con información a utilizar en la misma.

## 4 – Vistas

Para las vistas tenemos la siguiente estructura de carpetas

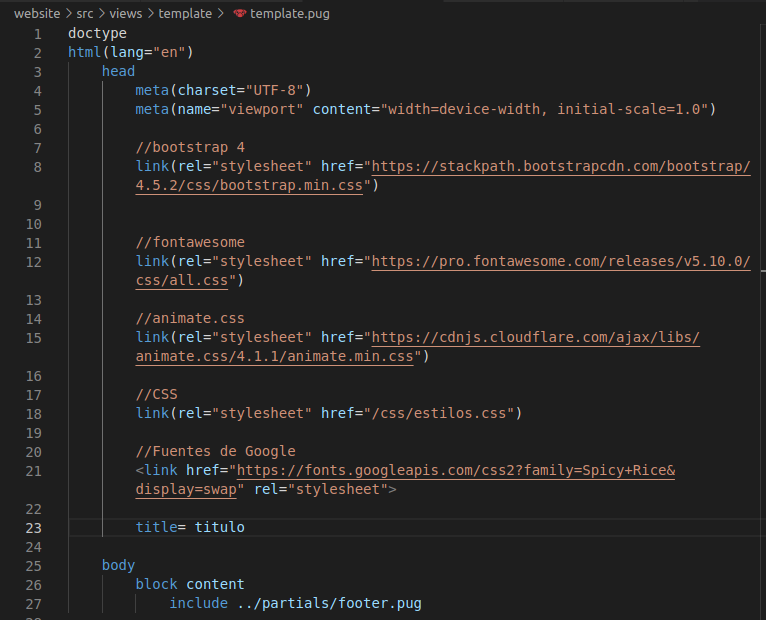
Partials contiene bloques de código de tipo **include**, mientras que template tiene bloques de tipo **extends**.

En la carpeta principal se ubican las url de las páginas que se renderizan desde las rutas.

Nuestra página tendrá las siguientes url:

* **‘/’**, (home) representada por la plantilla index.
* **‘/#about’**. No tiene una plantilla propia porque es un bloque dentro del home.(div con id #about)
* **‘/contact’**

El primer archivo que creamos es dentro de la carpeta template, **template.pug**

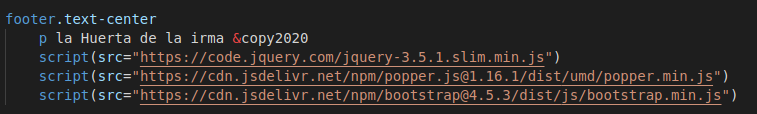


En este documento se implementa el head de todas las plantillas que vamos a utilizar, por lo tanto es donde se incluyen las referencias a los archivos css o los cdn respectivos:

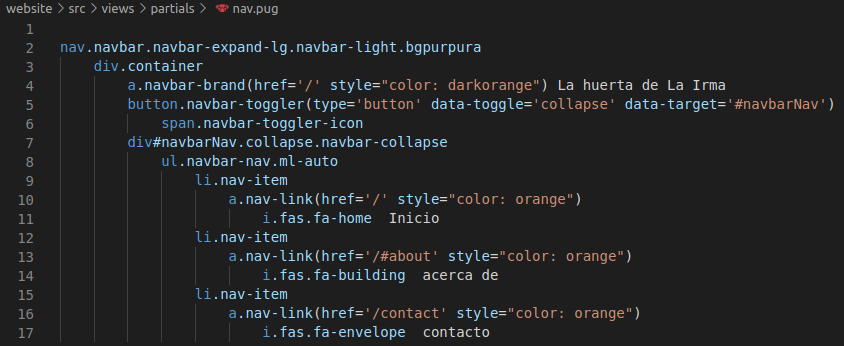
* bootstrap 4. framwork de css.
* Font awesome. Librería de íconos.
* Animate css. Animación de objetos html
* CSS propio.
* Fuentes de google. Fuentes tipográficas.

Luego en partials creamos los archivos nav.pug y footer.pug que contienen bloques de tipo include.

**Footer.pug**

Contiene el mensaje del footer y los scripts js de bootstrap.

**Nav.pug**

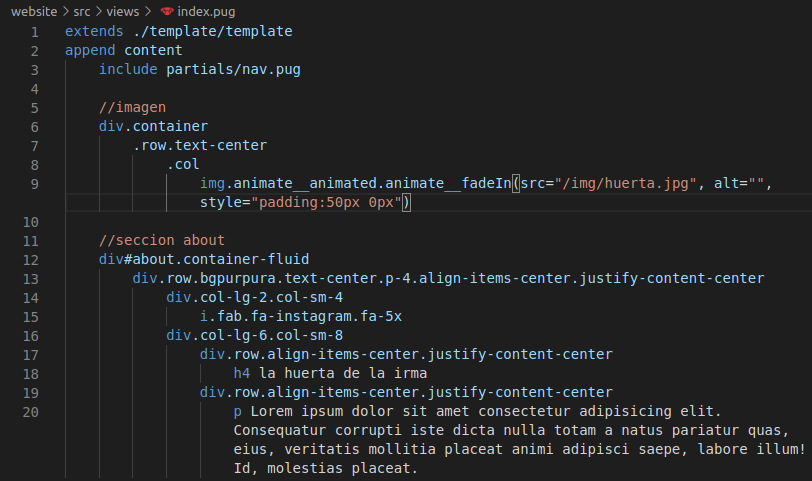
\*ver ‘11 – Navegación’ en el apunte ‘4 – bootstrap/pug’

Tambien puede copiarse una nevagación estandar de

<https://getbootstrap.com/docs/5.0/components/navbar/>

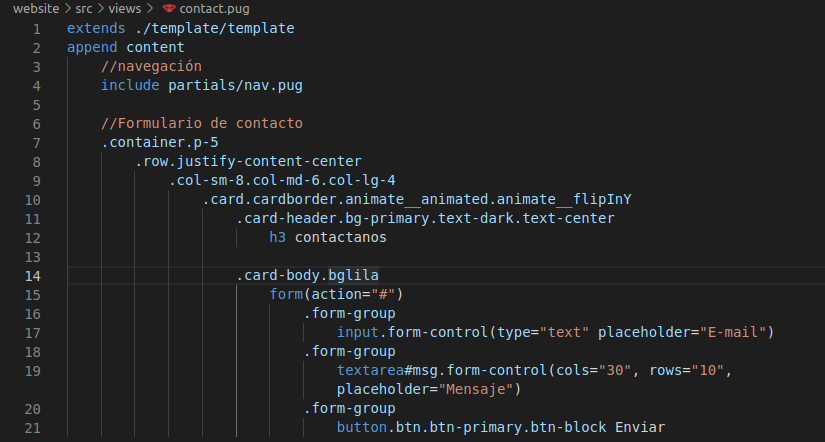
Por último creamos las vistas principales, estas extienden de template.pug

**index.pug**



Contiene dos secciones, una central con un logo de la aplicación, y la sección about con información acerca de la misma. Estudiar el código para ver como se aplican los estilos de bootstrap, íconos, animaciones, etc.

**contact.pug**



## 5 – Archivos estáticos

En la carpeta **src** del proyecto creamos un nuevo directorio llamado **public**, donde se van a almacenar todos los archivos de los que debe disponer el navegador para estilizar la página, es decir, archivos css, imágenes, fuentes, etc. Para que el navegador pueda acceder a estos archivos hay que indicarle desde el servidor cuál es la carpeta que los contiene mediante el método de express **static**.



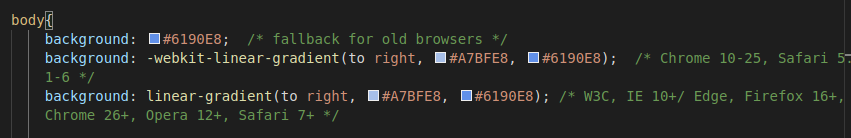
Al saber el navegador donde está la carpeta public, alcanza con indicar en los documentos html la ubicación de los estilos dentro de la misma. Entonces agregamos el siguiente código en head.pug



Dentro de la capeta **public** creamos la carpeta **css**, y en esta un archivo **estilos.css** con el siguiente contenido

**UIGradients**

La pagina <https://uigradients.com/>disponde de fondos degradados para las paginas web. Solo hace falta entrar, elegir el fondo y copiar el css

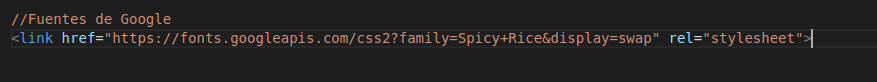


**Fuentes**

para utilizar fuentes de terceros buscamos en internet la fuente requerida, por ejemplo spicy rice de google.

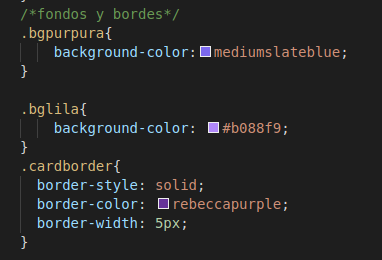
<https://fonts.google.com/specimen/Spicy+Rice?sidebar.open=true&selection.family=Spicy+Rice>

En la página seleccionamos la funte y de la ventana emergente copiamos el cdn.

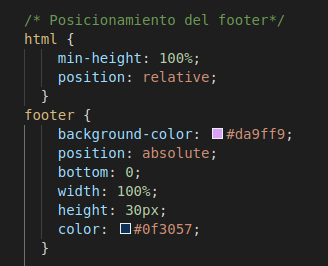
Agregamos este cdn en el archivo head.pug. Esto permite acceso a la fuente pero no la utiliza en la página. Para usar la fuente la declaramos en el archivo estilos.css de la carpeta public/css



El resto de estilos necesarios para la pagina son los siguientes



Algunos copiados de internet (<https://es.stackoverflow.com/questions/10757/posicionar-footer-siempre-pegado-al-pie-de-pagina>)



Y algunos media queries para adaptar el los tamaños

