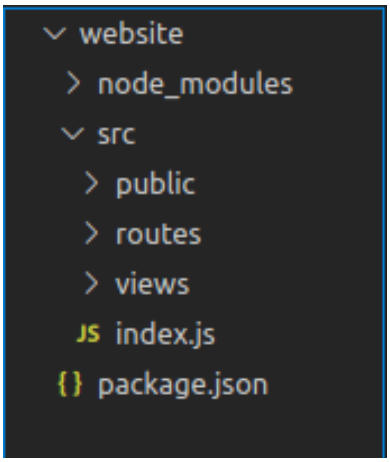
# Primer sitio web con express

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.

```
"scripts": {
  "start": "node src",
  "dev": "nodemon src"
},
```

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.

website > src > JS index.js > ...

```
1   //modulos
2   const express = require('express');
3   const path = require('path');
4   const morgan = require('morgan');
5
6   //inicializacion de express
7   const app = express();
8
9   //configuraciones
10  app.set('port', 3000);
11  app.set('view engine', 'pug');
12  app.set('views', path.join(__dirname,'views'));
13
14  //Middlewares
15  app.use(morgan('dev'));
16
17  //Rutas
18  app.use(require('./routes/index'));
19
20  //Archivos estáticos
21  app.use(express.static(path.join(__dirname,'public')));
22
23
24  //Servidor escuchando
25  app.listen(app.get('port'), ()=>{
26      console.log('server on port ', app.get('port'));
27  });
```

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** muestra el archivo estilos.css, ya que es un archivo público del servidor.
7. 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.

website > src > routes > JS index.js > ...

```
1   //modulos
2   const express = require('express');
3   const path = require('path');
4
5   //inicializacion
6   const router = express();
7
8   //config
9   router.set('views', path.join(__dirname, '/../views'));
10
11  //rutas
12  router.get('/', (req, res)=>{
13      res.render('index', {titulo: 'la huerta de la Irma'});
14  });
15
16  router.get('/contact', (req, res)=>{
17      res.render('contact', {titulo: 'contact', mensaje:'Contactanos'});
18  });
19
20  module.exports = router
```

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.

```
//config
router.set('views', path.join(__dirname, '/../views'));
```

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

```
module.exports = router
```

```
app.use(require('./routes/index'));
```

## 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.

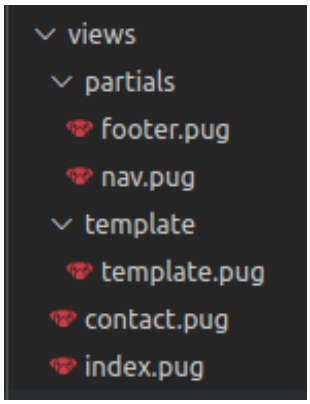
La función **get** recibe dos parámetros, el primero es la **url** desde donde se hizo la peticion, y el segundo una función de **callback** donde se implementa lo que debe hacer el servidor al recibir la petición. Esta función tiene a su vez tres parámetros:

- 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**

website > src > views > template > template.pug

```
doctype
html(lang="en")
    head
        meta(charset="UTF-8")
        meta(name="viewport" content="width=device-width, initial-scale=1.0")

        //bootstrap 4
        link(rel="stylesheet" href="https://stackpath.bootstrapcdn.com/bootstrap/4.5.2/css/bootstrap.min.css")


        //fontawesome
        link(rel="stylesheet" href="https://pro.fontawesome.com/releases/v5.10.0/css/all.css")

        //animate.css
        link(rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/animate.css/4.1.1/animate.min.css")

        //CSS
        link(rel="stylesheet" href="/css/estilos.css")

        //Fuentes de Google
        <link href="https://fonts.googleapis.com/css2?family=Spicy+Rice&display=swap" rel="stylesheet">

        title= titulo

    body
        block content
            include ../partials/footer.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**

```
footer.text-center
    p la Huerta de la irma &copy2020
    script(src="https://code.jquery.com/jquery-3.5.1.slim.min.js")
    script(src="https://cdn.jsdelivr.net/npm/popper.js@1.16.1/dist/umd/popper.min.js")
    script(src="https://cdn.jsdelivr.net/npm/bootstrap@4.5.3/dist/js/bootstrap.min.js")
```

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

**Nav.pug**

website > src > views > partials > nav.pug

```
1
2   nav.navbar.navbar-expand-lg.navbar-light.bgpurpura
3       div.container
4           a.navbar-brand(href='/' style="color: darkorange") La huerta de La Irma
5           button.navbar-toggler(type='button' data-toggle='collapse' data-target='#navbarNav')
6               span.navbar-toggler-icon
7           div#navbarNav.collapse.navbar-collapse
8               ul.navbar-nav.ml-auto
9                   li.nav-item
10                      a.nav-link(href='/' style="color: orange")
11                          i.fas.fa-home  Inicio
12                  li.nav-item
13                      a.nav-link(href='/#about' style="color: orange")
14                          i.fas.fa-building  acerca de
15                  li.nav-item
16                      a.nav-link(href='/contact' style="color: orange")
17                          i.fas.fa-envelope  contacto
```

*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**

website > src > views > index.pug

```
extends ./template/template
append content
    include partials/nav.pug

    //imagen
    div.container
        .row.text-center
            .col
                img.animate__animated.animate__fadeIn(src="/img/huerta.jpg", alt="",
                style="padding:50px 0px")

    //seccion about
    div#about.container-fluid
        div.row.bgpurpura.text-center.p-4.align-items-center.justify-content-center
            div.col-lg-2.col-sm-4
                i.fab.fa-instagram.fa-5x
            div.col-lg-6.col-sm-8
                div.row.align-items-center.justify-content-center
                    h4 la huerta de la irma
                div.row.align-items-center.justify-content-center
                    p Lorem ipsum dolor sit amet consectetur adipisicing elit.
                    Consequatur corrupti iste dicta nulla totam a natus pariatur quas,
                    eius, veritatis mollitia placeat animi adipisci saepe, labore illum!
                    Id, molestias placeat.
```

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**

website > src > views > contact.pug

```
extends ./template/template
append content
    //navegación
    include partials/nav.pug

    //Formulario de contacto
    .container.p-5
        .row.justify-content-center
            .col-sm-8.col-md-6.col-lg-4
                .card.cardborder.animate__animated.animate__flipInY
                    .card-header.bg-primary.text-dark.text-center
                        h3 contactanos

                    .card-body.bglila
                        form(action="#")
                            .form-group
                                input.form-control(type="text" placeholder="E-mail")
                            .form-group
                                textarea#msg.form-control(cols="30", rows="10",
                                placeholder="Mensaje")
                            .form-group
                                button.btn.btn-primary.btn-block Enviar
```

# 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**.

```
//Archivos estáticos
app.use(express.static(path.join(__dirname,'public')));
```

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

```
//CSS
link(rel="stylesheet" href="/css/estilos.css")
```

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

```
body{
    background: #6190E8;  /* fallback for old browsers */
    background: -webkit-linear-gradient(to right, #A7BFE8, #6190E8);  /* Chrome 10-25, Safari 5
    1-6 */
    background: linear-gradient(to right, #A7BFE8, #6190E8); /* W3C, IE 10+/ Edge, Firefox 16+,
    Chrome 26+, Opera 12+, Safari 7+ */
```

### 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.

```
//Fuentes de Google
<link href="https://fonts.googleapis.com/css2?family=Spicy+Rice&display=swap" rel="stylesheet">
```

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

```
font-family: 'Spicy Rice';
font-size: 100%;
```

El resto de estilos necesarios para la pagina son los siguientes

```css
/*fondos y bordes*/
.bgpurpura{
    background-color: mediumslateblue;
}

.bglila{
    background-color: #b088f9;
}
.cardborder{
  border-style: solid;
  border-color: rebeccapurple;
  border-width: 5px;
}
```

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

```css
/* Posicionamiento del footer*/
html {
    min-height: 100%;
    position: relative;
  }
footer {
    background-color: #da9ff9;
    position: absolute;
    bottom: 0;
    width: 100%;
    height: 30px;
    color: #0f3057;
  }
```

Y algunos media queries para adaptar el los tamaños

```css
/*Tamaños adaptables*/
h3 {font-size: 200%;}

@media screen and (max-width: 992px) {
h3 {
    font-size: 175%;
  }
img{
  width: 75%
}
}

@media screen and (max-width: 540px) {
h3 {
    font-size: 150%;
  }
img{
  width: 50%
}
}
```