

Documentación Prueba BSALE

Tienda
BSALE

ES


Buscar un producto

Buscar

BSALE TEST

Filtrar por categorías


Filtrar Productos



Octubre de Ofertas Especiales


Convierte esos momentos especiales en algo inolvidable

Bebidas




ENERGETICA MIR BIG

\$ 1490




ENERGETICA RED BULL

\$ 1490




ENERGETICA SCORE

\$ 1290




PISCO ALTO DEL CARMEN 35°

\$ 7990




PISCO ALTO DEL CARMEN 40°

\$ 5990




PISCO ARTESANOS 35°

\$ 3990




PISCO BATZA 40°

\$ 4990




PISCO CAMPANARIO 35°

\$ 2990




PISCO CAMPANARIO 40°

\$ 3990




PISCO ESPIRITU DEL ELQUI 40°

\$ 5990




PISCO ESPIRITU DEL ELQUI 40°

\$ 4990




PISCO BORCON QUEMADO 35°

\$ 4990




PISCO BORCON QUEMADO 40°

\$ 7990




PISCO BORCON QUEMADO 40°

\$ 8990



PISCO MISTRAL 35°

\$ 4990



PISCO MISTRAL 40°

\$ 4990

Bsale Test

Síguenos en nuestras redes sociales como @Bsale

Bebidas

Pisco
Ron
Cerveza
Sanco

Contactanos

Chile, Mexico, Colombia
contacto@bsale.com
+54 143243424

Visa

Manual Técnico Orlin Hernández 2022

Contenido

Documentación Prueba BSALE	1
Prueba BSALE.....	3
Requerimientos.....	4
Paradigma de Programación Seleccionado y sus Beneficios.....	6
Lenguaje de Programación Seleccionado	8
Diseño del Sistema.....	9
Configuración del Servidor	9
Rutas	12
Deployment	13
Requisitos	13
Enlace al Deploy.....	13
Enlace a Github	13

Prueba BSALE

- Construir una tienda online.
- Utilizar la base de datos que se encuentra en la hoja 2 de este desafío.
- Desplegar productos agrupados por la categoría a la que pertenecen.
- Generar por separado backend (API REST) y frontend (aplicación que la consuma)
- Agregar un buscador, el cual tiene que estar implementado a nivel de servidor, mediante una Api Rest cuyo lenguaje y framework puede ser de libre elección. Es decir, los datos de productos deben llegar filtrados al cliente.
- Desarrollar la aplicación de cliente con vanilla javascript/Vanillajs.
- ¿Puedo usar react y similares? La respuesta es NO, sólo usar vanilla javascript/ Vanillajs. Si lo desarrollas con react o similares, quedará el ejercicio sin efecto.
- ¿Puedo usar librerías o componentes específicos?, Si, tales como; bootstrap, material, JQuery, entre otros.
- Disponibilizar la aplicación en un hosting como a modo de ejemplo, puede ser; Heroku, Netlity, Aws u otro.
- Disponibilizar el código en Github.

Requerimientos

Prueba Técnica Bsale Orlin Hernandez

Arquitectura

Para el desarrollo de la aplicación opté por el patrón MVC; Patrón de diseño de la capa de presentación, pues define la forma en que se organizan los componentes de presentación en sistemas distribuidos.

Requerimientos

Para el desarrollo y ejecución del proyecto deberás instalar las dependencias con el comando

Node

- Instalar Node en Windows

Visita oficial Node.js website y descarga el instalador.

- Instalación en otros sistemas operativos

Puedes encontrar información en el sitio oficial(<https://nodejs.org/>) y Sitio Oficial de NPM.

Dependencias

Express: Framework de preferencia para trabajar las APIs en Node js

Mysql2: Compatible con la mayoría de apis, aparte que ofrece características adicionales.

EJS: EJS nos permite generar aplicaciones rápidas cuando no necesitamos algo demasiado complejo

Dotenv: Para facilitar el acceso a nuestras variables de entorno.

- Dependencias de seguridad extra para nuestro servidor

RateLimiter: Con esta dependencia limitamos las peticiones a nuestra API

Helmet: Nos brinda una póliza de seguridad para mitigar ataques cross-site entre otras cosas.

Cors: Provee características de seguridad al navegador para mitigar peticiones cross-origin desde HTTP.

Xss: Middleware para depurar el input de los usuarios provenientes del POST body, Get y parámetros URL.

Deploy

Para el deployment se utilizó la herramienta de Heroku mediante el repositorio de Github.

Paradigma de Programación Seleccionado y sus Beneficios

Para el desarrollo de la prueba se optó por utilizar el patrón MVC

El uso del patrón MVC ofrece múltiples ventajas sobre otras maneras de desarrollar aplicaciones con interfaz de usuario, y en especial para la Web. Sin entrar en detalles aquí, porque la extensión del artículo ya es grande, comentaré a continuación algunas de ellas:

- La clara separación de responsabilidades impuesta por el uso del patrón MVC hace que los componentes de nuestras aplicaciones tengan sus misiones bien definidas. Por lo tanto, nuestros sistemas serán más limpios, simples, más fácilmente mantenibles y, a la postre, más robustos.
- Mayor velocidad de desarrollo en equipo, que es consecuencia de lo anterior, ya que, al estar separado en tres partes tan diferenciadas, diferentes programadores pueden ocuparse de cada parte en paralelo. Esto la hace ideal para el desarrollo de aplicaciones grandes.
- Múltiples vistas a partir del mismo modelo, pudiendo reaprovechar mucho mejor los desarrollos y asegurando consistencia entre ellas.
- Facilidad para realización de pruebas unitarias.

Sin embargo, es justo tomar en cuenta sus desventajas:

- Hay que ceñirse a las convenciones y al patrón. El uso de las convenciones impuestas por el framework y la estructura propuesta por el patrón arquitectural MVC nos obliga a ceñirnos a las mismas, lo que puede resultar a veces algo tedioso si lo comparamos con la forma habitual de trabajar con otros frameworks que dan más libertad al desarrollador. La división impuesta por el patrón MVC obliga a mantener un mayor número de archivos, incluso para tareas simples.
- Curva de aprendizaje. Dependiendo del punto de partida, el salto a MVC puede resultar un cambio radical y su adopción requerirá cierto esfuerzo. Además, utilizarlo implica conocer bien las tecnologías subyacentes con las que se implemente: la plataforma de programación utilizada, además de la tecnología utilizada para la interfaz de usuario (HTML, CSS, JavaScript en el caso de la Web).

De todos modos, el uso del patrón MVC y sus variantes está claro que ha triunfado en todo tipo de desarrollos (Web, móvil, de escritorio...) y en todo tipo de plataformas (rara es la plataforma actual que no lo implementa para uno o varios tipos de desarrollos). En la actualidad no te puedes permitir el lujo de no conocerlo.

Lenguaje de Programación Seleccionado

Se utilizó el lenguaje de programación JavaScript y a su vez Node.js®, Node.js, es un entorno en tiempo de ejecución multiplataforma para la capa del servidor (en el lado del servidor) basado en JavaScript.

Node.js es un entorno controlado por eventos diseñado para crear aplicaciones escalables, permitiéndote establecer y gestionar múltiples conexiones al mismo tiempo. Gracias a esta característica, no tienes que preocuparte con el bloqueo de procesos, pues no hay bloqueos.

Diseño del Sistema

Configuración del Servidor

```
require('dotenv').config();
const express = require("express");
const connect = require ('./db/connect')
const ProductRoutes = require("./routes/product");
const path = require("path");
const app = express();
//Seguridad Extra
const helmet = require('helmet')

const xss = require('xss-clean')
const rateLimiter = require('express-rate-limit')

//Puerto de Servidor
const PORT = process.env.PORT || 4000;
```

- Para comenzar hacemos el llamado a nuestros paquetes que harán posible el levantamiento de nuestro servidor. En el siguiente orden:
- Hacemos el llamado a nuestras variables de entorno de desarrollo ubicadas en el /Src bajo el nombre “. dotenv” .
- Inicializamos el framework express.
- Conectamos a nuestra función encargada de hacer la conexión a nuestra base de datos.
- Llamamos a nuestras rutas que nos permitirán visualizar y hacer cambios en nuestro navegador.
- Declaramos Path haciendo el llamado a este paquete de node para facilitar la ruta interna de nuestros archivos.
- Declaramos nuestra app igualándolo a express.
- Seguridad Extra.
Llamamos el paquete Helmet, Xss y RateLimiter para cuidar la salud de nuestra API limitando las peticiones por IP.
- Extraemos nuestro Puerto de servicio que en este caso será el 4000.

```

//Middlewares
app.use(express.json());
app.use(express.urlencoded({ extended: false }));

//Rutas a peticiones de productos
app.use("/", ProductRoutes);

app.set("view engine", "ejs"); // motor de plantilla EJS
app.set("views", path.join(__dirname, "../frontend/views"));

//Archivos Estaticos
app.use(express.static(path.join(__dirname, "assets")));

```

- Hacemos uso de express.json para poder formatear los archivos recibidos a Json.
- Urleconded nos permite leer formatos Json en el navegador.
- Declaramos la ruta base para todas nuestras rutas extraídas de nuestro directorio rutas.
- Declaramos el motor de visualización para el formato EJS.
- Indicamos donde debe buscar para desplegar nuestras vistas.
- Hacemos uso de los Assets, donde alojaríamos imágenes de ser necesario y nuestros archivos Css.

```

// Seguridad y control de trafico
app.set('trust proxy', 1)
app.use(rateLimiter({
  windowMs: 25 * 60 *1000, //15 minutos
  max: 100, // limite de 100 peticiones por ventana
}))
app.use(helmet())

app.use(xss())

//Inicializamos servidor
app.listen(PORT, async(req, res) => {
  try {
    console.log(`Servidor activo on port: ${PORT}`);
  } catch (error) {
    console.log(error)
  }
});

module.exports = app;

```

- Hacemos un trust proxy para temas de seguridad interna en nuestra API.
- Limitamos a 15 minutos por ventana nuestra aplicación y a 100 peticiones por IP.
- Utilizamos Helmet para contemplar la posibilidad de ataques Cross Site y Xss.
- Inicializamos nuestro servidor en el puerto 400 mediante un try catch para evitar el colapso del sistema en caso de un error y también contemplar la optimización por el complejo de single thread que encontramos en lenguajes como JavaScript.

Rutas

```
const {getProductos, buscarProductos, detalleProducto, productoAsc, productoAlfa,
productoDesc, buscarCategoria} = require("../controllers/product");

router.route('/').get(getProductos)
router.route('/prod').get(buscarProductos)
router.route('/prod/:id').get(detalleProducto)
router.route('/asc').get(productoAsc)
router.route('/desc').get(productoDesc)
router.route('/alfa').get(productoAlfa)
router.route('/:id').get(buscarCategoria)

module.exports = router;
```

Para comenzar hacemos el llamado a nuestras funciones dentro de nuestro directorio Controller las cuales son las encargadas de mediar con los parámetros de petición en el navegador.

En este caso todos los métodos son Get ya que se estableció la conexión con una base de datos previamente creada por el equipo de BSALE.

- “/” = Llamado a nuestro controlador get Products, el cual nos arroja todos los objetos disponibles en nuestra base de datos en formato Json.
- “/prod” = Búsqueda de productos en base al nombre de este mismo.
- “/prod:id” = Detalle de productos de manera individual, basados en el parámetro ID ingresado por el usuario.
- “/asc” = Obtenemos todos los datos de la base formateados en orden ascendente en base al campo precio.
- “/desc” = Obtenemos todos los datos de la base formateados en orden descendiente en base al campo precio.
- “/alfa” = Obtenemos todos los datos de la base formateados en orden alfabético.
- “/id” = Hacemos una consulta en la cual mediante un inner join obtenemos todos los productos pertenecientes a una categoría.

Deployment

Requisitos

Para el deploy del sistema se hizo uso de la plataforma Heroku por motivos de costos y facilidad para hacer deployments mediante el repositorio de GitHub.

El orden y versión de los paquetes instalados en la instancia fue el siguiente.

```
"cors": "^2.8.5",  
  "dotenv": "^16.0.3",  
  "ejs": "^3.1.8",  
  "express": "^4.18.2",  
  "express-rate-limit": "^6.6.0",  
  "helmet": "^6.0.0",  
  "mysql2": "^2.3.3",  
  "rate-limiter": "^0.2.0",  
  "xss-clean": "^0.1.1"
```

```
"start": "node ./backend/index.js"
```

Comando para iniciar el servidor.

Enlace al Deploy

Para previsualizar los cambios en el servidor https deberá ingresar al siguiente enlace:

[Tienda BSALE \(bsaleorlin.herokuapp.com\)](https://bsaleorlin.herokuapp.com)

Enlace a Github

Para ver el código fuente deberá ingresar al repositorio Github.

[ORLINHNDZ/DesafioBSALE \(github.com\)](https://github.com/ORLINHNDZ/DesafioBSALE)