INSTITUTO TECNOLÓGICO DE TIJUANA

Ingeniería en Sistemas Computacionales

DISEÑO Y DESARROLLO DE SISTEMA MULTI-PLATAFORMA PARA GESTIÓN DE VENTAS Y LOGÍSTICA DE MICROEMPRESA.

POR

JOSÉ LUIS MURILLO RÍOS

PARA OBTENER EL TÍTULO DE INGENIERO EN SISTEMAS COMPUTACIONALES

TIJUANA, B.C. AGOSTO 2019

DISEÑO Y DESARROLLO DE SISTEMA MULTI-PLATAFORMA PARA GESTIÓN DE VENTAS Y LOGÍSTICA DE MICROEMPRESA.

POR

JOSÉ LUIS MURILLO RÍOS

AGOSTO 2019

RESUMEN

Cada día mas emprendedores inician su negocio con la ayuda de servcicios de venta en linea; muchas de estas empresas tienden a crecer y requerir de sistemas mas especializados que se ajusten a necesidades particulares.

Se desarrolla un sistema que tiene como objetivo el llevar control de los procesos de venta y logística de una empresa. El propósito del proyecto es integrar los servicios existentes de venta en linea de la compañía y extender su uso, adaptándalos a los requerimientos específicas del cliente, así como diseñar una interfáz gráfica que permita proporcionar un servicio mas rápido y agilice los proceso de elaboracion y entrega de productos.

DEDICATORIA

A mis abuelos, por creer siempre en mi
Gracias

AGRADECIMIENTOS

Quiero dar gracias a Bolt Media por incluirme en su equipo y darme la oportunidad de crecer.

Índice general

Índice de tablas

Índice de figuras

Código Ejemplo

Capítulo 1

Introducción

Muchas pequeñas empresas dependen en gran medida al éxito de sus ventas en linea, por lo que adoptan el uso de tecnologías robustas (como Shopify y WooCommerce) para su administración. su funcion principal es servir como puntos de entrada para los clientes pero dichos servicios no estan diseñdos para mostrar informacion detallada a distintas areas de la empresa.

1.1. Antecedentes y definición del problema

Se solicitó a la empresa Bolt Media Internacional S. de R.L. de C.V. desarollar una plataforma que conectara las transacciones de rosesland.com, una plataforma de ventas en linea desarrolada en Shopify a las operaciones internas de la florería, como lo son las ventas de mostrador, la manofactura y la entrega de los productos.

Actualmente la florería realiza todo este proceso con comandas que los empleados de ventas llenan a mano, para despues enviarlos al area de elaboración de arreglos florales y posteriormente se entregan al encargado de logistica para dar indicaciones de la distribucion de los productos. Este proceso de venta es lento y genera una experiencia poco placentera para los compradores, a su vez el area de manofactura está conformado por artesanos y trabajadores del campo que tienen un nivel de comprensión de lectura bajo y pueden llegar a

cometer errores que retrasen todos los procesos. Por tales motivos este sistema debe ser fácil de usar para usuarios de distintas areas y a su vez el diseño debe de adaptarse tanto a diferentes tamaños de pantallas como a diferentes dispositivos móviles. Se requiere una base de datos que notifique en tiempo real los cambios en la información asi como un servidor que administre los permisos adecuados para su manipulación.

1.2. Motivación para atender el problema

Una de las herramientas mas poderosas para el comercio es el internet, saber aprovecharlo genera importantes beneficios para cualquier negocio, es por ello que la empresa Rosesland debe adoptar un proceso mas automatizado en sus funciones y tomar ventaja de los ordenadores y dispositivos con los que cuenta la empresa.

1.3. Propuesta de solución

Implementar un sistema diseñado principalmente para ser utilizado en pantallas táctiles que muestre las transacciones realizadas durante el día tanto en mostrador como en la pagina web, dando un informe detallado de las operaciones de la compañia.

1.4. Objetivos generales

Automatizar el proceso de ventas y entregas de la floreía Rosesland e integrar los servicios de su tienda en linea rosesland.com mediante una aplicación web progresiva, que sea compatible con los dispositivos existentes de la empresa y que funcione adecuadamente en navegadores de internet modernos.

1.5. Objetivos específicos

1. Desarrollar un servidor que adminstre la autenticación de los empleados y su acceso a la información.

2. Implementar una base de datos que notifique a los usuarios cambios en la información en tiempo real.

3. Diseñar una aplicacion web progesiva que garantice una experiencia de usuario óptima para todas las areas de la empresa

4. Integrar los servicios de la tienda en linea rosesland.com con las operaciones internas de la empresa, uniendo los procesos en una sola plataforma

Capítulo 2

Marco teórico

Un ***stack de aplicaciones*** es una colección de software o tecnologías que se utilizan para crear una aplicación web. Las aplicaciones de una sola página (SPA - Single Page Application) han crecido en popularidad ya que proporcionan una experiencia de usuario más fluida: llamadas de servidor livianas cambian lo que se muestra en la pantalla sin tener que actualizar toda la página. El resultado parece bastante ingenioso en comparación con la antigua forma de volver a cargar la página por completo. Esto provocó un aumento en los *frameworks* de *front-end*, ya que gran parte del trabajo se realizó en el lado del cliente. Aproximadamente al mismo tiempo, aunque completamente sin relación, las bases de datos NoSQL también comenzaron a ganar popularidad. El término *stack* fue popularizado por primera vez por LAMP Stack: Linux, Apache, MySQL y PHP. Linux es el sistema operativo, Apache actúa como el servidor HTTP, MySQL proporciona la base de datos relacional para manejar la información de la aplicación y PHP es el lenguaje de programación en el que se construye la aplicación. MERN es un paquete de software que significa MongoDB, ExpressJS, ReactJS y NodeJS. Juntos, estos programas gratuitos mejoran la simplicidad del proceso de desarrollo web. Las opciones son muchas, pero elegir una puede ser difícil.

2.1. MEAN/MERN

En comparación con LAMP, el paquete de aplicaciones MEAN es bastante nuevo. Una de sus mayores diferencias es que MEAN no depende de un sistema operativo específico. Node.js se encarga de la ejecución del lado del servidor. MEAN Stack se recomienda especialmente para desarrolladores de JavaScript, ya que utiliza JavaScript en todos los niveles, tanto para el código del lado del cliente, así como el código del lado del servidor. Angular, el potente *framework* de *front-end*, utiliza un patrón de diseño Modelo-Vista-Controlador. A medida que React crece en popularidad, ha presentado una opción alternativa para el desarrollo *frontend*, aunque React es simplemente una biblioteca, no un *framework* MVC completo. MongoDB es un juega un papel importante y una opción increíblemente popular en el mundo de la gestión de bases de datos NoSQL. Node.js le permite al programador escribir el back-end de la aplicación en Javascript, Express.js se usa encima de Node para manejar las solicitudes de enrutamiento y proporcionar una API REST, o incluso puede usarse para generar el HTML final para ser utilizado por el *framework* de *front-end*.

2.1.1. Componentes MEAN

- MongoDB (base de datos)
- Express.js (servidor)
- Angular.js (cliente)
- Node.js (entorno del servidor)

Derivados:

- MERN (React.js en lugar de Angular.js)
- MEEN (Ember.js en lugar de Angular.js)

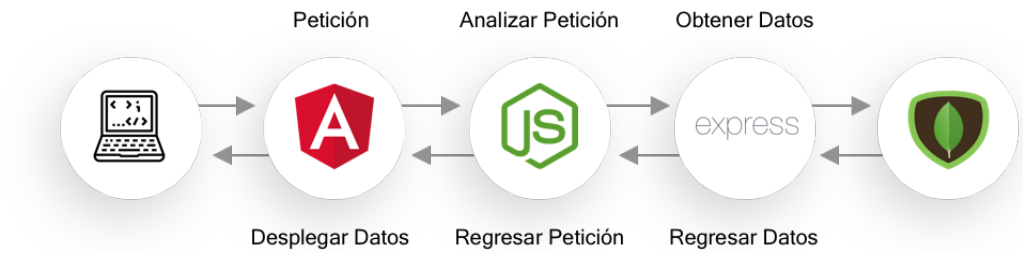

Figura 2.1: Flujo de informacion en MEAN Stack.

2.1.2. Base de datos NOSQL

Las bases de datos NOSQL son una alternativa emergente a las bases de datos relacionales más utilizadas. Como su nombre lo indica, no reemplaza completamente a SQL, sino que lo complementa de tal manera que puedan coexistir

El concepto de NOSQL se desarrolló hace mucho tiempo, pero fue después de la introducción de la base de datos como servicio (DBaaS) que obtuvo un reconocimiento destacado. Debido a la alta escalabilidad proporcionada por NOSQL, fue visto como un importante competidor del modelo de base de datos relacional. A diferencia de RDBMS, las bases de datos NOSQL están diseñadas para escalar fácilmente a medida que crecen. La mayoría de los sistemas NOSQL han eliminado el soporte multiplataforma y algunas características adicionales innecesarias de RDBMS, haciéndolos mucho más livianos y eficientes que sus contrapartes RDMS.

2.1.3. Orientado a documentos

El concepto principal de una base de datos orientada a documentos es que el documento contiene grandes cantidades de datos que pueden estar disponibles de manera útil. Se puede acceder a estos documentos como un directorio regular en el que puede tener diferentes colecciones y cada colección tiene documentos que contienen la información deseada. Además, cada colección puede tener colecciones internas. Puede tener un árbol completo de documentos, sin

embargo, esta práctica no se recomienda y debe evitar tener más de tres niveles de anidamiento.

Las bases de datos NoSQL no son necesariamente bases de datos relacionales. Los datos no son representados en términos de filas y columnas de tablas. En MongoDB, los datos se visualizan como objetos o documentos. Esto ayuda a un programador a evitar una capa de traducción, por lo que no es necesario convertir o asignar los objetos con los que trata el código en tablas relacionales. Dichas traducciones se denominan capas de Mapeo Relacional de Objetos (ORM).

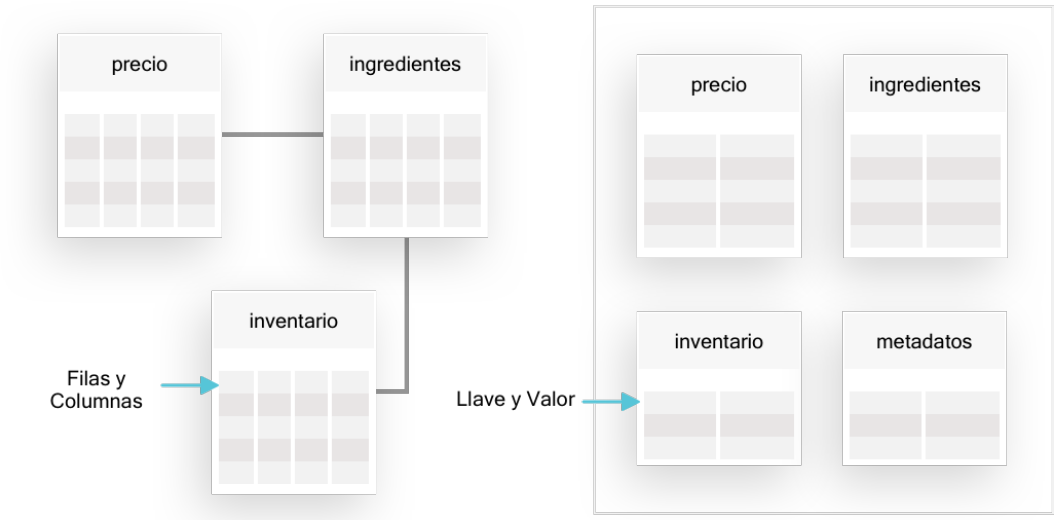

Figura 2.2: Diagrama que representa las diferencias clave entre la base de datos SQL y las bases de datos NoSQL.

2.1.4. Angular/React

Angular o React, proporcionan la interfaz de usuario reactiva de su aplicación. Utilizan componentes, son reactivos porque el usuario recibe cambios inmediatos cuando interactúa con la aplicación y, por lo general, se ejecutan dentro del navegador de un usuario (aunque ambos son isomórficos, capaces de ejecutarse en un servidor).

2.1.5. Beneficios

Usar JavaScript como el lenguaje de programación principal es una gran ventaja. Todo se puede configurar rápidamente y hacer en JavaScript, lo que hace que sea mucho más fácil encontrar desarrolladores, y los desarrolladores de LAMP generalmente también conocen JavaScript. Otra gran ventaja es la capacidad de crear fácilmente aplicaciones móviles o de escritorio, por ejemplo con Ionic. El código y los componentes se pueden reutilizar o agregar fácilmente.

2.1.6. Desventajas

Muchas librerías y *frameworks* son bastante nuevos, y las nuevas versiones se lanzan rápidamente, por lo que mantener una aplicación puede ser una molestia. Dado que muchas tecnologías desaparecen después de unos años, la sostenibilidad puede convertirse en un problema. También es más difícil mantener una base de código limpia y seguir las mejores prácticas a medida que su aplicación crece. Además, debe confiar en el cliente y las tecnologías disponibles del cliente.

2.2. FERN

Firebase es una plataforma propiedad de Google que tiene como objetivo proporcionar un enfoque holístico para un rápido desarrollo web y móvil. En resumen, le permite centrarse en las partes frontales de la aplicación. Se completa con una base de datos visual sin tablas (NoSQL), alojamiento, almacenamiento de archivos, procesamiento del lado del servidor para cosas que deben protegerse de la interfaz y un sistema de autenticación, todo lo necesario para aplicaciones pequeñas y medianas.

Cloud Firestore es el servicio de base de datos de Google Firebase para aplicaciones móviles. Firestore permite una experiencia de programación increíble cuando se usa en un stack de aplicaciónes completo. Los datos en tiempo real y la facilidad de conexión a la base de datos hacen que FERN Stack sea una forma rápida de conectar estas tecnologías.

2.2.1. Componentes

- Firebase (base de datos)
- Express.js (servidor)
- React.js (cliente)
- Node.js (entorno del servidor)

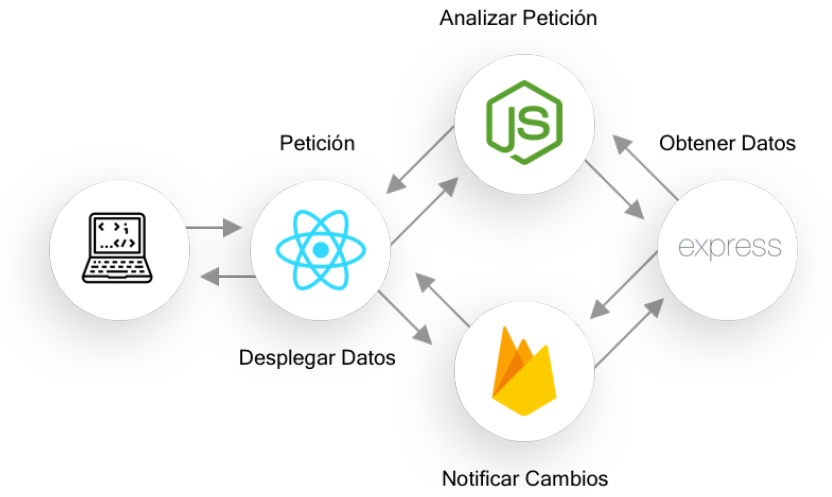

Figura 2.3: Flujo de informacion en FERN Stack.

2.2.2. Cloud Firestore

Firestore es una bases de datos orientada a documentos, toda la información se guarda en colecciones como JSON, principalmente diseñada para almacenar, recuperar y administrar información orientada a documentos, también conocida como datos semiestructurados.

La escalabilidad es completamente automática, lo que significa que no es necesario compartir sus datos en varias instancias. Los cargos de Cloud Firestore se basan en las operaciones realizadas en su base de datos (lectura, escritura, borrado), ancho de banda y almacenamiento. Admite límites de gasto diario para proyectos de Google App Engine, para garantizar que no exceda los costos con los que el usuario se sienta cómodo.

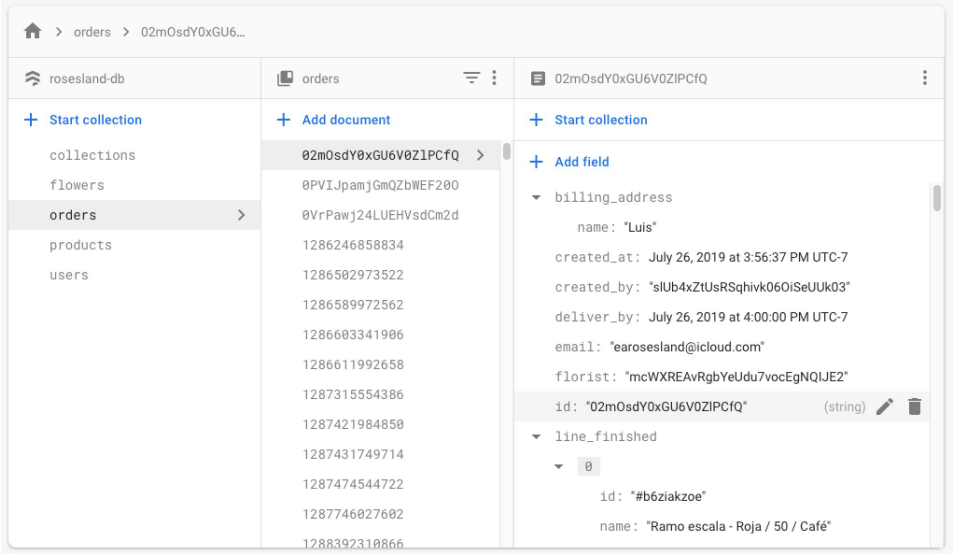

Figura 2.4: Colección de Firestore con sus documentos internos.

2.2.3. Node.js

Node.js es un entorno multiplataforma de código abierto para ejecutar código JavaScript del lado del servidor. El entorno de tiempo de ejecución de Node.js incluye todo lo que necesita para ejecutar un programa escrito en JavaScript.

Node.js surgió cuando los desarrolladores originales de JavaScript lo extendieron de algo que solo podía ejecutar en el navegador a algo que podría ejecutar en su máquina como una aplicación independiente. Ahora puede hacer mucho más con JavaScript que simplemente hacer que los sitios web sean interactivos. JavaScript ahora tiene la capacidad de hacer cosas que otros lenguajes de secuencias de comandos como Python pueden hacer. Tanto su navegador JavaScript como Node.js se ejecutan en el motor de tiempo de ejecución JavaScript V8. Este motor toma su código JavaScript y lo convierte en un código de máquina más rápido. El código de máquina es un código de bajo nivel que la computadora puede ejecutar sin necesidad de interpretarlo primero.

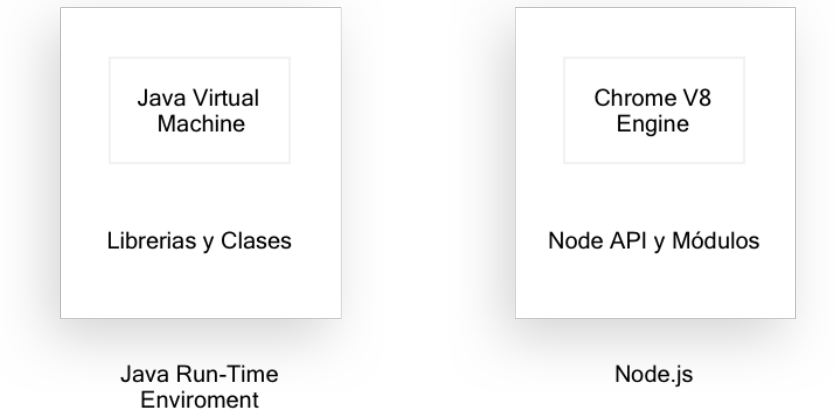

Figura 2.5: Analogía Node.js con Java.

Motor V8 de Google Chrome

Node.js utiliza el motor de ejecución ultra rápido V8 de Google Chrome. Hasta el lanzamiento de Chrome, la mayoría de los navegadores leían JavaScript de manera ineficiente: el código se leía e interpretaba poco a poco. Tomó mucho tiempo leer JavaScript y convertirlo a lenguaje máquina para que el procesador pudiera entenderlo.

El motor V8 de Google Chrome funciona completamente diferente. Está altamente optimizado y lleva a cabo lo que llamamos compilación JIT (Just In Time). Transforma rápidamente el código JavaScript en lenguaje máquina.

2.2.4. El ecosistema NPM

NPM (Node Package Manager) es el administrador de paquetes predeterminado para Node.js. se instala en el sistema con la instalación de Node.js. Los paquetes y módulos necesarios en un proyecto Node se instalan utilizando *npm*.

NPM consta de tres componentes:

1. Sitio web
2. Registro
3. CLI

Sitio web

El sitio web oficial de npm es https://www.npmjs.com/. Con este sitio web puede encontrar paquetes, ver documentación, compartir y publicar paquetes.

Registro

El registro npm es una gran base de datos que consta de más de medio millón de paquetes. Los desarrolladores descargan paquetes del registro npm y publican sus paquetes en el registro.

CLI (interfaz de línea de comando)

Esta es la línea de comando que ayuda a interactuar con el npm para instalar, actualizar y desinstalar paquetes y administrar dependencias.

2.2.5. Comandos NPM

Npm tiene muchos paquetes que puedes usar en una aplicación para que su desarrollo sea más rápido y eficiente. Instalar módulos usando NPM no representa un gran problema. Hay una sintaxis simple para instalar cualquier módulo Node.js:

```
npm install nombre-del-paquete
ejemplo: npm install express
```

2.2.6. Express.js

Escribir un servidor web completo a mano en Node.js directamente no es tan fácil, ni es necesario, Express.js es un paquete de aplicación web minimalista y extensible creado para el ecosistema Node.js. Permite crear un servidor web legible, flexible y fácil de mantener.

Express.js le permite definir rutas, especificaciones de qué hacer cuando llega una solicitud HTTP que coincide con un patrón determinado. La especificación coincidente se basa en expresiones regulares (regex) y es muy flexible, como la mayoría de los otros entornos de aplicaciones web. La parte de qué hacer es solo una función que recibe la solicitud HTTP analizada.

Express.js analiza la URL de solicitud, encabezados y parámetros. En el lado de la respuesta, tiene, como se esperaba, toda la funcionalidad requerida por las aplicaciones web. Esto incluye la configuración de códigos de respuesta, configuración de cookies, envío de encabezados personalizados, etc. Además, puede escribir middleware Express, piezas de código personalizadas que se pueden insertar en cualquier ruta de procesamiento de solicitud/respuesta para lograr una funcionalidad común como el registro, la autenticación, entre otras.

Código Ejemplo 2.1: Simple aplicación Express.js

```
const express = require('express');
const app = express();
const port = 3000;

app.get('/', (req, res) => res.send('Hola mundo!'));

app.listen(port, () => console.log(`Servidor iniciado el puerto ${port}`));
```

2.2.7. React.js

React es una biblioteca de JavaScript declarativa, eficiente y flexible creada en 2013 por el equipo de desarrollo de Facebook. React quería que las interfaces de usuario fueran más modulares (o reutilizables) y más fáciles de mantener. Según el sitio web de React, se utiliza para *construir componentes encapsulados que administran su propio estado, y unirlos para crear interfaces de usuario complejas*. React es una biblioteca JavaScript que permite componer interfaces de usuario complejas a partir de piezas de código pequeñas y aisladas llamadas *componentes*.

En terminos generales, al crear aplicaciones con React.js, se crean componentes que corresponden a distintos elementos de una interfaz de usuario. Después se organizan estos elementos dentro de componentes de orden superior que definen la estructura de la aplicación. Es importante destacar que cada componente en una aplicación React se rige por principios estrictos de gestión de datos. Interfaces avanzadas comunmente involucran datos complejos y manejo de estado. React.js es limitado y tiene como objetivo darnos las herramientas para poder anticipar cómo se verá una aplicación con un conjunto de circunstancias dado.

2.2.8. Componentes React

Un componente es una pequeña parte de la interfaz de usuario. Todas las piezas reutilizables de una página web se abstraen en un componente.

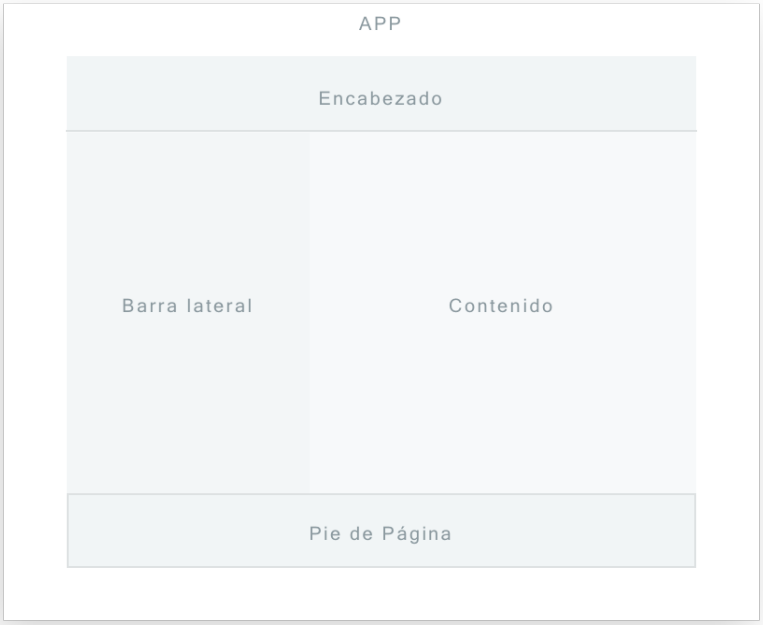

Figura 2.6: Componentes principales de una página web.

En primer lugar, hay un componente principal llamado componente APP. Este componente de la aplicación contiene cuatro componentes secundarios o se divide en cuatro componentes:

1. Encabezado
2. Barra lateral
3. Contenido
4. Pie de página

La función de cada componente se manejará independientemente con otros componentes. Cada componente es una pieza reutilizable, y se puede pensar en cada componente de forma aislada.

Dentro de un componente, tendremos subcomponentes o componentes dentro de un componente padre. Esos serán reutilizables también.

Código Ejemplo 2.2: Ejemplo de página con componentes React.js

```
class Header extends React.Component {
  render() {
    return (
      <header className="navigation">
        {this.props.name}
      </header>
    );
  }
}

class Content extends React.Component {
  render() {
    return (
      <div className="content">
        Hola Mundo
      </div>
    );
  }
}

class App extends React.Component {
  render() {
    return (
      <div>
        <Header name="Mi App" />
        <Content />
      </div>
    );
  }
}
```

2.2.9. DOM Virtual

React utiliza un DOM virtual, que es una representación virtual del DOM (Document Object Model). Detrás de escena, React hace un gran trabajo para editar y volver a renderizar eficientemente el DOM cuando algo en la interfaz necesita cambiar.

2.3. Preprocesamiento

2.3.1. ES5/ES6

ES5 (ES significa ECMAScript) es básicamente 'JavaScript normal'. La quinta actualización de JavaScript, ES5 se finalizó en 2009. Ha sido compatible con todos los principales navegadores durante muchos años.

ES6 es una nueva versión de JavaScript que agrega algunas buenas adiciones sintácticas y funcionales. Se finalizó en 2015. ES6 es casi totalmente compatible con todos los navegadores principales. Pero pasará algún tiempo hasta que las versiones anteriores de los navegadores web estén fuera de uso. Por ejemplo, Internet Eyplorer 11 no es compatible con ES6, pero tiene aproximadamente el 8 % de uso de mercado de navegadores.

Para aprovechar los beneficios de ES6 hoy, se tienen que hacer que hacer algunos procedimientos para que funcione en tantos exploradores de internet como sea posible:

1. Se debe preprocesar el código para que una gama más amplia de navegadores entiendan nuestro JavaScript. Esto significa convertir ES6 JavaScript en ES5 JavaScript.
2. Tenemos que incluir un 'shim' o 'polyfill' que brinde una funcionalidad adicional agregada en ES6 que un navegador puede o no tener.

JSX

JSX es una extensión de sintaxis similar a XML para ECMAScript sin ninguna semántica definida. NO está destinado a ser implementado por motores o navegadores. NO es una propuesta para incorporar JSX en la propia especificación ECMAScript. Está destinado a ser utilizado por varios preprocesadores (transpiladores) para transformar estos tokens en ECMAScript estándar.

BabelJS

BabelJS es un transpilador de JavaScript que transpila nuevas características ES6 al antiguo estándar ES5. Con esto, las funciones se pueden ejecutar en navegadores antiguos y nuevos, sin problemas.

El preprocesador BabelJS convierte la sintaxis de JavaScript moderno en un formulario, que los navegadores más antiguos pueden entender fácilmente. Por ejemplo, const y let se convertirán en var, la función flecha se convierte en una función normal manteniendo la funcionalidad igual en ambos casos.

Código Ejemplo 2.3: Ejemplo de sintaxis ES5 y ES6

```
Funciones

// ES5
var cuadrado = function cuadrado(num) {
  return num * num;
};

// ES6
const cuadrado = num => num * num;


Acceder a los valores de un objeto

var obj1 = { a: 1, b: 2 };

// ES5
var a = obj1.a;
var b = obj1.b;

// ES6
var { a, b } = obj1;
```

Sass (Syntactically Awesome Style Sheets)

Sass es un preprocesador de CSS, que ayuda a reducir la repetición con CSS y ahorra tiempo. Es un lenguaje de extensión CSS más estable y potente que describe el estilo de una página estructuralmente. Sus principales atributos son:

1. Es un súper conjunto de CSS, lo que significa que contiene todas las características de CSS y es un preprocesador de código abierto, codificado en Ruby.
2. Proporciona algunas características, que se utilizan para crear hojas de estilo que permiten escribir código más eficiente y fácil de mantener.

Webpack

Webpack es un empaquetador de módulos. Webpack toma un archivo de entrada, encuentra todos los archivos de los que depende y genera un archivo que contiene todo el código de una aplicación. Con él es posible importar archivos CSS e imágenes directamente a JavaScript. Se puede compilar CoffeeScript, TypeScript, SASS y LESS. También es capaz de compilar la sintaxis de ES6 en JavaScript amigable para el navegador. En otras palabras, Webpack toma diferentes archivos (como CSS, JS, SASS, JPG, SVG, PNG, etc.) y los combina en paquetes, un paquete separado para cada tipo de archivo.

2.4. E-commerce (Comercio electrónico)

El comercio electrónico se refiere al proceso de compra o venta de productos o servicios a través de Internet. Las compras en línea se están volviendo cada vez más populares debido a la velocidad y facilidad de uso para los clientes. Las actividades de comercio electrónico, como la venta en línea, pueden dirigirse a consumidores u otras empresas. Vender en línea puede ayudar a su empresa a llegar a nuevos mercados y aumentar sus ventas e ingresos (ya sea a través de su propio sitio web o de un sitio de mercado electrónico).

Shopify

Shopify es un servicio web que le permite configurar una tienda en línea para vender sus productos. Le da la facilidad organizar sus productos, personalizar el diseño de su tienda, aceptar pagos con tarjeta de crédito, rastrear y responder a pedidos. Shopify.com permite a los vendedores elegir entre opciones de diseño gratuitas o diseños personalizados creados por los usuarios.