UNIVERSIDAD DEL VALLE DE GUATEMALA
Ingeniería de Software 1, sección 30

TAREA INVESTIGATIVA: FRAMEWORKS

VENTAS E INVENTARIO DIVINO SEAS

Sofia Garcia - 22210
Julio Garcia Salas - 22076
Joaquin Campos - 22155
Juan Fernando Menéndez - 17444
Juan Pablo Solis - 22102

GUATEMALA, marzo de 2024

Resumen

Este documento ofrece un vistazo detallado a los frameworks React y Express.js, destacando su relevancia y funcionalidad dentro del proyecto "Ventas e Inventario Divino Seas". React, asignado a Juan Fernando Menéndez, Juan Pablo Solis y Joaquin Campos, se examina en términos de su estructura basada en componentes, el uso del Virtual DOM y sus patrones de diseño, revelando su eficacia para construir interfaces de usuario dinámicas y reactivas. Por otro lado, Express.js, a cargo de Sofía y Julio, se presenta como un framework minimalista y flexible para Node.js, ideal para la creación de aplicaciones web y APIs rápidas y escalables. A través de una organización efectiva y un reparto de responsabilidades claro, el equipo planea desarrollar un informe comprensivo que no solo detalle los aspectos técnicos de estos frameworks, sino que también proporcione ejemplos prácticos y recomendaciones basadas en su experiencia. Este análisis busca no solo cumplir con los requisitos académicos de la asignatura sino también servir como un recurso valioso para futuros proyectos de desarrollo en el ámbito de ventas e inventario.

Introducción

Este análisis forma parte de un avance de tarea investigativa bajo el título "Ventas e Inventario Divino Seas", con el propósito de proporcionar una base sólida para la implementación de soluciones tecnológicas en dicho contexto. A través de una planificación meticulosa y una asignación de tareas estratégica, el equipo abordará desde los fundamentos hasta los casos de uso prácticos de ambos frameworks, ofreciendo así una perspectiva integral sobre su aplicabilidad y eficacia en el desarrollo de proyectos de software.

Frameworks seleccionados

Encargados	Framework
Juanfer, Juan Pablo y Joaquín	React
Sofía y Julio	Express JS

Desglose y asignación de tareas

Planificación General

Fecha de inicio: 8 de marzo
Fecha de finalización: 15 de marzo a las 5 p.m.
Tiempo total disponible: 7 días

Preparación y Organización del Proyecto (8 de marzo)

- Tarea: Definir estructura del informe
    - Encargado: Juanfer
    - Descripción: Crear la estructura base del documento con secciones para resumen, introducción, desarrollo (React y Express.js), conclusiones y bibliografía.

Investigación Preliminar (8-9 de marzo)

React

- Fundamentos de React (Juan Pablo)
    - Buscar información sobre los principios básicos de React, incluyendo su JSX, componentes y estado.
- Características de React (Joaquín)
    - Recopilar información sobre características clave como el Virtual DOM, componentes reutilizables y props.
- Casos de Uso de React (Juanfer)
    - Identificar y describir ejemplos de aplicaciones web que utilizan React.

Express.js

- Fundamentos de Express.js (Sofía)
    - Investigar los aspectos básicos de Express.js, incluyendo routing y middleware.
- Características de Express.js (Julio)
    - Detallar las características principales de Express.js, como su simplicidad y flexibilidad para construir aplicaciones web y API.

Recopilación y Organización de la Información (10-11 de marzo)

React

- Ejemplos y Código de React (Joaquín)
    - Buscar ejemplos de código de React que ilustran su uso en aplicaciones reales.
- Gráficos y Esquemas de React (Juan Pablo)
    - Encontrar o crear gráficos y esquemas que faciliten la comprensión de React.

Express.js

- Ejemplos y Código de Express.js (Sofía)
    - Recopilar ejemplos de código de Express.js, mostrando su aplicación en el backend.
- Gráficos y Esquemas de Express.js (Julio)
    - Buscar o elaborar gráficos y esquemas que expliquen el funcionamiento de Express.js.

Redacción del Documento (12-13 de marzo)

Desarrollo del Tema (React)

- Integración de la Información de React (Joaquín, Juan Pablo, Juanfer)

    - Redactar la sección de desarrollo sobre React, integrando la investigación realizada.

Desarrollo del Tema (Express.js)

- Integración de la Información de Express.js (Sofía, Julio)
    - Componer la sección de desarrollo sobre Express.js, utilizando la información recopilada.

Revisión y Edición (14 de marzo)

Revisión General del Documento (Juan Pablo)

- Revisar la coherencia y cohesión del documento, asegurando que cumple con los requisitos.

Corrección de Gramática y Ortografía (Sofía)

- Corregir errores gramaticales y ortográficos en todo el documento.

Preparación para la Presentación (15 de marzo)

Preparar Materiales de Soporte (Julio)

- Crear diapositivas o cualquier otro material de soporte para la presentación.

Investigación

React

Descripción General

Principios de funcionamiento y componentes

React utiliza un Virtual DOM para optimizar las actualizaciones en la interfaz de usuario, DOM virtual es un concepto de programación donde una representación ideal de la IU se mantiene en memoria y en sincronía con el DOM "real" lo que se conoce como reconciliación.

Entre algunas de las ventajas se encuentra la mejora en rendimiento, ciclo de vida simplificado, reutilización de componentes, compatibilidad con otras bibliotecas, entre otras.

React se basa en componentes, estos son unidades autónomas que guardan la lógica y la interfaz de usuario, estos pueden contener su estado y se pueden componer entre sí para construir aplicaciones complejas.

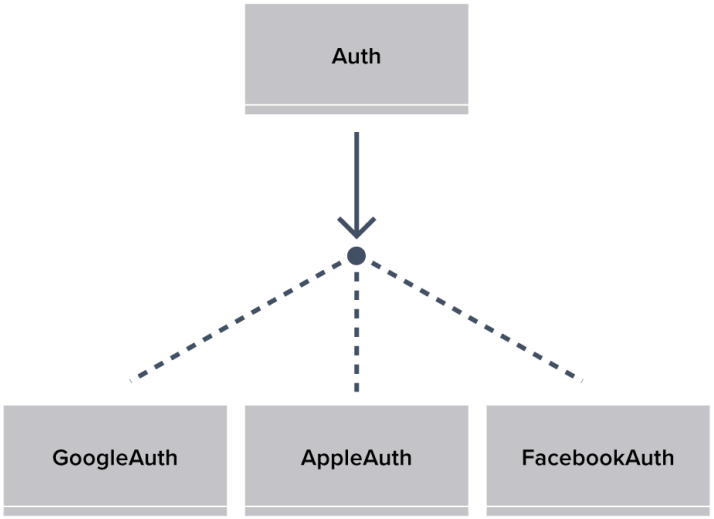
Sus componentes son funcionales y de clase, de manera que se basan simplemente en funciones y se consideran más complejos con estado y ciclo de vida.

Tipo de framework y cobertura de desarrollo

React es un framework de interfaz de usuario (UI) de JavaScript que cubre principalmente el desarrollo de la capa de presentación de una aplicación web. Aunque puede ser utilizado para desarrollar aplicaciones completas de una sola página, es común combinarlo con otras tecnologías como Redux para gestionar el estado de la aplicación y React Router para la navegación.

Patrones de diseño

React implementa el patrón de diseño de componentes, que promueve la creación de interfaces de usuario modulares y reutilizables. Además, se puede integrar con patrones arquitectónicos como Flux o Redux para gestionar el estado de la aplicación de manera más eficiente. Estos patrones se implementan principalmente en la estructura y organización de los componentes y en la gestión del estado de la aplicación.



Situaciones recomendadas

React implementa el patrón de diseño de componentes, que promueve la creación de interfaces de usuario modulares y reutilizables. Además, se puede integrar con patrones arquitectónicos como Flux o Redux para gestionar el estado de la aplicación de manera más eficiente. Estos patrones se implementan principalmente en la estructura y organización de los componentes y en la gestión del estado de la aplicación.

Semejanzas y diferencias

Tanto React como otros frameworks como Angular y Vue.js son utilizados para el desarrollo de interfaces de usuario. Sin embargo, React se destaca por su enfoque en la creación de componentes reutilizables y su filosofía de "aprender una vez, escribir en todas partes". A diferencia de Angular, que es un framework más completo y opinionado, React ofrece más flexibilidad y se integra fácilmente con otras bibliotecas y herramientas. Comparado con Vue.js, React tiende a ser más adecuado para proyectos de mayor escala y complejidad, mientras que Vue.js es más sencillo y rápido de aprender para proyectos más pequeños.

Ejemplos de uso

1. Facebook: React fue creado inicialmente por Facebook y es utilizado en gran parte de su plataforma, incluyendo el feed de noticias, los perfiles de usuario y la sección de comentarios.

2. Instagram: Propiedad de Facebook, utiliza React en su interfaz de usuario para proporcionar una experiencia interactiva y receptiva a sus usuarios.

3. WhatsApp: La versión web de WhatsApp utiliza React para crear una interfaz de usuario dinámica y fluida que permite a los usuarios chatear y compartir contenido desde sus navegadores.

4. Netflix: Utiliza React en su sitio web para ofrecer una experiencia de usuario consistente y optimizada al navegar por su extenso catálogo de contenido y ver videos en streaming.

5. Airbnb: Utiliza React en su plataforma web para permitir a los usuarios buscar, reservar y gestionar alojamientos de forma fácil e intuitiva.

6. Dropbox: Usa React en su interfaz web para proporcionar a los usuarios una experiencia de usuario fluida y eficiente al acceder y gestionar sus archivos en la nube.

7. Uber: Implementa React en su plataforma web para proporcionar una experiencia de reserva de viajes intuitiva y eficiente a sus usuarios.

Express JS

Nombre y descripción general

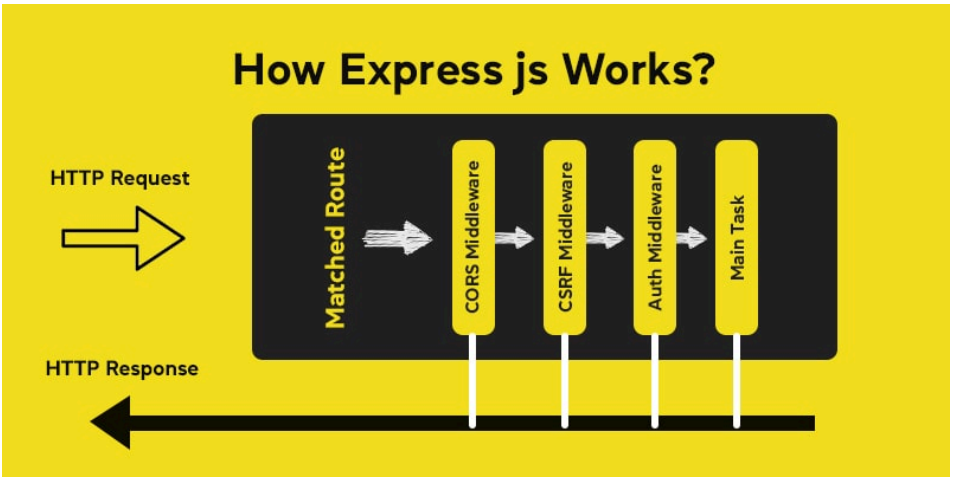
Express.js es un framework de aplicación web para Node.js, diseñado para facilitar la creación de aplicaciones web y APIs. Su naturaleza minimalista y flexible lo hace adecuado para una amplia gama de proyectos, desde simples páginas web hasta complejas aplicaciones de una sola página (SPA) y APIs RESTful.

Principios de Funcionamiento y Componentes

Express.js se basa en el patrón middleware, que interviene en el procesamiento de solicitudes y respuestas HTTP. Los componentes clave son:

- Aplicación: El núcleo de Express, que inicia un servidor y define rutas.
- Rutas: Determinan cómo responde la aplicación a una solicitud en un endpoint particular.
- Middleware: Funciones que acceden y modifican los objetos de solicitudes y respuestas.
- Motor de Vistas: Permite renderizar vistas en el servidor antes de enviar el HTML al cliente

Diagrama de como funciona Express.js



Solicitud HTTP

Todo comienza con una solicitud HTTP enviada por un cliente, que podría ser un navegador web o una aplicación. Esta solicitud se dirige a un servidor donde Express.js está corriendo.

Ruta Coincidente

Express.js examina la solicitud para determinar cuál de las rutas definidas por el desarrollador coincide con la URL y el método HTTP (GET, POST, etc.) de la solicitud. Una vez encontrada la ruta correspondiente, Express comienza a procesar la solicitud.

Middleware CORS

El primer middleware que se muestra en el diagrama es el que maneja el intercambio de recursos de origen cruzado (CORS). Este es un mecanismo que permite o restringe los recursos solicitados en un servidor web dependiendo de dónde se originó la solicitud HTTP. Es una práctica de seguridad importante en el desarrollo web.

Middleware CSRF

El siguiente es un middleware para la protección de falsificación de solicitud en sitios cruzados (CSRF). Este tipo de middleware valida que las solicitudes hechas al servidor son legítimas y no han sido alteradas o iniciadas en sitios maliciosos.

Middleware de Autenticación (Auth)

Luego sigue un middleware de autenticación, que verifica si el usuario que hace la solicitud tiene permiso para realizar la acción solicitada. Este middleware podría verificar tokens, sesiones o cualquier otro mecanismo de autenticación.

Tarea Principal

Finalmente, la tarea principal se refiere a la lógica de negocio que se debe ejecutar si se pasan todos los chequeos de middleware anteriores. Esto podría ser cualquier cosa, desde servir una

página web, procesar un formulario, interactuar con una base de datos, hasta enviar una respuesta JSON.

Respuesta HTTP

Después de pasar por la tarea principal y cualquier otro middleware que pueda estar en la cadena, Express.js envía una respuesta HTTP de vuelta al cliente, completando así el ciclo de solicitud-respuesta.

Código de ejemplo

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

app.get('/', (req, res) => {
  res.send('Hola Mundo con Express.js');
});

app.listen(3000, () => {
  console.log('Aplicación escuchando en el puerto 3000');
});
```

Tipo de Framework y Cobertura del Proceso de Desarrollo

Express.js es clasificado como un framework de servidor backend para Node.js. Ofrece un amplio conjunto de características para aplicaciones web y móviles y es parte de la pila MEAN y MERN (MongoDB, Express.js, Angular/React, y Node.js), que cubre todo el desarrollo de aplicaciones de pila completa.

Como framework, Express.js ofrece una capa delgada de características fundamentales para aplicaciones web, sin imponer una estructura rígida, dejando a los desarrolladores la libertad de elegir las mejores herramientas y prácticas según las necesidades del proyecto. Esto significa que, aunque Express.js maneja la interacción entre el servidor y el cliente, para tareas como la interfaz de usuario, la persistencia de datos, y ciertas funcionalidades del lado del servidor como la autenticación, los desarrolladores suelen recurrir a bibliotecas y frameworks adicionales.

Cobertura en el Proceso de Desarrollo

- Enrutamiento: Define rutas de aplicación y métodos HTTP, ofreciendo una interfaz sencilla para dividir la lógica del servidor en múltiples rutas.
- Middleware: Utiliza funciones intermedias para extender funcionalidades y manejar solicitudes y respuestas.
- Manejo de errores: Provee un mecanismo robusto para capturar y manejar errores.
- Integración con Bases de Datos: Soporta una amplia variedad de bases de datos a través de módulos adicionales.

- Plantillas: Permite utilizar motores de plantillas para generar HTML y servir interfaces de usuario.

Express.js no cubre el desarrollo de la interfaz de usuario (front-end), la lógica de negocio directa, o la persistencia de datos directamente, aunque facilita la conexión con bases de datos a través de middlewares como Mongoose para MongoDB.

Patrones de Diseño y Arquitectónicos

Express.js permite la implementación del patrón Modelo-Vista-Controlador (MVC), que separa los datos de una aplicación (el modelo) de la lógica de control (el controlador) y la interfaz de usuario (la vista). El framework no impone un patrón MVC, pero su arquitectura es compatible con este, permitiendo así la creación de aplicaciones escalables y fáciles de mantener.

El framework también se basa en el patrón Middleware, que permite a los desarrolladores ejecutar cualquier código, hacer cambios en la solicitud y el objeto de respuesta, y finalizar el ciclo de solicitud-respuesta. Los middlewares pueden ser utilizados para tareas como:

- Validar solicitudes: Verificación de datos de entrada.
- Autorización y autenticación: Controlar el acceso a ciertas rutas.
- Manejo de errores: Capturar y gestionar errores de forma centralizada.

Implementación de Patrones

- Middlewares básicos: Se incluyen en Express.js y son suficientes para tareas simples de middleware.
- Middlewares de terceros: Para funcionalidades adicionales, como manejo de sesiones, seguridad CSRF, CORS, etc., se suelen utilizar middlewares de terceros.

Situaciones Recomendadas para su Uso

- Desarrollar APIs RESTful de manera eficiente.
- Crear aplicaciones web server-side renderizadas.
- Prototipos rápidos y proyectos ágiles.
- SPA y PWA en combinación con frameworks de frontend.

Semejanzas y Diferencias con Otros Frameworks

Al comparar Express.js con otros frameworks de Node.js como Koa y Fastify, se pueden identificar las siguientes semejanzas y diferencias:

Semejanzas

- Asincronía: Todos aprovechan el modelo de I/O no bloqueante de Node.js.
- Middleware: Utilizan una arquitectura basada en middleware.
- Comunidad: Cuentan con una comunidad activa y ofrecen una gran cantidad de plugins.

Diferencias

- Curva de aprendizaje: Express.js tiene una curva de aprendizaje más suave en comparación con Koa y Fastify.
- Flexibilidad vs. Rendimiento: Mientras que Express.js se enfoca en ser minimalista y flexible, Fastify se centra en la velocidad y el rendimiento.
- Modernidad: Koa fue creado por el equipo detrás de Express para ser más moderno y aprovechar las nuevas características de JavaScript como async/await.
- Ecosistema: Express.js tiene el ecosistema más grande de los tres, en términos de middleware y tutoriales disponibles.

Referencias

Alaburda, V. (2023, July 25). *Why is React a library and Next.js a framework?* Built In. https://builtin.com/software-engineering-perspectives/react-framework

React – A JavaScript library for building user interfaces. (n.d.). React. https://legacy.reactjs.org/

Documentación oficial de Express.js. (s. f.). Recuperado de http://expressjs.com/

MDN Web Docs. (s. f.). Express Tutorial Part 3: Using a Database (with Mongoose). Recuperado de https://developer.mozilla.org/en-US/docs/Learn/Server-side/Express_Nodejs/mongoose

MDN Web Docs. (s. f.). Express Tutorial Part 4: Routes and controllers. Recuperado de https://developer.mozilla.org/en-US/docs/Learn/Server-side/Express_Nodejs/routes

MDN Web Docs. (s. f.). Express/Node introduction. Recuperado de https://developer.mozilla.org/en-US/docs/Learn/Server-side/Express_Nodejs/Introduction

MDN Web Docs. (s. f.). Express web framework (Node.js/JavaScript). Recuperado de https://developer.mozilla.org/en-US/docs/Learn/Server-side/Express_Nodejs

MDN Web Docs. (s. f.). Server-side web frameworks. Recuperado de https://developer.mozilla.org/en-US/docs/Learn/Server-side/First_steps/Web_frameworks

Node.js Foundation. (s. f.). Documentación de Node.js. Recuperado de https://nodejs.org/en/docs/