Documentación del Reto Café Ole

Descripción del Proyecto

El proyecto de hacer una empresa de cafeterías tiene como objetivo desarrollar una página web interactiva para promocionar y vender productos. El trabajo se ha realizado utilizando React que es una biblioteca muy popular y ampliamente utilizada en el desarrollo de interfaces de usuario modernas y dinámicas y GitHub que se implementó para administrar y controlar las versiones del código. Esto implicó la configuración de dos repositorios en GitHub que son para el cliente y para el servidor, el uso de ramas para trabajar en diferentes características y la utilización de herramientas de control de versiones para fusionar y resolver conflictos. La integración de GitHub permitió un flujo de trabajo colaborativo y aseguró una mayor transparencia en el desarrollo del proyecto. Para la parte del servidor utilizamos SpringBoot que es un framework desarrollado para el trabajo con Java como lenguaje de programación y facilita la creación de todo tipo de aplicaciones basadas en él de manera independiente con el mínimo esfuerzo.

El uso de React presentó desafíos interesantes durante el desarrollo del proyecto. Uno de los principales desafíos fue comprender y aplicar correctamente los conceptos fundamentales de React, como los componentes, el estado y los ciclos de vida. La arquitectura basada en componentes de React permite dividir la interfaz de usuario en piezas reutilizables, lo que facilita la gestión y el mantenimiento del código. Sin embargo, fue necesario comprender cómo estructurar y comunicar eficientemente los componentes para lograr una aplicación coherente y sin errores.

Estructura del Repositorio

Cliente:

El repositorio del cliente contiene todo el código relacionado con la parte del front-end de la aplicación web. Aquí se encuentran los archivos y directorios que componen la interfaz de usuario y la lógica de presentación. La estructura típica del repositorio del cliente puede incluir los siguientes elementos:

- **Parte de fol:** Contiene todos los archivos de la parte realizada de fol: prevención de riesgos y el video de prevención de riesgos.
- **src:** Este directorio contiene los archivos fuente de la aplicación.
    - **Assets:** En este directorio se encuentran las fotos que se han usado en la pagina.
    - **Components:** En este directorio se encuentran los componentes reutilizables de React utilizados en la interfaz de usuario. Cada componente se encuentra en su propio archivo para facilitar la reutilización y el mantenimiento.
    - **Errors:** En este directorio se encuentran los archivos de errores.

    - **Estilos:** En este directorio se encuentra todos los estilos que se han empleado en la pagina.
    - **Layout:** En este directorio se encuentra el header y el footer.
    - **Modals:** En este directorio se encuentran diferentes funciones de la pagina
    - **Pages:** En este directorio se encuentran las diferentes páginas de la aplicación. Cada página puede tener su propio archivo que contiene la lógica y la estructura de esa página en particular.
    - **Scripts:** En este directorio se encuentran diferentes scripts para la pagina
    - **main.jsx:** Este archivo es para que las rutas de la pagina funcionen.
- **Public:** Este directorio se utiliza para almacenar los archivos estáticos que serán accesibles desde la aplicación web.
- **.eslintrc.cjs:** Este archivo es utilizado por ESLint, una herramienta de análisis estático de JavaScript, para configurar las reglas y configuraciones de estilo de código.
- **.gitignore:** Este archivo especifica los archivos y directorios que deben ser ignorados por Git, el sistema de control de versiones. Los archivos y directorios listados aquí no se incluirán en el control de versiones.
- **index.html:** Este archivo es la página principal de la aplicación web. Contiene la estructura básica de HTML y puede incluir referencias a los archivos JavaScript y CSS necesarios para la aplicación.
- **package-lock.json:** Este archivo es generado por npm (Node Package Manager) y se utiliza para bloquear las versiones exactas de las dependencias instaladas en el proyecto. Ayuda a mantener la consistencia en las versiones de las dependencias entre diferentes entornos de desarrollo.
- **package.json:** Este archivo es el archivo de configuración de npm. Contiene información sobre el proyecto, como las dependencias utilizadas, los scripts de construcción y las configuraciones específicas del proyecto.
- **readme.md:** Este archivo es una descripción o documentación del proyecto. Suele contener información sobre la configuración, instalación y uso de la aplicación.
- **vite.config.js:** Este archivo es la configuración del proyecto para Vite, una herramienta de construcción rápida para aplicaciones web. Puede contener ajustes específicos de construcción y configuraciones del entorno de desarrollo.

Servidor:

El repositorio del servidor contiene el código relacionado con la parte del back-end de la aplicación web. Aquí se encuentran los archivos y directorios que se encargan de la lógica de negocio, la comunicación con la base de datos y cualquier otro proceso del lado del servidor. La estructura típica del repositorio del servidor puede incluir los siguientes elementos:

- **Base de datos consultas:** Este directorio contiene los archivos relacionados con las creaciones de tablas de la base.
- **Insertar:** Este archivo contiene instrucciones SQL o scripts relacionados con la inserción de datos en la base de datos.
- **demo:** Este directorio contiene la programación para la base de datos.
    - **Src:** Este directorio contiene los archivos fuente de la aplicación.
        - **Main:** proyecto de java
            - **java/com/example/demo**
                - **Controller:** Este directorio almacena las clases responsables de manejar las solicitudes HTTP y las respuestas del servidor. Los controladores son componentes clave en una arquitectura MVC (Modelo-Vista-Controlador) y se encargan de recibir las solicitudes del cliente, procesar los datos y devolver una respuesta adecuada.
                - **Entity:** Aquí se encuentran las clases que representan las entidades del dominio del proyecto. Las entidades son objetos que mapean a tablas en una base de datos o representan conceptos principales dentro del sistema. Estas clases generalmente contienen atributos y métodos para acceder y manipular los datos de las entidades.
                - **Repository:** Aquí alberga las interfaces o clases que definen las operaciones de acceso a datos. Estas operaciones incluyen consultas y manipulación de la base de datos, como inserción, actualización y eliminación de registros. Los repositorios proporcionan una capa de abstracción entre la capa de servicio y la capa de persistencia.
                - **Service:** Aquí se encuentran las clases que contienen la lógica de negocio o la lógica de aplicación del proyecto. Estas clases encapsulan las operaciones y la lógica necesaria para manipular y procesar los datos del sistema. Los servicios interactúan con los controladores y los repositorios para realizar operaciones específicas.
                - **RunServer.java:** Este archivo representa una clase principal o punto de entrada del servidor. Es el punto de inicio de la aplicación y contiene el método "main" desde donde se inicia el servidor y se configura el entorno de ejecución.

        - **test/java/com/example/demo/demo:** Se encuentran las pruebas unitarias o de integración del proyecto. Estas pruebas se utilizan para verificar el comportamiento y la funcionalidad de las clases y métodos implementados. Las pruebas unitarias permiten identificar y solucionar problemas de manera temprana, garantizando la calidad del código y el correcto funcionamiento de la aplicación.
- **BBDDTFG.drawio:** Este archivo representa un diagrama actualizado de la base de datos del proyecto utilizando la herramienta Draw.io. Proporciona una representación visual de la estructura y relaciones de las tablas en la base de datos.
- **FinalDB.sql:** Este archivo representa la exportación de la base de datos.
- **backup.sql:** Este archivo representa la antigua base de datos.
- readme.md: Este archivo readme.md ha sido actualizado y contiene información o documentación relacionada con el servidor y la base de datos. Puede proporcionar instrucciones de configuración, detalles sobre el funcionamiento del servidor o cualquier otra información relevante.
- **tfg.sql:** Este archivo representa la antigua base de datos.
- **tfgActualizada.sql:** Este archivo es un script SQL que representa una versión actualizada de la base de datos

Tecnologías Utilizadas

React: React es una biblioteca de JavaScript ampliamente utilizada para construir interfaces de usuario interactivas y reactivas. Proporciona una forma eficiente de renderizar componentes y manejar el estado de la aplicación. En este proyecto, se eligió React como el framework principal para desarrollar la parte del cliente de la aplicación web.

GitHub: GitHub es una plataforma de alojamiento de código fuente y colaboración basada en Git. Se utilizó GitHub para almacenar y gestionar el código fuente del proyecto, permitiendo a los miembros del equipo colaborar, realizar seguimiento de cambios y versiones, y facilitar la integración y despliegue continuo.

HTML (HyperText Markup Language): HTML es el lenguaje de marcado estándar utilizado para estructurar y presentar contenido web. En el proyecto, se utilizó HTML para definir la estructura y el diseño de las páginas web.

CSS (Cascading Style Sheets): CSS es un lenguaje utilizado para definir el estilo y la apariencia visual de los elementos HTML en una página web. Se utilizó CSS para aplicar estilos personalizados, como colores, fuentes, diseños y efectos visuales, a la interfaz de usuario del proyecto.

Node.js: Node.js es un entorno de ejecución de JavaScript en el lado del servidor. Se utilizó Node.js para el desarrollo del servidor en el proyecto, permitiendo la creación

de una API RESTful para la comunicación entre el cliente y la base de datos, así como la ejecución de operaciones en el servidor.

Base de datos (SQL): Para almacenar y gestionar los datos del proyecto, se utilizó una base de datos SQL. Aunque no se mencionan detalles específicos sobre el sistema de gestión de bases de datos utilizado, es común utilizar sistemas como MySQL, PostgreSQL o SQLite para proyectos basados en Node.js.

npm: npm es el administrador de paquetes predeterminado para Node.js y se utiliza para gestionar las dependencias del proyecto. Permite instalar, actualizar y desinstalar paquetes de software reutilizables de manera sencilla. Los paquetes npm son módulos de código que pueden ser utilizados en el proyecto para agregar funcionalidades específicas o facilitar tareas comunes.

SpringBoot: Spring Boot es un framework de desarrollo en Java que simplifica la creación de aplicaciones empresariales, proporcionando configuración automática y características integradas, lo que permite a los desarrolladores concentrarse en la lógica de la aplicación.

Instrucciones para Ejecutar el Proyecto

Clonar el repositorio: Hemos clonado el repositorio con Visual Studio Code. Para hacerlo hay que darle ctrl + shift + p y darle a clonar en git hub.

Navegar al directorio del proyecto: cd nombre-del-repositorio

Instalar las dependencias del proyecto: npm i

Acceder a la aplicación en el navegador web: npm run dev

Flujo de Trabajo en GitHub

Creación del repositorio en GitHub.

Clonación del repositorio en local: hecho con Visual. Ctrl + shift + p. Luego clone git hub

Creación de una rama de desarrollo: abajo a la izquierda y le das créate new Branch

Commits: a la izquierda en el apartado Source Control podras crear commits

Actualización de la rama principal del repositorio: git pull

Obtener los últimos cambios de un repositorio remoto sin fusionarlos automáticamente con tu rama actual: git fetch