CARRERA

NRC - ASIGNATURA:	14564 – Desarrollo web avanzado
PROFESORA:	Ing. DIEGO MEDARDO SAAVEDRA GARCIA
PERÍODO ACADÉMICO:	2024_50

Prueba parcial

TÍTULO:

Evaluación Final 2do Parcial

ESTUDIANTE

Erick Mijail Andrade Pichoasamin

FECHA DE ENTREGA: 21 / Julio / 2024

CALIFICACIÓN OBTENIDA:

Documentación del Proyecto "Task Manager"

I. Introducción

A. Descripción del Proyecto

El proyecto "Task Manager" es una aplicación web diseñada para facilitar la gestión de tareas de manera eficiente y organizada. Utiliza Next.js, un framework basado en React que permite la creación de aplicaciones web tanto estáticas como dinámicas. Este proyecto fue iniciado utilizando create-next-app, una herramienta que permite el rápido inicio de proyectos con Next.js, asegurando una configuración inicial óptima y lista para el desarrollo.

Next.js ofrece características como el renderizado del lado del servidor (SSR), generación de sitios estáticos (SSG), y una experiencia de desarrollo altamente optimizada. Además, "Task Manager" aprovecha Tailwind CSS para el diseño y estilizado de la interfaz de usuario, proporcionando un enfoque de diseño utilitario que permite crear componentes estilizados de manera rápida y eficiente. TypeScript también se utiliza para asegurar la calidad del código mediante la adición de tipos estáticos, lo que facilita el desarrollo y mantenimiento del código a largo plazo.

B. Objetivos

El objetivo principal de "Task Manager" es proporcionar a los usuarios una plataforma intuitiva y eficiente para gestionar sus tareas diarias. Los objetivos específicos incluyen:

1. **Facilidad de Uso:** Crear una interfaz de usuario amigable y fácil de usar, que permita a los usuarios agregar, editar, y eliminar tareas sin complicaciones.
2. **Eficiencia:** Utilizar tecnologías modernas que permitan un rendimiento óptimo de la aplicación, asegurando una experiencia de usuario rápida y sin interrupciones.
3. **Escalabilidad:** Diseñar la arquitectura del proyecto de manera que pueda escalar fácilmente para soportar un mayor número de usuarios y funcionalidades en el futuro.
4. **Mantenibilidad:** Asegurar que el código sea fácil de mantener y extender, utilizando buenas prácticas de desarrollo como la modularización, el uso de tipos estáticos con TypeScript, y la escritura de pruebas automatizadas.

C. Alcance

El alcance de este documento incluye una guía detallada para la configuración del entorno de desarrollo necesario para ejecutar el proyecto, instrucciones paso a

paso para la ejecución del servidor de desarrollo, y una explicación exhaustiva de las decisiones de diseño tomadas durante el desarrollo del proyecto.

1. **Configuración del Entorno:** Se proporcionarán instrucciones detalladas sobre los prerrequisitos necesarios y los pasos a seguir para configurar el entorno de desarrollo. Esto incluye la instalación de Node.js y los gestores de paquetes necesarios, así como la configuración de variables de entorno.
2. **Ejecución del Proyecto:** Se detallarán los comandos necesarios para iniciar el servidor de desarrollo y cómo interactuar con la aplicación en su entorno local.
3. **Decisiones de Diseño:** Se explicarán las decisiones de diseño clave, incluyendo la estructura del proyecto, la elección de tecnologías, y las estrategias de optimización utilizadas. Se discutirá la razón detrás de cada decisión y cómo estas contribuyen a los objetivos del proyecto.
4. **Conclusiones y Trabajo Futuro:** Se presentarán las conclusiones obtenidas durante el desarrollo del proyecto, así como posibles mejoras y extensiones que podrían ser implementadas en el futuro.

La documentación está dirigida a desarrolladores que deseen contribuir al proyecto o utilizarlo como referencia para sus propios desarrollos, así como a cualquier persona interesada en aprender sobre el uso de Next.js, Tailwind CSS, y TypeScript en el desarrollo de aplicaciones web modernas.

D. Importancia del Proyecto

La gestión de tareas es una necesidad común tanto a nivel personal como profesional. Aplicaciones como "Task Manager" proporcionan una herramienta esencial para organizar y priorizar tareas, lo que puede llevar a una mayor productividad y eficiencia. Al utilizar tecnologías modernas y seguir buenas prácticas de desarrollo, este proyecto no solo ofrece una solución práctica para la gestión de tareas, sino que también sirve como un ejemplo de cómo construir aplicaciones web robustas y escalables.

En el contexto de la industria de desarrollo de software, proyectos como "Task Manager" son importantes porque demuestran el uso efectivo de frameworks y herramientas modernas, fomentan el aprendizaje continuo, y promueven la adopción de nuevas tecnologías que pueden mejorar significativamente la calidad del software.

E. Estructura del Documento

Este documento está organizado en varias secciones, cada una enfocada en un aspecto específico del proyecto. Comienza con la introducción y la descripción

general, seguida de instrucciones detalladas para la configuración y ejecución del entorno de desarrollo. Posteriormente, se exploran las decisiones de diseño y las razones detrás de ellas, y se concluye con las conclusiones y posibles mejoras futuras. Finalmente, se proporciona una lista de referencias bibliográficas que respaldan la información presentada y ofrecen recursos adicionales para profundizar en los temas tratados.

II. Configuración del Entorno

A. Prerrequisitos

Antes de comenzar con la configuración del entorno, es esencial asegurarse de tener instalados todos los programas necesarios. Estos prerrequisitos son fundamentales para garantizar que el proyecto se ejecute correctamente. A continuación, se detallan los pasos para la instalación de cada uno:

1. **Node.js**: Node.js es un entorno de ejecución para JavaScript que permite ejecutar código JavaScript en el servidor. Para instalar Node.js, siga estos pasos:
    - Diríjase al sitio oficial de Node.js aquí.
    - Descargue el instalador adecuado para su sistema operativo (Windows, macOS, o Linux).
    - Siga las instrucciones del instalador para completar la instalación.
2. **npm**: npm (Node Package Manager) se instala automáticamente con Node.js. Para verificar la instalación de npm, ejecute el siguiente comando en su terminal:

```
npm -v
```

Esto debería mostrar la versión instalada de npm.

3. **Yarn** (opcional): Yarn es un gestor de paquetes alternativo a npm. Para instalar Yarn, ejecute el siguiente comando:

```
npm install --global yarn
```

Verifique la instalación de Yarn ejecutando:

```
yarn -v
```

4. **pnpm** (opcional): pnpm es otro gestor de paquetes que puede utilizarse en lugar de npm o Yarn. Para instalar pnpm, ejecute:

```
npm install -g pnpm
```

Verifique la instalación de pnpm ejecutando:

pnpm -v

5. **Bun** (opcional): Bun es un nuevo entorno de ejecución para JavaScript que también puede gestionar paquetes. Para instalar Bun, siga las instrucciones en la página oficial de Bun.

B. Instalación del Proyecto

Una vez instalados los prerrequisitos, puede proceder con la instalación del proyecto. Los pasos son los siguientes:

1. **Clonar el Repositorio**: Primero, debe clonar el repositorio del proyecto en su máquina local. Para ello, ejecute el siguiente comando en su terminal, reemplazando <URL_DEL_REPOSITORIO> con la URL real del repositorio:

git clone <URL_DEL_REPOSITORIO>

cd task-manager

2. **Instalar Dependencias**: A continuación, debe instalar todas las dependencias del proyecto. Dependiendo de su gestor de paquetes preferido, puede elegir uno de los siguientes comandos:

npm install

yarn install

pnpm install

bun install

Este comando leerá el archivo package.json y descargará todas las dependencias necesarias para el proyecto.

C. Configuración de Variables de Entorno

Algunos proyectos requieren configuraciones específicas que se definen mediante variables de entorno. Para este proyecto, debe crear un archivo .env.local en la raíz del proyecto. Este archivo contendrá todas las variables de entorno necesarias.

1. **Crear el Archivo .env.local**: En la raíz del proyecto, cree un archivo llamado .env.local.

2. **Agregar Variables de Entorno**: Edite el archivo .env.local y agregue las variables de entorno necesarias. A continuación se muestra un ejemplo:

DATABASE_URL= http://localhost:5000/api

API_KEY=NEXT_PUBLIC_API_URL=http://localhost:5000/api

D. Configuración Adicional

En algunos casos, puede ser necesario realizar configuraciones adicionales específicas del proyecto, como la configuración de linters, formateadores de código, o scripts personalizados. Revise los archivos de configuración incluidos en el proyecto, como .eslintrc.json, tsconfig.json, y tailwind.config.ts para asegurarse de que todo está configurado según sus necesidades.

III. Ejecución del Proyecto

Una vez que haya completado la configuración del entorno, puede proceder a ejecutar el proyecto. Aquí se detallan los pasos necesarios:

A. Iniciar el Servidor de Desarrollo

Para iniciar el servidor de desarrollo, ejecute uno de los siguientes comandos en su terminal, según el gestor de paquetes que esté utilizando:

npm run dev

yarn dev

pnpm dev

bun dev

B. Acceder a la Aplicación

Abra su navegador web y diríjase a http://localhost:3000. Esto le permitirá ver la aplicación en ejecución. Durante el desarrollo, cualquier cambio que realice en el código se reflejará automáticamente en el navegador gracias al hot-reloading de Next.js.

C. Editar y Actualizar Páginas

Puede comenzar a editar las páginas de la aplicación modificando el archivo app/page.tsx. Next.js actualizará automáticamente la página a medida que realice cambios en el código. Esto permite un desarrollo rápido y eficiente, ya que puede ver los resultados de sus cambios casi en tiempo real.

D. Generación de Sitios Estáticos

Next.js permite generar sitios estáticos (SSG) para mejorar el rendimiento y la seguridad de la aplicación. Para generar el sitio estático, ejecute el siguiente comando:

npm run build

yarn build

pnpm build

bun build

Este comando creará una versión optimizada y lista para producción de la aplicación en el directorio out.

E. Despliegue

Para desplegar la aplicación en un entorno de producción, se recomienda utilizar la Plataforma Vercel, desarrollada por los creadores de Next.js. Siga las instrucciones en la documentación de despliegue de Next.js para más detalles.

IV. Decisiones de Diseño

A. Estructura del Proyecto

El diseño de la estructura del proyecto "Task Manager" se realizó con el objetivo de maximizar la modularidad, la mantenibilidad y la escalabilidad. La estructura básica del proyecto es la siguiente:

- **backend**: Este directorio contiene la lógica del servidor y las API. La decisión de separar la lógica del servidor en un directorio independiente facilita el desarrollo y el mantenimiento, permitiendo que los desarrolladores trabajen en el backend sin interferir con el frontend.
- **pages**: Este directorio contiene las páginas del proyecto Next.js. Cada archivo dentro de este directorio representa una ruta en la aplicación. Esta estructura es una convención en Next.js que facilita la navegación y el enrutamiento dentro de la aplicación.
- **public**: Este directorio contiene archivos estáticos que pueden ser servidos directamente al cliente, como imágenes, fuentes y otros recursos estáticos. Al mantener los archivos estáticos en un directorio separado, se mejora la organización y la claridad del proyecto.
- **src**: Este directorio contiene los componentes y la lógica del frontend. Dentro de src, los archivos y directorios están organizados de manera que los componentes reutilizables y la lógica del negocio estén claramente separados y sean fácilmente accesibles.

La elección de esta estructura modular permite que diferentes partes del equipo de desarrollo trabajen simultáneamente en distintas áreas del proyecto sin interferencias, mejorando la eficiencia y facilitando el control de versiones.

B. Uso de Tecnologías

El proyecto "Task Manager" se diseñó utilizando varias tecnologías modernas para aprovechar sus ventajas y mejorar la calidad del software desarrollado. A continuación, se describen las principales tecnologías utilizadas y las razones detrás de su elección:

1. **Next.js**: Next.js es un framework de React que permite la creación de aplicaciones web estáticas y dinámicas. Se eligió Next.js por varias razones:
    - **Renderizado del lado del servidor (SSR)**: Next.js permite el renderizado del lado del servidor, mejorando el rendimiento de la aplicación y la optimización para motores de búsqueda (SEO).
    - **Generación de sitios estáticos (SSG)**: La capacidad de generar sitios estáticos permite que las páginas se carguen más rápidamente y sean más seguras.
    - **Experiencia de desarrollo optimizada**: Next.js proporciona una configuración inicial mínima, permitiendo a los desarrolladores centrarse en el desarrollo de características en lugar de en la configuración del entorno.
2. **Tailwind CSS**: Tailwind CSS es un framework de CSS que utiliza una metodología de diseño utilitario. Las razones para elegir Tailwind CSS incluyen:
    - **Estilos personalizados rápidos**: Tailwind permite la creación de estilos personalizados sin necesidad de escribir CSS adicional.
    - **Consistencia**: Al utilizar utilidades predefinidas, se garantiza una consistencia visual en toda la aplicación.
    - **Productividad**: La creación de interfaces de usuario se acelera al evitar la escritura repetitiva de CSS.
3. **TypeScript**: TypeScript es un superset de JavaScript que añade tipos estáticos. La elección de TypeScript se basó en los siguientes beneficios:
    - **Detección temprana de errores**: Los tipos estáticos permiten detectar errores en tiempo de compilación, reduciendo los errores en tiempo de ejecución.
    - **Mantenibilidad**: TypeScript mejora la mantenibilidad del código al proporcionar una mejor documentación y autocompletado en los editores de código.
    - **Escalabilidad**: TypeScript facilita el desarrollo y mantenimiento de aplicaciones grandes y complejas.

C. Optimización de Fuentes

Este proyecto utiliza next/font para optimizar y cargar automáticamente Inter, una fuente personalizada de Google. Las razones para esta elección son:

1. **Rendimiento**: La optimización de fuentes reduce el tiempo de carga de las páginas, mejorando el rendimiento general de la aplicación.
2. **Accesibilidad**: Al utilizar una fuente personalizada como Inter, se mejora la legibilidad y accesibilidad de la aplicación para todos los usuarios.
3. **Consistencia Visual**: Una fuente personalizada garantiza una experiencia visual coherente en todas las páginas de la aplicación.

D. Arquitectura de la Aplicación

La arquitectura de la aplicación "Task Manager" sigue un enfoque de separación de responsabilidades, asegurando que cada componente y módulo tenga una responsabilidad clara y definida. Esto se logra mediante:

1. **Componentes Reutilizables**: Los componentes de UI están diseñados para ser reutilizables en diferentes partes de la aplicación, lo que reduce la duplicación de código y facilita las actualizaciones.
2. **APIs Modularizadas**: La lógica del servidor y las APIs están organizadas en módulos separados, lo que facilita la escalabilidad y el mantenimiento.
3. **Gestión del Estado**: La gestión del estado de la aplicación se maneja mediante Context API o librerías de gestión de estado como Redux, según las necesidades específicas del proyecto.

V. Recursos Adicionales

A. Documentación Oficial

Para aprender más sobre las tecnologías utilizadas en este proyecto, se recomiendan las siguientes fuentes oficiales:

1. **Next.js**:
    - Documentación de Next.js: Una guía completa sobre las características y API de Next.js.
    - Aprende Next.js: Un tutorial interactivo que cubre los conceptos básicos y avanzados de Next.js.
2. **Tailwind CSS**:
    - Documentación de Tailwind CSS: Una guía completa sobre cómo utilizar Tailwind CSS para diseñar interfaces de usuario.

- Tailwind CSS Cheat Sheet: Una referencia rápida a las clases y utilidades de Tailwind CSS.

3. **TypeScript**:
    - Documentación de TypeScript: Una guía completa sobre el uso de TypeScript, incluyendo tutoriales y ejemplos.
    - TSConfig Reference: Una referencia detallada sobre las opciones de configuración de TypeScript.

B. Artículos y Tutoriales

Además de la documentación oficial, se recomiendan los siguientes artículos y tutoriales para profundizar en el conocimiento de estas tecnologías:

1. **Next.js**:
    - "Getting Started with Next.js" por Vercel: Un tutorial introductorio para comenzar con Next.js.
    - "Building Scalable Applications with Next.js" por Smashing Magazine: Un artículo que explora cómo utilizar Next.js para crear aplicaciones escalables.
2. **Tailwind CSS**:
    - "A Utility-First CSS Framework for Rapid UI Development" por Adam Wathan: Un artículo introductorio sobre el diseño utilitario con Tailwind CSS.
    - "Refactoring UI with Tailwind CSS" por Steve Schoger: Un tutorial sobre cómo mejorar las interfaces de usuario utilizando Tailwind CSS.
3. **TypeScript**:
    - "Understanding TypeScript" por Academind: Un curso completo sobre TypeScript que cubre desde los conceptos básicos hasta los avanzados.
    - "TypeScript for Beginners" por Microsoft: Una serie de tutoriales para principiantes que introducen los conceptos básicos de TypeScript.

C. Libros y Referencias Académicas

Para aquellos interesados en una comprensión más profunda y académica de las tecnologías y patrones utilizados, se recomiendan los siguientes libros y referencias:

1. **Diseño de Software y Patrones**:

    - E. Gamma, R. Helm, R. Johnson, y J. Vlissides, *Design Patterns: Elements of Reusable Object-Oriented Software*. Addison-Wesley, 1994.
    - M. Fowler, *Patterns of Enterprise Application Architecture*. Addison-Wesley, 2002.

2. **Desarrollo de Software**:
    - K. Beck, *Test-Driven Development: By Example*. Addison-Wesley, 2002.
    - R. C. Martin, *Clean Code: A Handbook of Agile Software Craftsmanship*. Prentice Hall, 2008.

3. **Arquitectura de Software**:
    - R. Fielding, "Architectural Styles and the Design of Network-based Software Architectures," Doctoral dissertation, University of California, Irvine, 2000.
    - M. Richards y N. Ford, *Fundamentals of Software Architecture: An Engineering Approach*. O'Reilly Media, 2020.

D. Comunidad y Soporte

Además de los recursos documentales, participar en comunidades y foros puede ser de gran ayuda para resolver dudas y aprender de otros desarrolladores. Algunos recursos recomendados son:

1. **GitHub**: Únete al repositorio de Next.js en GitHub para contribuir al código y obtener soporte de la comunidad.
2. **Stack Overflow**: Busca y pregunta sobre Next.js, Tailwind CSS y TypeScript en Stack Overflow.
3. **Discord**: Únete al servidor de Discord de Next.js para interactuar con otros desarrolladores y obtener soporte en tiempo real.