Tarea#1

Parcial I

Nombre: Mathius Steven Moyano Jara

Tema:

Aplicaciones para web components

Índice

1. Introducción
2. Modelo de negocio
3. Estructura del Proyecto
4. Desarrollo del HTML
5. Estilos con CSS/SASS
6. Funcionalidad con JavaScript
7. Publicación

INTRODUCCION

Introducción a la Programación Orientada a Objetos (OOP) en JavaScript

La Programación Orientada a Objetos (OOP, por sus siglas en inglés) es un paradigma de programación que utiliza "objetos" y sus interacciones para diseñar y programar aplicaciones. JavaScript, aunque inicialmente no fue diseñado con la OOP en mente, ha evolucionado para soportar este paradigma, permitiendo a los desarrolladores construir aplicaciones más robustas y mantenibles.

Fundamentos y Principios de OOP

Encapsulamiento: Es el principio de agrupar datos (propiedades) y métodos (funciones) que operan sobre esos datos dentro de una única unidad llamada objeto. Esto restringe el acceso directo a algunos de los componentes del objeto y puede evitar la modificación no intencionada de datos.

Abstracción: Este principio se refiere a ocultar los detalles complejos y mostrar solo las funcionalidades esenciales del objeto. Permite manejar la complejidad mediante la reducción de información relevante y la implementación de funciones o métodos.

Herencia: Es el mecanismo por el cual una clase puede heredar propiedades y métodos de otra clase. Esto permite la creación de nuevas clases basadas en clases existentes, promoviendo la reutilización del código y la creación de estructuras jerárquicas.

Polimorfismo: Este principio permite que las clases hereden atributos y métodos de otras clases, pero también permite que diferentes clases respondan de manera diferente a la misma interfaz o método. Facilita el uso de una interfaz única para entidades de diferentes tipos.

Implementación de OOP en JavaScript

En JavaScript, los objetos se pueden crear de varias maneras, pero con la introducción de la sintaxis de clase en ECMAScript 6 (ES6), la creación y manipulación de objetos se ha vuelto más sencilla y más alineada con otros lenguajes orientados a objetos como Java o C#.

MODELO DE NEGOCIO

Introducción

En un mundo donde la productividad y la gestión eficiente del tiempo son esenciales, una aplicación de gestión de tareas puede ser una herramienta invaluable tanto para individuos como para equipos. La propuesta aquí presentada es una aplicación web basada en web components que facilita la creación, organización y seguimiento de tareas diarias.

Visión

Ser la aplicación líder en gestión de tareas, ofreciendo una experiencia de usuario excepcional a través de tecnología de vanguardia y funcionalidades adaptables a las necesidades de usuarios individuales y corporativos.

Misión

Proveer una plataforma intuitiva y robusta para la gestión de tareas que permita a los usuarios optimizar su tiempo, aumentar su productividad y alcanzar sus objetivos personales y profesionales.

Público Objetivo

- **Individuos:** Profesionales, estudiantes y cualquier persona que busque mejorar su organización personal.

- **Pequeñas y medianas empresas (PYMES):** Equipos que necesiten una herramienta para gestionar proyectos y tareas de manera colaborativa.

- **Educadores y estudiantes:** Instituciones educativas que busquen una plataforma para asignar y seguir tareas.

Funcionalidades Clave

1. **Creación y Gestión de Tareas:**
   - Añadir, editar y eliminar tareas.
   - Asignar prioridades y fechas de vencimiento.
   - Categorizar tareas por proyectos o etiquetas.

2. **Navegación Intuitiva:**
   - Interfaz de usuario amigable.
   - Barra de navegación clara con secciones bien definidas.

3. **Notificaciones y Recordatorios:**
   - Alertas automáticas para fechas de vencimiento próximas.
   - Recordatorios configurables por el usuario.

4. **Colaboración en Equipo:**
   - Compartir tareas y proyectos con otros usuarios.
   - Comentarios y actualización de estado de las tareas.

5. **Integración con Otras Herramientas:**
   - Sincronización con calendarios (Google Calendar, Outlook).
   - Integraciones con herramientas de gestión de proyectos (Trello, Asana).

6. **Personalización:**
   - Temas y colores personalizables.
   - Configuración de vistas (lista, tablero, calendario).

Modelo de Monetización

1. **Freemium:**
   - **Versión gratuita:** Funcionalidades básicas limitadas.

- **Versión premium:** Suscripción mensual/anual con acceso a todas las funcionalidades avanzadas.

2. **Planes para Empresas:**
   - Tarifas escalonadas basadas en el número de usuarios.
   - Funcionalidades adicionales como reportes avanzados, soporte prioritario y más almacenamiento.

3. **Publicidad:**
   - Anuncios no intrusivos en la versión gratuita.

Estrategia de Marketing

1. **Marketing Digital:**
   - Publicidad en redes sociales (Facebook, LinkedIn, Instagram).
   - Campañas de Google Ads y SEO para mejorar la visibilidad en motores de búsqueda.

2. **Colaboraciones y Alianzas:**
   - Asociaciones con empresas de software complementario.
   - Programas de afiliados y referidos.

3. **Contenido Educativo:**
   - Blog con consejos sobre productividad y gestión de tareas.
   - Tutoriales y webinars para maximizar el uso de la aplicación.

4. **Prueba Gratuita:**
   - Periodo de prueba gratuita para la versión premium para atraer a nuevos usuarios.

Desarrollo y Tecnología

1. **Tecnología Utilizada:**
   - **Frontend:** HTML5, CSS3, JavaScript (Web Components, ES6+).
   - **Backend:** Node.js, Express.js.
   - **Base de Datos:** MongoDB.
   - **Hosting:** Servidores en la nube (AWS, Heroku).

2. **Metodología de Desarrollo:**
   - **Agile/Scrum:** Desarrollo iterativo con sprints de dos semanas.
   - **Pruebas y QA:** Pruebas unitarias, de integración y de usuario final.

Escalabilidad y Futuro

1. **Escalabilidad:**
   - Arquitectura basada en microservicios para facilitar la escalabilidad horizontal.
   - Infraestructura en la nube para manejar picos de demanda.

2. **Futuras Expansiones:**
   - Aplicación móvil nativa para iOS y Android.
   - Funcionalidades de inteligencia artificial para recomendaciones de tareas y análisis de productividad.

Resumen

La aplicación de gestión de tareas propuesta no solo busca mejorar la productividad de individuos y equipos, sino también ofrecer una plataforma robusta, personalizable y escalable que se adapte a las necesidades cambiantes del usuario. Con un modelo de negocio bien definido y una estrategia de marketing efectiva, esta aplicación tiene el potencial de convertirse en un líder en el mercado de gestión de tareas.

Manual Paso a Paso (Ejemplo)

1. Configuración del Entorno de Desarrollo

1. **Instalar Node.js y npm:**

   - Descargar e instalar desde [Node.js](https://nodejs.org/).

2. **Crear Proyecto:**

   ```sh
   mkdir my-web-app
   cd my-web-app
   npm init -y
   ```

3. **Instalar Dependencias:**

   ```sh
   npm install express
   npm install mongodb
   ```

2. Estructura del Proyecto

1. **Carpetas y Archivos:**

   - Crear las carpetas y archivos necesarios siguiendo la estructura mencionada anteriormente.

3. Desarrollo de Web Components

1. **Crear y Registrar Componentes:**

   - Desarrollar cada componente en su propio archivo JS.

2. **Integrar Componentes en HTML:**

- Incluir los componentes en el archivo `index.html`.

3. **Añadir Funcionalidad:**
   - Implementar la lógica en `main.js` para la interacción entre componentes.

4. Publicación

1. **Preparar Archivos para Producción:**
   - Minimizar y empaquetar los archivos.

2. **Subir al Servidor de Hosting:**
   - Utilizar herramientas como FTP o Git para subir los archivos al servidor.

5. Documentación y Video

1. **Crear Manual en PDF:**
   - Documentar cada paso del proceso y generar un PDF.

2. **Grabar Video:**
   - Grabar la pantalla mientras se explica el desarrollo paso a paso y editar el video.

6. Entrega

1. **Empaquetar Archivos:**
   - Comprimir todos los archivos en un archivo .rar.

2. **Subir al Aula Virtual:**
   - Subir el archivo comprimido al aula virtual según las instrucciones del Tutor.

ESTRUCTURA DEL PROYECTO

DESAROLLO DEL HTML

El archivo index.html contiene la estructura básica de tu aplicación:

Index.html

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Gestión de Tareas</title>
  <link rel="stylesheet" href="css/styles.css">
</head>
<body>
  <app-header></app-header>
  <create-task></create-task>
  <task-list></task-list>

  <script src="js/components/app-header.js"></script>
  <script src="js/components/create-task.js"></script>
  <script src="js/components/task-list.js"></script>
  <script src="js/main.js"></script>
</body>
</html>
```

Estilos con scss/css

El archivo main.scss define los estilos para tu aplicación. Compílalo a main.css si utilizas SASS.

Main.css

```
/* styles/styles.css */
```

```css
body {
    font-family: Arial, sans-serif;
    margin: 0;
    padding: 0;
    display: flex;
    flex-direction: column;
    align-items: center;
    background-color: #f4f4f4;
    color: #333;
}

header {
    background-color: #007BFF;
    width: 100%;
    padding: 10px 0;
    text-align: center;
    color: #fff;
}

h1 {
    margin: 20px 0;
}

nav {
    margin: 20px 0;
}

nav ul {
    list-style-type: none;
    padding: 0;
    display: flex;
    gap: 20px;
}

nav ul li {
    margin: 0;
}

nav ul li a {
    text-decoration: none;
    color: #007BFF;
    font-size: 18px;
    border: 2px solid #007BFF;
    padding: 5px 10px;
    border-radius: 5px;
    transition: background-color 0.3s, color 0.3s;
```

```css
}

nav ul li a:hover {
    background-color: #007BFF;
    color: #fff;
}

main {
    width: 100%;
    max-width: 800px;
    margin: 20px;
    padding: 20px;
    background-color: #fff;
    border: 1px solid #ddd;
    border-radius: 5px;
    box-shadow: 0 0 10px rgba(0, 0, 0, 0.1);
}

form {
    display: flex;
    flex-direction: column;
}

label {
    margin-bottom: 5px;
    font-weight: bold;
}

input[type="text"], input[type="number"] {
    width: 100%;
    padding: 10px;
    margin-bottom: 15px;
    border: 1px solid #ddd;
    border-radius: 5px;
}

button {
    background-color: #007BFF;
    color: #fff;
    border: none;
    padding: 10px 15px;
    border-radius: 5px;
    cursor: pointer;
    transition: background-color 0.3s;
}
```

```css
button:hover {
    background-color: #0056b3;
}

h2 {
    margin-top: 20px;
}

p {
    background-color: #f9f9f9;
    padding: 10px;
    border: 1px solid #ddd;
    border-radius: 5px;
    box-shadow: 0 0 10px rgba(0, 0, 0, 0.1);
    margin-top: 15px;
    white-space: pre-line;
}
```

Funcionalidad con JavaScript.

El archivo main.js maneja la interacción con el DOM:

Main.js

```javascript
   document.addEventListener('DOMContentLoaded', () => {
    const taskList = document.querySelector('task-list');
    const createTask = document.querySelector('create-task');

    createTask.addEventListener('taskAdded', (event) => {
      taskList.addTask(event.detail);
    });
  });
```

App-header.js

```
class AppHeader extends HTMLElement{
 constructor(){
  super();
  this.attachShadow({ mode: 'open' });
  this.shadowRoot.innerHTML = `
```

```
   <style>
    header {
     background-color: #4CAF50;
     padding: 10px;
     text-align: center;
     color: white;
     font-size: 20px;
    }
   </style>
   <header>
    <h1>Gestión de Tareas</h1>
   </header>
  `;
 }
}

customElements.define('app-header', AppHeader);
```

Task-list.js

```
class TaskList extends HTMLElement {
  constructor() {
    super();
    this.attachShadow({ mode: 'open' });
    this.tasks = [];
    this.shadowRoot.innerHTML = `
      <style>
        ul {
          list-style: none;
          padding: 0;
        }
        li {
          background: #f9f9f9;
          margin: 5px 0;
          padding: 10px;
          border: 1px solid #ddd;
        }
      </style>
      <ul></ul>
    `;
  }

  connectedCallback() {
```

```
    this.render();
  }

  addTask(task) {
    this.tasks.push(task);
    this.render();
  }

  render() {
    const ul = this.shadowRoot.querySelector('ul');
    ul.innerHTML = this.tasks.map(task => `<li>${task}</li>`).join('');
  }
}

customElements.define('task-list', TaskList);
```

Create-task.js

```
class CreateTask extends HTMLElement {
  constructor() {
    super();
    this.attachShadow({ mode: 'open' });
    this.shadowRoot.innerHTML = `
      <style>
        form {
          display: flex;
          flex-direction: column;
        }
        input, button {
          margin: 5px 0;
          padding: 10px;
        }
      </style>
      <form>
        <input type="text" placeholder="Nueva tarea">
        <button type="button">Agregar Tarea</button>
      </form>
    `;
  }

  connectedCallback() {
    this.shadowRoot.querySelector('button').addEventListener('click', () =>
this.addTask());
  }
```

```
  addTask() {
    const input = this.shadowRoot.querySelector('input');
    const event = new CustomEvent('taskAdded', {
      detail: input.value,
      bubbles: true,
      composed: true
    });
    this.dispatchEvent(event);
    input.value = '';
  }
}

customElements.define('create-task', CreateTask);
```

Publicación

Pasos para Subir Archivos por FTP en Total Commander

1. Descargar e Instalar Total Commander

Descargar: Ve al sitio web oficial de Total Commander (https://www.ghisler.com/) y descarga la versión correspondiente a tu sistema operativo.

Instalar: Ejecuta el instalador y sigue las instrucciones para completar la instalación.

2. Configurar la Conexión FTP

Abrir Total Commander: Inicia Total Commander desde el menú de inicio o el acceso directo en el escritorio.

Acceder al Cliente FTP: En la barra de menú superior, selecciona Red y luego Conectar al servidor FTP... (o presiona Ctrl + F).

3. Crear una Nueva Conexión FTP

Agregar Nueva Conexión: En la ventana de conexiones FTP, haz clic en el botón Nueva conexión.

Configurar Parámetros de Conexión: Completa los campos necesarios:

Sesión: Escribe un nombre descriptivo para la conexión (por ejemplo, "Repositorio FTP ESPE").

Servidor [Puerto]: Introduce la dirección del servidor FTP proporcionada por el tutor.

Nombre de usuario: Introduce tu nombre de usuario FTP.

Contraseña: Introduce tu contraseña FTP.

Opcionalmente, puedes marcar la casilla Guardar contraseña para evitar tener que ingresarla cada vez.

Guardar la Conexión: Haz clic en Aceptar para guardar la configuración.

4. Conectar al Servidor FTP

Seleccionar la Conexión: En la ventana de conexiones FTP, selecciona la conexión que acabas de crear.

Conectar: Haz clic en Conectar. Si la conexión es exitosa, verás los archivos y carpetas del servidor FTP en una de las dos paneles de Total Commander.

5. Subir Archivos al Servidor FTP

Navegar en el Panel Local: En el panel izquierdo (o derecho, dependiendo de tu configuración), navega hasta la carpeta de tu proyecto local.

Seleccionar Archivos: Selecciona los archivos y carpetas que deseas subir. Puedes usar el ratón para seleccionar varios archivos o usar la tecla Ctrl para seleccionar múltiples elementos.

Copiar Archivos al Servidor: Con los archivos seleccionados, presiona F5 o haz clic en el botón Copiar en la barra inferior. Asegúrate de que el panel derecho (o izquierdo) esté en la carpeta del servidor donde deseas subir los archivos.

Confirmar la Transferencia: Aparecerá una ventana de confirmación. Revisa que todo esté correcto y haz clic en Aceptar para iniciar la transferencia.

6. Verificar los Archivos en el Servidor

Verificar la Subida: Una vez completada la transferencia, verifica que todos los archivos se hayan subido correctamente revisando el panel del servidor FTP en Total Commander.

Comprobar en el Navegador: Abre tu navegador web y navega a la URL del repositorio FTP proporcionada por el tutor para confirmar que los archivos están accesibles en línea.

7. Desconectar del Servidor FTP

Cerrar Conexión: En Total Commander, selecciona Red y luego Desconectar del servidor FTP para cerrar la conexión una vez que hayas terminado.