Trabajo Práctico Integrador

Como se estableció previamente, en el régimen de promocionalidad de la materia se establece la realización de un Proyecto Integrador, el cual representa un 70% de la nota final. Este proyecto consistirá en desarrollar una aplicación web utilizando **ReactJS** y una API RESTful para consumir y manipular datos.

API

La cátedra proporcionará una **API privada** que los alumnos utilizarán para desarrollar la aplicación web. La API ofrece varias opciones de proyectos sobre los cuales los alumnos pueden elegir trabajar. Cada alumno o grupo de alumnos tendrá acceso a la API mediante JSON Web Tokens (**JWT**), garantizando un acceso seguro y privado.

No obstante, los alumnos también pueden optar por utilizar una API propia, siempre y cuando cumpla con ciertos requisitos mínimos. En este caso, los alumnos deberán solicitar la aprobación de la cátedra antes de comenzar a trabajar en el proyecto.

Al finalizar el proyecto, **los alumnos deberán realizar una entrega y defensa** del mismo, demostrando los conocimientos adquiridos durante el curso.

Metodología

Los alumnos deberán trabajar en grupos de 1 a 3 personas, siempre buscando una distribución equitativa de tareas y responsabilidades. Además, tendrán la libertad de elegir el tema del proyecto de una lista proporcionada por la cátedra.

Una vez concluidas las clases teóricas y prácticas, los alumnos tendrán un período de tiempo para trabajar en el proyecto. El cual se estipula en el cronograma tentativo de la materia. Durante este tiempo, los alumnos deberán trabajar en el desarrollo de la aplicación y preparar una presentación para la defensa del proyecto.

Temas del Proyecto

- Aplicación de Música
- Aplicación de Recetas
- Aplicación de Noticias
- Aplicación de Tareas
- Aplicación de Mensajería
- Aplicación de Viajes
- Aplicación de Deportes

- Aplicación de Juegos

Requerimientos del Proyecto

El proyecto deberá cumplir con los siguientes requerimientos mínimos:

0. General

- Utilizar **ReactJS** para el desarrollo de la aplicación web.
- Consumir una API RESTful para obtener y manipular datos. La API puede ser proporcionada por la cátedra o ser propia/pública, siempre y cuando cumpla con ciertos requisitos mínimos. Si la API es propia, esta debe implementar autenticación mediante alguno de los mecanismos vistos en clase (por ejemplo, JWT, OAuth, Basic Auth, Session Auth, etc.).
- Utilizar **Git** para el control de versiones del código fuente. Cada integrante del grupo deberá realizar aportes al proyecto y mantener un historial de cambios en un repositorio de **GitHub**, **GitLab** o **Bitbucket**.
- El proyecto debe ser desarrollado en un entorno de desarrollo local y desplegado en un servidor de producción. Se recomienda utilizar **Vercel**, **Netlify**, **Glitch** o cualquier otro servicio de hosting para desplegar la aplicación (recomendamos que sea un servicio gratuito).

1. Autenticación y Autorización

- Implementar autenticación y manejo de sesiones utilizando JWT proporcionados por la API, o cualquier otro mecanismo de autenticación que la API requiera.
- Asegurar que todas las solicitudes a la API estén autenticadas.

2. UI e Implementación de Componentes

- El desarrollo de la interfaz de usuario **debe ser sobre todo funcional** y fácil de usar. Se valorará la creatividad y originalidad en el diseño de la interfaz.
- Utilizar **componentes** de React para manejar eficientemente el estado y el ciclo de vida de la aplicación.
- Debe emplearse **al menos un contexto** para compartir información entre componentes de la aplicación, como el estado de autenticación del usuario, temas, etc.
- Se recomienda utilizar los **hooks** de React vistos en clases para manejar el estado y el ciclo de vida de los componentes, como `useState`, `useEffect`, `useContext`, etc. Pero se valora el uso de hooks personalizados y el manejo de estados complejos. Pueden emplearse librerías de gestión de estados

como **Redux**, **MobX**, **Context API**, etc. si se considera necesario, o si tienen experiencia previa con ellas.

- - No es necesario utilizar un framework de estilos como Bootstrap, Bulma o Tailwind CSS, pero su uso es recomendado para acelerar el desarrollo de la interfaz de usuario. Sin embargo, los alumnos pueden optar por utilizar **CSS puro** si lo prefieren.

3. Consumo de la API

- Consumir múltiples endpoints de la API proporcionada para realizar operaciones **CRUD** (*Create*, *Read*, *Update*, *Delete*). Por ejemplo, si el tema seleccionado es la aplicación de música, los alumnos podrían consumir los endpoints de `HarmonyHub` para obtener información sobre las canciones, artistas, álbumes, géneros, etc.
- Manejar errores de API de forma adecuada y mostrar mensajes de error informativos al usuario mediante componentes de React que representen notificaciones o alertas. Por ejemplo, si una petición `POST` a la API falla, mostrará una alerta con el mensaje de error correspondiente.

4. Enrutamiento

Utilizar un mecanismo de enrutamiento para navegar entre las diferentes secciones de la aplicación, como **React Router**. Como mínimo, se deben implementar las siguientes rutas:

- `/` - **Página de inicio de la aplicación**. Dependiendo del tema seleccionado, esta página podría mostrar una lista de elementos (por ejemplo, canciones, películas, recetas, etc.).
- `/login` - **Página de inicio de sesión**. Esta página debe contener un formulario para que el usuario pueda ingresar su nombre de usuario y contraseña.
- `/profile` - **Página de perfil de usuario**. Esta página debe mostrar información sobre el usuario autenticado, como su nombre, correo electrónico, etc.
- `/<resource>/` - **Página de detalle de un recurso específico**. Por ejemplo, si el tema seleccionado es la aplicación de música y el recurso es una **canción**, la ruta podría ser `/songs/` y mostrar información sobre cada canción.
  La creación de nuevas canciones, la edición y eliminación de canciones también se pueden realizar en esta página mediante formularios emergentes o modales.

Sin embargo, si se prefiere, se pueden implementar rutas adicionales para estas operaciones, como `/songs/new`, `/songs/edit/:id`, `/songs/delete/:id`, etc. Dejamos a criterio de los alumnos la implementación de estas rutas adicionales.

Además, debe incluirse **al menos una ruta protegida** que requiera autenticación para acceder a ella. Por ejemplo, la página de perfil de usuario solo debe ser accesible para usuarios autenticados. Por el contrario, la página principal de la aplicación debe ser accesible para todos los usuarios, incluso si no están autenticados.

Implementar un componente de navegación que muestre enlaces a las diferentes rutas de la aplicación. Este componente debe actualizarse automáticamente según el estado de autenticación del usuario.

Por último, se debe implementar al menos un componente para manejar errores, como una **página 404** para rutas no encontradas.

5. Documentación

- Documentar el código fuente de manera clara y concisa.
- Preparar una presentación que resuma el proyecto, la arquitectura utilizada y las decisiones de diseño tomadas.

Evaluación

La evaluación del proyecto se basará en los siguientes criterios:

- Funcionalidad y cumplimiento de los requerimientos.
- Calidad del código y uso de buenas prácticas de desarrollo.
- Estabilidad y manejo de errores.
- Diseño e interacción del usuario.

Entrega Final

La entrega final del proyecto deberá incluir el código fuente completo, la documentación asociada (si la hubiera), y una presentación del proyecto. La defensa del proyecto se realizará ante el profesor correspondiente a cada comisión.

La fecha límite para la entrega del proyecto será el **domingo 11 de agosto a las 23:59 hs**. Los alumnos deberán enviar un enlace al repositorio de **GitHub**, **GitLab** o

Bitbucket donde se encuentre el código fuente del proyecto, así como un enlace a la aplicación desplegada en un servidor de producción.

A partir del **lunes 12 de agosto** y hasta el **viernes 16 de agosto**, los alumnos iniciarán la defensa del proyecto, donde deberán presentar y explicar el trabajo realizado vía Zoom. La defensa del proyecto tendrá una duración máxima de **20 minutos**. Durante la defensa, los alumnos deberán responder preguntas y demostrar el funcionamiento de la aplicación.

El orden de presentación de los proyectos se establecerá previamente y se comunicará a los alumnos con anticipación. Si así lo desean, los alumnos pueden solicitar voluntariamente ser los primeros en presentar su proyecto para evitar esperas innecesarias.

Si un proyecto no cumple con los requerimientos mínimos establecidos, la cátedra podrá solicitar a los alumnos que realicen correcciones y mejoras para una nueva entrega.

Recuperación

En caso de solicitarse una nueva entrega, los alumnos deberán realizar las correcciones y mejoras necesarias para defender el proyecto nuevamente el **lunes 19 de agosto**. En caso de no cumplir con los requerimientos mínimos en la segunda entrega, el proyecto será considerado **desaprobado**.

Recomendaciones Finales

- Planificar y organizar el trabajo en equipo. El uso de metodologías ágiles como **Scrum** o **Kanban** puede ser de gran ayuda, y pueden utilizarse herramientas como **Trello**, **Notion** o **Jira** para gestionar las tareas.
- Crear un repositorio lo antes posible y establecer una estructura de proyecto clara y concisa. Se recomienda utilizar un flujo de trabajo basado en **Git Flow** para organizar las ramas y los cambios en el código fuente.
- Durante las clases de consulta y tutorías, los alumnos pueden solicitar ayuda y asesoramiento a los profesores para resolver dudas y problemas relacionados con el proyecto.
- En cuanto a la implementación de la API, se recomienda utilizar herramientas como **Postman** o **Insomnia** para probar los endpoints y asegurarse de que la comunicación con la API sea correcta.

- Recomendamos emplear los **hooks** analizados en clase para manejar el estado y el ciclo de vida de los componentes de React, así como el uso de componentes funcionales en lugar de componentes de clase.
- A lo largo de las clases impartidas, se ha utilizado en repetidas ocasiones la función `fetch` para realizar solicitudes HTTP a la API, pero los alumnos pueden optar por utilizar librerías como **Axios**, **Superagent**, **XMLHttpRequest**, etc. para simplificar el manejo de las solicitudes HTTP.
- Sobre todo, mantener una comunicación fluida y constante entre los miembros del equipo y con los profesores en caso de dudas o problemas. La comunicación es clave para el éxito del proyecto.