Estructura del Proyecto app alquiler de libros Frontend con Vue y Pinia

Tecnologías Utilizadas

El frontend de la aplicación se desarrollará con las siguientes tecnologías:

- **Vue 3**: Framework progresivo para la construcción de interfaces de usuario.
- **Pinia**: Librería de gestión de estado recomendada para Vue 3.
- **Vue Router**: Manejo de rutas y navegación dentro de la aplicación.
- **Axios**: Cliente HTTP para la comunicación con la API backend en Django.
- **Bootstrap** (Opcional): Para la esterilización rápida y moderna de los componentes.

Estructura del Proyecto

El proyecto se organizará de la siguiente manera:

```
/
├── src/
│   ├── api/            # Módulos para interactuar con la API backend
│   ├── assets/         # Recursos estáticos como imágenes, fuentes, etc.
│   ├── components/         # Componentes reutilizables de Vue
│   ├── composables/        # Funciones reutilizables (composition API)
│   │   ├── useAuth.ts      # Composable para autenticación
│   ├── router/         # Configuración del enrutador Vue Router
│   │   ├── index.ts
│   ├── stores/         # Stores de Pinia para la gestión del estado
│   ├── views/          # Vistas principales de la aplicación
│   ├── App.vue         # Componente raíz
│   ├── main.ts         # Punto de entrada principal de la aplicación
├── public/             # Archivos públicos estáticos
├── tests/              # Pruebas del proyecto
├── package.json            # Dependencias y scripts del proyecto
├── README.md               # Documentación del proyecto
├── tsconfig.json       # Configuración de TypeScript
├── vite.config.ts      # Configuración de Vite
```

Descripción de los Módulos

1. `api/` - Módulos de Interacción con la API

Cada archivo en esta carpeta representa un conjunto de métodos para realizar peticiones HTTP al backend en Django.

`axiosConfig.ts` Configura Axios para incluir el token Bearer del usuario en los headers.

2. `stores/` - Gestión del Estado con Pinia

Cada módulo en esta carpeta representa un store de Pinia que almacena y maneja el estado de la aplicación.

Cada módulo en esta carpeta representa un store de Pinia que gestiona una parte del estado de la aplicación.

`authStore.ts` Maneja la autenticación del usuario, incluyendo almacenamiento y limpieza del token.

`booksStore.ts` Gestiona la lista de libros, carga, edición y manejo de errores.

`profileStore.ts` Maneja los datos del perfil del usuario y sus actualizaciones.

`rentalStore.ts` Controla las rentas activas, historial de alquileres y devoluciones.

3. `views/` - Páginas Principales

Cada vista corresponde a una ruta en la aplicación y representa una página completa.

`BooksView.vue` Muestra la lista de libros disponibles para los usuarios.

`BookDetailView.vue` Vista detallada de un libro individual.

`AddBookView.vue` Formulario para agregar un nuevo libro.

`UpdateBookView.vue` Formulario para actualizar los datos de un libro.

`rentals.vue` Muestra las rentas del usuario y permite nuevas rentas o devoluciones.

`ProfileView.vue` Vista con los datos del perfil del usuario.

`UpdateProfileView.vue` Formulario para actualizar la información del perfil.

`LoginView.vue` Formulario de inicio de sesión.

`RegisterView.vue` Formulario de registro de nuevo usuario.

`WelcomeView.vue` Vista de bienvenida al iniciar sesión.

`AboutView.vue` Información general sobre la aplicación.

`HomeView.vue` Pantalla principal inicial.

4. `components/` - Componentes Reutilizables

Esta carpeta contiene componentes reutilizables que se utilizan dentro de las vistas.

`Layout.vue` Componente contenedor que define la estructura de la página.

`LoginForm.vue` Formulario de login utilizado por LoginView.vue.

`icons/` Carpeta que contiene íconos reutilizables en formato SVG.

`__tests__/` Contiene pruebas unitarias de componentes.

5. `router/` - Configuración de Vue Router

El archivo `index.ts` maneja la configuración de las rutas y la navegación dentro de la aplicación.

6. `main.ts` - Punto de Entrada de la Aplicación

Este archivo inicializa Vue, Pinia, Bootstrap y Vue Router, y monta la aplicación en el DOM.

7. `composables/`- Funciones Reutilizables (Composition API)

- `useAuth.ts` Función reutilizable para manejar autenticación como login y logout.

Flujo de Trabajo

1. **Autenticación:**
    - El usuario se registra o inicia sesión a través de `api/users.ts`.
    - Se almacena el token en el estado de Pinia (`users.ts`).
    - Se utiliza el token en las peticiones protegidas (como rentar un libro).
2. **Gestión de Libros:**
    - Se obtiene la lista de libros desde `api/books.ts`.
    - Se almacena en el store `books.ts` y se muestra en `BooksView.vue`.
3. **Gestión de Alquileres:**
    - Un usuario puede alquilar un libro enviando una petición a `api/rentals.ts`.

- La información de las rentas activas se almacena en `rentals.ts`.

- Se muestran en `RentalsView.vue`.
- Además también podrá devolver las rentas de libros que haya hecho.