Sistema de Gestión de Aplicaciones Frontend

Descripción

Este es un sistema completo de gestión de aplicaciones desarrollado con React y Vite. El sistema implementa una arquitectura moderna y escalable, con un enfoque en la experiencia del usuario y el rendimiento. La aplicación está diseñada para manejar múltiples roles de usuario, incluyendo administradores, usuarios regulares y solicitantes.

Tecnologías Principales

- React 18.3.1 - Framework principal para la construcción de la interfaz
- Vite 6.0.5 - Bundler y servidor de desarrollo de alta velocidad
- React Router 7.1.1 - Sistema de enrutamiento avanzado
- Chart.js 4.4.7 - Visualización de datos y gráficos interactivos
- Axios 1.7.9 - Cliente HTTP para comunicación con APIs
- SASS 1.83.1 - Preprocesador CSS para estilos modulares
- React Icons 5.4.0 - Biblioteca de iconos vectoriales
- TSParticles 2.12.2 - Efectos visuales y animaciones
- Microsoft Clarity 1.0.0 - Análisis de comportamiento de usuarios

Arquitectura del Proyecto

Estructura de Directorios

```
src/
├── assets/          # Recursos estáticos
│   ├── images/      # Imágenes y gráficos
│   ├── fonts/       # Fuentes tipográficas
│   ├── icons/       # Iconos personalizados
│   └── [otros]/     # Componentes específicos
│
├── components/      # Componentes reutilizables
│   ├── Toast/       # Sistema de notificaciones
│   ├── ThemeToggle/# Control de tema claro/oscuro
│   ├── Loader/      # Indicadores de carga
│   ├── Layout/      # Componentes de estructura
│   ├── Header/      # Navegación principal
│   └── [otros]/     # Componentes específicos
│
├── context/        # Gestión de estado global
│   ├── AuthContext    # Estado de autenticación
│   ├── ThemeContext   # Estado del tema
│   └── AppContext     # Estado general de la aplicación
│
├── pages/          # Páginas principales
│   ├── admin/      # Panel de administración
│   │   ├── Users/          # Gestión de usuarios
│   │   │   ├── List.jsx    # Listado de usuarios
│   │   │   ├── Create.jsx # Creación de usuarios
│   │   │   └── Edit.jsx    # Edición de usuarios
│   │   ├── Reviewers/      # Gestión de revisores
│   │   │   ├── List.jsx    # Listado de revisores
│   │   │   └── Assign.jsx # Asignación de revisores
│   │   ├── Applications/  # Gestión de aplicaciones
│   │   │   ├── List.jsx    # Listado de aplicaciones
│   │   │   └── View.jsx    # Vista detallada
│   │   ├── Applicants/    # Gestión de solicitantes
│   │   │   ├── List.jsx    # Listado de solicitantes
│   │   │   └── Details.jsx# Detalles del solicitante
│   │   ├── Emails/         # Sistema de correos
│   │   │   ├── Templates/ # Plantillas de correo
│   │   │   └── History/    # Historial de envíos
│   │   ├── dashboard/      # Dashboard administrativo
│   │   │   ├── Stats.jsx   # Estadísticas
│   │   │   └── Charts.jsx # Gráficos
│   │   └── login/          # Login administrativo
│   │
```

```
|   ├── auth/      # Autenticación y registro
|   |   ├── Login.jsx      # Página de inicio de sesión
|   |   ├── Register.jsx   # Página de registro
|   |   ├── Login.module.scss
|   |   └── Register.module.scss
|   |
|   ├── users/     # Área de usuarios
|   |   ├── dashboard/     # Dashboard de usuario
|   |   |   ├── Overview.jsx    # Vista general
|   |   |   └── Activities.jsx  # Actividades recientes
|   |   ├── DiagnosticExam/     # Examen diagnóstico
|   |   |   ├── Questions.jsx   # Preguntas
|   |   |   └── Results.jsx     # Resultados
|   |   ├── ApplicationClosed/  # Aplicaciones cerradas
|   |   |   ├── List.jsx        # Listado
|   |   |   └── Details.jsx     # Detalles
|   |   └── login/              # Login de usuario
|   |
|   ├── home/      # Página principal
|   |   ├── Home.jsx           # Componente principal
|   |   ├── Sections/          # Secciones de la página
|   |   |   ├── Hero.jsx       # Sección hero
|   |   |   ├── Features.jsx   # Características
|   |   |   └── Contact.jsx    # Contacto
|   |   └── home.module.scss   # Estilos
|   |
|   ├── home2/     # Versión alternativa de la página principal
|   |   ├── Home2.jsx          # Componente principal
|   |   ├── Components/        # Componentes específicos
|   |   └── home2.module.scss  # Estilos
|   |
|   ├── contact/   # Página de contacto (inglés)
|   |   ├── Contact.jsx        # Componente principal
|   |   ├── Form.jsx           # Formulario
|   |   └── contact.module.scss# Estilos
|   |
|   └── contacto/  # Página de contacto (español)
|       ├── Contacto.jsx       # Componente principal
|       ├── Formulario.jsx     # Formulario
|       └── contacto.module.scss# Estilos
|
├── interceptors/  # Interceptores de Axios
|   ├── axiosConfig.js    # Configuración base de interceptores
|   |   ├── requestInterceptor  # Interceptor de peticiones
```

```
│   │   │   ├── Token Injection    # Inyección de tokens
│   │   │   ├── Headers Setup      # Configuración de headers
│   │   │   └── Error Prevention   # Prevención de errores
│   │   │
│   │   └── responseInterceptor # Interceptor de respuestas
│   │       ├── Error Handling     # Manejo de errores
│   │       ├── Token Refresh      # Renovación de tokens
│   │       └── Response Format    # Formateo de respuestas
│   │
│   └── axiosAdmin.js     # Interceptores específicos para admin
│       ├── adminRequestInterceptor  # Interceptor de peticiones admin
│       │   ├── Admin Token Validation # Validación de tokens admin
│       │   ├── Role Verification     # Verificación de roles
│       │   └── Admin Headers         # Headers específicos admin
│       │
│       └── adminResponseInterceptor # Interceptor de respuestas admin
│           ├── Admin Error Handling  # Manejo de errores admin
│           ├── Admin Logging         # Registro de actividades
│           └── Admin Response Format # Formateo de respuestas admin
│
├── services/      # Servicios y APIs
│   ├── api/       # Configuración de API
│   │   ├── axios.config.js    # Configuración base de Axios
│   │   ├── interceptors.js    # Interceptores de peticiones
│   │   └── endpoints.js       # Definición de endpoints
│   │
│   ├── auth.service.js        # Servicio de autenticación
│   │   ├── login()           # Inicio de sesión
│   │   ├── register()        # Registro de usuarios
│   │   ├── logout()          # Cierre de sesión
│   │   └── refreshToken()    # Renovación de token
│   │
│   ├── usersauth.service.js   # Servicio de autenticación de usuarios
│   │   ├── userLogin()       # Login específico de usuarios
│   │   └── userLogout()      # Logout de usuarios
│   │
│   ├── users.service.js       # Servicio de gestión de usuarios
│   │   ├── getUsers()        # Obtener lista de usuarios
│   │   ├── createUser()      # Crear nuevo usuario
│   │   ├── updateUser()      # Actualizar usuario
│   │   └── deleteUser()      # Eliminar usuario
│   │
│   ├── application.service.js # Servicio principal de aplicaciones
│   │   ├── createApplication()# Crear aplicación
```

```
│   │   ├── getApplications() # Obtener aplicaciones
│   │   ├── updateStatus()    # Actualizar estado
│   │   └── getDetails()      # Obtener detalles
│   │
│   ├── applications.service.js# Servicio de gestión múltiple
│   │   ├── bulkUpdate()      # Actualización masiva
│   │   ├── filterByStatus()  # Filtrado por estado
│   │   └── getStatistics()   # Estadísticas
│   │
│   ├── applicants.service.js  # Servicio de solicitantes
│   │   ├── getApplicants()   # Listar solicitantes
│   │   ├── getApplicantDetails()# Detalles del solicitante
│   │   └── updateApplicant() # Actualizar información
│   │
│   ├── reviewers.service.js   # Servicio de revisores
│   │   ├── getReviewers()    # Listar revisores
│   │   ├── assignReviewer()  # Asignar revisor
│   │   └── getWorkload()     # Carga de trabajo
│   │
│   ├── diagnostic.service.js  # Servicio de exámenes
│   │   ├── getQuestions()    # Obtener preguntas
│   │   ├── submitAnswers()   # Enviar respuestas
│   │   └── getResults()      # Obtener resultados
│   │
│   ├── emails.service.js      # Servicio de correos
│   │   ├── sendEmail()       # Enviar correo
│   │   ├── getTemplates()    # Obtener plantillas
│   │   └── getHistory()      # Historial de envíos
│   │
│   └── statistics.service.js  # Servicio de estadísticas
│       ├── getDashboardStats()# Estadísticas del dashboard
│       ├── getApplicationStats()# Estadísticas de aplicaciones
│       └── getUserStats()     # Estadísticas de usuarios
│
└── utils/         # Utilidades y helpers
    ├── validators/    # Validaciones
    ├── formatters/    # Formateadores
    └── constants/     # Constantes globales
```

Características Detalladas

Sistema de Autenticación

- Login y registro de usuarios
- Recuperación de contraseña
- Validación de tokens JWT
- Protección de rutas
- Gestión de sesiones

Panel de Administración

- Dashboard con métricas en tiempo real
- Gestión de usuarios y permisos
- Configuración del sistema
- Logs de actividad
- Gestión de contenido

Área de Usuarios

- Perfil personalizable
- Gestión de aplicaciones
- Historial de actividades
- Notificaciones en tiempo real
- Sistema de mensajería

Características Técnicas

- Diseño responsive con breakpoints optimizados
- Lazy loading de componentes
- Code splitting por rutas
- Caché inteligente de datos
- Optimización de rendimiento
- SEO friendly
- Accesibilidad WCAG 2.1

Sistema de Interceptores

Los interceptores son una parte crucial del sistema que maneja todas las comunicaciones HTTP. Proporcionan una capa de middleware para las peticiones y respuestas, permitiendo:

Interceptores Base (`axiosConfig.js`)

- **Interceptores de Petición**
    - Inyección automática de tokens de autenticación
    - Configuración de headers comunes
    - Validación de datos antes del envío
    - Prevención de peticiones duplicadas

- Manejo de timeouts

- **Interceptores de Respuesta**

  - Manejo centralizado de errores
  - Renovación automática de tokens
  - Formateo consistente de respuestas
  - Redirección en caso de errores de autenticación
  - Logging de respuestas

Interceptores de Administración (`axiosAdmin.js`)

- **Interceptores de Petición Admin**

  - Validación de tokens de administrador
  - Verificación de roles y permisos
  - Headers específicos para operaciones administrativas
  - Validación de datos administrativos
  - Prevención de accesos no autorizados

- **Interceptores de Respuesta Admin**

  - Manejo específico de errores administrativos
  - Registro detallado de actividades administrativas
  - Formateo especial para respuestas administrativas
  - Notificaciones de seguridad
  - Auditoría de operaciones

Beneficios del Sistema de Interceptores

1. **Seguridad Mejorada**

   - Validación consistente de tokens
   - Prevención de accesos no autorizados
   - Manejo seguro de credenciales

2. **Mantenibilidad**

   - Código centralizado para manejo de peticiones
   - Fácil actualización de lógica común
   - Reducción de código duplicado

3. **Experiencia de Usuario**

   - Manejo consistente de errores
   - Feedback inmediato de problemas
   - Redirecciones automáticas cuando es necesario

4. **Monitoreo y Debugging**
    - Logging centralizado de peticiones
    - Trazabilidad de errores
    - Métricas de rendimiento
5. **Optimización**
    - Caché de peticiones
    - Prevención de peticiones duplicadas
    - Manejo eficiente de tokens

Guía de Desarrollo

Requisitos Previos

- Node.js 18.x o superior
- npm 9.x o superior
- Git
- Editor de código (VS Code recomendado)

Configuración del Entorno

1. Clonar el repositorio

```
git clone [URL_DEL_REPOSITORIO]
cd [NOMBRE_DEL_PROYECTO]
```

2. Instalar dependencias

```
npm install
```

3. Configurar variables de entorno

```
cp .env.example .env
# Editar .env con las configuraciones necesarias
```

Scripts Disponibles

- `npm run dev` - Inicia el servidor de desarrollo con hot-reload
- `npm run build` - Construye la aplicación para producción con optimizaciones
- `npm run lint` - Ejecuta el linter con reglas personalizadas
- `npm run preview` - Previsualiza la build de producción localmente

- `npm run test` - Ejecuta las pruebas unitarias
- `npm run format` - Formatea el código según los estándares

Convenciones de Código

- ESLint para linting
- Prettier para formateo
- Component naming: PascalCase
- File naming: kebab-case

Optimización y Rendimiento

- Code splitting por rutas
- Lazy loading de componentes
- Optimización de imágenes
- Caché de API
- Bundle size optimization
- Tree shaking
- Minificación de assets

Dependencias Principales

Dependencias de Producción

- @microsoft/clarity: Análisis de comportamiento de usuarios
- axios: Cliente HTTP con interceptores
- chart.js: Visualización de datos avanzada
- react-icons: Iconos vectoriales
- react-modal: Sistema de modales accesible
- react-router-dom: Enrutamiento declarativo
- react-tsparticles: Efectos visuales
- swiper: Carruseles responsivos

Dependencias de Desarrollo

- @vitejs/plugin-react: Optimización de React
- eslint: Linting avanzado
- sass: Estilos modulares
- vite: Build tooling
- @types/react: Tipos TypeScript
- prettier: Formateo de código