## Documentación Técnica — Sistema de Registro de Estudiantes

### Arquitectura General

**Frontend:**

* React con Vite
* React Router para navegación
* Context API para manejo de autenticación
* Material UI para componentes
* Axios para llamadas HTTP

**Backend (.NET Core):**

* ASP.NET Core Web API
* Entity Framework Core con SQL Server
* Autenticación con JWT
* Roles: Admin, Student

### 🔄 Flujo de Autenticación y Autorización

1. **Inicio de sesión (**/login**)**
   * El usuario envía email y contraseña al backend.
   * El backend valida y responde con un token JWT y una fecha de expiración.
   * El frontend guarda el token en localStorage y lo usa en futuras peticiones (vía Axios interceptor).
2. **Protección de rutas**
   * El frontend protege rutas usando un componente ProtectedRoute, que verifica el token y el rol del usuario.
   * Si no hay token o el rol no coincide, redirige a /login.

### 📁 Endpoints de la API

#### 🔐 Autenticación

* POST /api/auth/login  
  **Request:**

json

CopyEdit

{

"email": "paola@email.com",

"password": "miContraseñaSegura123"

}

**Response:**

json

CopyEdit

{

"token": "JWT-TOKEN-HERE",

"expiration": "2025-05-20T19:16:22.935936Z"

}

#### 👨‍🎓 Estudiantes (/api/students)

* GET /api/students — Obtener todos los estudiantes (Admin)
* GET /api/students/{id} — Obtener un estudiante por ID
* POST /api/students — Crear un nuevo estudiante
* PUT /api/students/{id} — Actualizar estudiante
* DELETE /api/students/{id} — Eliminar estudiante

#### 👩‍🏫 Profesores (/api/teachers)

* GET /api/teachers
* POST /api/teachers
* PUT /api/teachers/{id}
* DELETE /api/teachers/{id}

#### 📚 Asignaturas (/api/subjects)

* GET /api/subjects — (Admin y Student)
* POST /api/subjects — Crear nueva asignatura (Admin)
* PUT /api/subjects/{id}
* DELETE /api/subjects/{id}

#### 📝 Matrículas (/api/enrollments)

* GET /api/enrollments — Solo estudiantes autenticados
* POST /api/enrollments — Registrar estudiante en asignatura
* DELETE /api/enrollments/{id} — Cancelar matrícula

### 🔒 Seguridad

* **JWT Token:** Incluido en cada request en Authorization: Bearer <token>
* **Validaciones por rol:** Implementadas en frontend (React) y backend (con [Authorize(Roles = "Admin")] o "Student").

### Manejo de Errores

El backend responde con:

* 200 OK en éxito
* 400 Bad Request en errores de validación
* 401 Unauthorized si no hay token o es inválido
* 403 Forbidden si el usuario no tiene permisos

### Consideraciones Finales

* El token se guarda en localStorage.
* El frontend decodifica el token para obtener el role, fullName y userId.
* El backend utiliza User.Claims para validar y aplicar lógica según el rol.