Universidad Iberoamericana

Facultad de Ingeniería

Ingeniería de software.

**Proyecto de software**

Andres Felipe Pipicano Villa Código Banner 100149864

Unidad de Aprendizaje III

**Manual Técnico**

**Enlaces y accesos importantes del entregable**

* OneDrive: Se encuentra la documentación correspondiente al entregable (Prototipo en PDF, Documentación del proyecto, Diagramas, Scripts SQL , Documentación de las pruebas unitarias de integración y pruebas automatizadas prototipo wirefre y demás documentación generada) [Click para abrir OneDrive](https://laiberocol-my.sharepoint.com/:f:/g/personal/apipican_estudiante_ibero_edu_co/EjEO6ZMf-7xOrF3RqMc6jy8BYrAxurZkUt-TUCgzRvJSXg?e=TbLXas)
* Link ejecutable de prototipo figma: [Click para abrir ejecutable prototipo figma](https://www.figma.com/proto/i7e68PEGorysiRwR0hRIjn/Guarderia?node-id=5-14&p=f&t=k3EvcXOH8IFloaWr-1&scaling=scale-down-width&content-scaling=fixed&page-id=0%3A1&starting-point-node-id=5%3A14&show-proto-sidebar=1)
* Enlace Repositorio GitHub: <https://github.com/AnDrEs-F15/Guarderia-Ibero> (También está la documentación del proyecto en la carpeta Documentación V2) El repositorio esta tanto el API en SpringBoot y el APP en React
* Enlace aplicación desplegada: <https://guarderia-ibero.onrender.com>
* Enlace Video: [Click ver video](https://laiberocol-my.sharepoint.com/:v:/g/personal/apipican_estudiante_ibero_edu_co/EWkvkVIyCqxLh6nXhbcjtMYBzsm52OqqqT_jAk4jjL6T0Q?nav=eyJyZWZlcnJhbEluZm8iOnsicmVmZXJyYWxBcHAiOiJTdHJlYW1XZWJBcHAiLCJyZWZlcnJhbFZpZXciOiJTaGFyZURpYWxvZy1MaW5rIiwicmVmZXJyYWxBcHBQbGF0Zm9ybSI6IldlYiIsInJlZmVycmFsTW9kZSI6InZpZXcifX0%3D&e=eVc93p)

**Credenciales Para ingresar a la página administradora:**

**User: IberoUser**

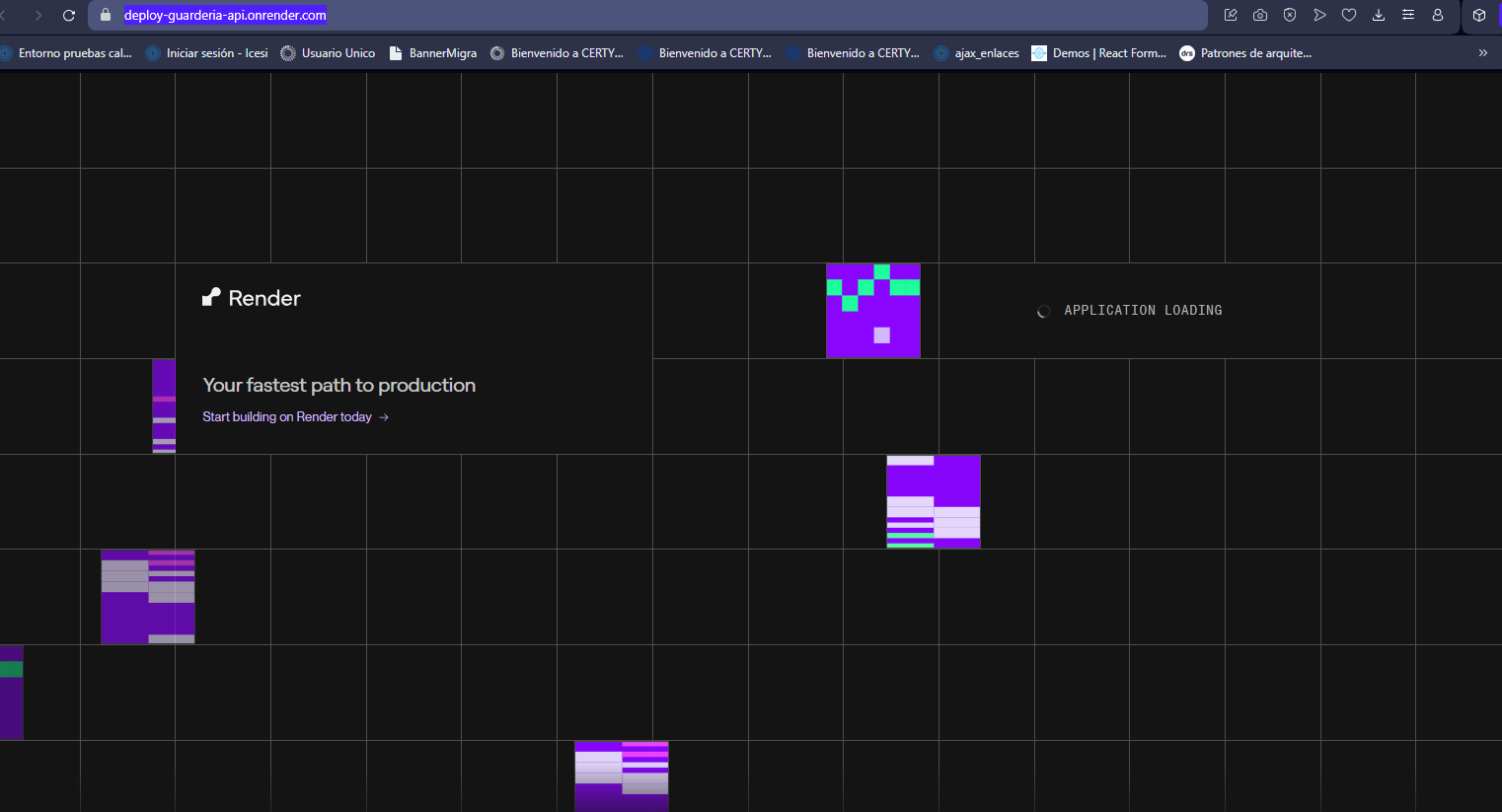
**Password:** **ProySoftware123**

Nota importante el servicio (Render) el cual use para desplegar la aplicación detecta si esta inactivo y se pausa es posible que cuando trate de ingresar el servicio este inactivo para esto por favor antes de realizar una petición al backend por favor ingrese a este URL cuando este nuevamente up le dará un mensaje de ok

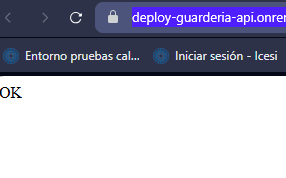
****

<https://deploy-guarderia-api.onrender.com>

Así se ve cuando esta levantando el servicio



Cuando termine y pueda realizar peticiones al backend nuevamente se vería así



**Documento de formulación del proyecto.**

**Resumen del Proyecto**

Este proyecto consiste en el desarrollo de un sistema cliente-servidor para gestionar digitalmente el proceso de inscripción en una guardería. Se implementa una aplicación web que permite a los padres registrar a sus hijos mediante un formulario en línea, y a su vez proporciona al personal administrativo una interfaz para gestionar inscripciones y visualizar la información de forma estructurada y segura. El sistema busca mejorar la eficiencia, reducir errores y facilitar el acceso a los datos.

Tecnologías Principales Utilizadas

**Backend**:

* Java 17
* Spring Boot 3
* Spring Security (con JWT)
* Spring Data JPA
* Base de datos Oracle (SQLDeveloper)

**Frontend**:

* React 18.3.1
* Material UI (MUI)
* React Hook Form (RHF) + Yup para validaciones
* Axios
* Redux Toolkit
* React Router Dom
* React Query

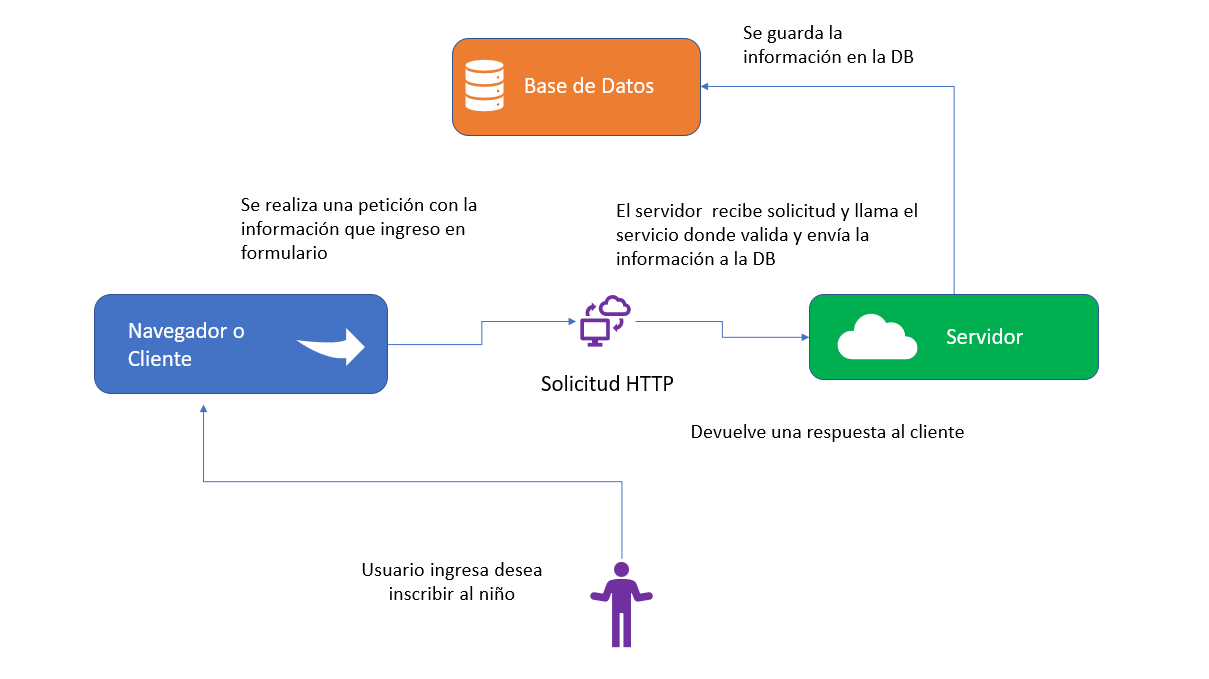
**Despliegue:**

* Render
* Docker
* PostgreSQL (Debido que render tiene solo servicio para DB PostgreSQL fue más fácil cambiarla)

**Arquitectura General**

La aplicación sigue una arquitectura cliente-servidor desacoplada. El frontend React se comunica con el backend Spring Boot mediante API REST. La seguridad se implementa con Spring Security y JWT, protegiendo rutas y datos. Los formularios se validan en el cliente con RHF (React Hook Form) + Yup, y los datos se almacenan en una base de datos Oracle a través de Spring Data JPA. El flujo general garantiza una experiencia de usuario fluida y una gestión administrativa eficiente.

**Arquitectura del funcionamiento de la aplicación grafico**



**Estructura de proyecto Backend (Spring Boot)**

Dentro de la carpeta src/main/java, se encuentra el paquete principal del proyecto, que contiene la organización base de los distintos componentes de la API.

* **Auth** – Configuración de Spring Security Filtros o interceptores y servicio para gestionar el JWT
* **Constants** – Paquete donde para las constantes EJ. esta las constantes de URI de los endpoint
* **Controllers** - contiene los controladores REST que gestionan las solicitudes entrantes.
* **Dto** - define los objetos de transporte de datos entre capas.
* **Entity** - Las entidades que representan las tablas en la base de datos Oracle.
* **Repository** – Define las interfaces que extienden de JpaRepository utilizadas para realizar operaciones CRUD con la base de datos. Spring Data se encarga de su implementación automática.
* **Service** – Implementa la lógica de negocio de la aplicación. Los servicios intermedian entre los controladores y los repositorios, aplicando reglas de negocio y validaciones.

**Estructura principal del proyecto Frontend (React)**

Dentro de la carpeta src/, se encuentra el paquete principal del proyecto, que contiene la organización base de los distintos componentes de la APP React.

* Api – Carpeta donde agrupa los llamados HTTP al API con axios
* Components – Carpeta que agrupa los componentes reutilizables de la aplicación,
* Config -incluye la configuración de librerías clave de la aplicación, como React Query
* Context -> Define y exporta contextos globales para el manejo del estado compartido entre componentes, cuando no se requiere Redux.
* Hooks - Carpeta que agrupa los custom hooks globales de la app
* Layout - Define los templates estructurales de la interfaz, como el Dashboard
* Routes - Rutas de navegación de la app
* Store – Configuración y gestión de slices de redux toolkit
* Views – Representa las páginas principales de la aplicación, como el login, el formulario de inscripción, la gestión de cursos, el panel administrativo, entre otros. Cada vista puede estar compuesta por múltiples componentes internos.

**Seguridad**

El sistema implementa un esquema de seguridad basado en Spring Security 6 junto con JSON Web Tokens (JWT) para proteger los recursos del backend y controlar el acceso según roles definidos.

**Procesamiento:**

La autenticación se realiza mediante JWT, que se genera en el login y se incluye en las solicitudes posteriores mediante la cabecera Authorization. El token contiene los datos del usuario y sus roles, y se valida en cada petición protegida por filtros personalizados.

**Flujo de Autenticación**

* El usuario ingresa sus credenciales mediante el formulario de login.
* El backend valida las credenciales y genera un JWT si son correctas.
* El token es devuelto al cliente y almacenado en el frontend (localStorage).
* En cada solicitud protegida, el cliente incluye el token en la cabecera: (Authorization: Bearer <token generado>)

**Modelado de Datos**

Entidades Principales

**Teachers (Profesores):**

Guarda información básica de los profesores, incluyendo documento de identidad, nombre, teléfono y correo. Tiene un campo ACTIVE para indicar si el profesor está activo.

**Courses (Cursos):**

Define los cursos disponibles en la guardería con atributos como nombre, rango de edad permitido (MIN\_AGE\_MONTHS, MAX\_AGE\_MONTHS), capacidad máxima, turno (SHIFT), ubicación y un profesor principal asignado.

**Children (Niños):**

Registra la información personal de los niños inscritos, incluyendo documento, nombre, fecha de nacimiento, género y datos médicos relevantes (notas médicas y alergias).

**Guardians (responsables):**

Contiene datos de los tutores o responsables legales de los niños, con información de contacto y parentesco.

**Enrollments (Inscripciones):**

Relaciona a un niño con un curso y un tutor, registrando la fecha de inscripción y el estado actual. Es la entidad central que vincula a niños, tutores y cursos.

**Documents (Documentos):**

Almacena archivos asociados a una inscripción, tales como documentos de identidad o certificados de vacunación. Puede almacenar el archivo directamente o solo la ruta.

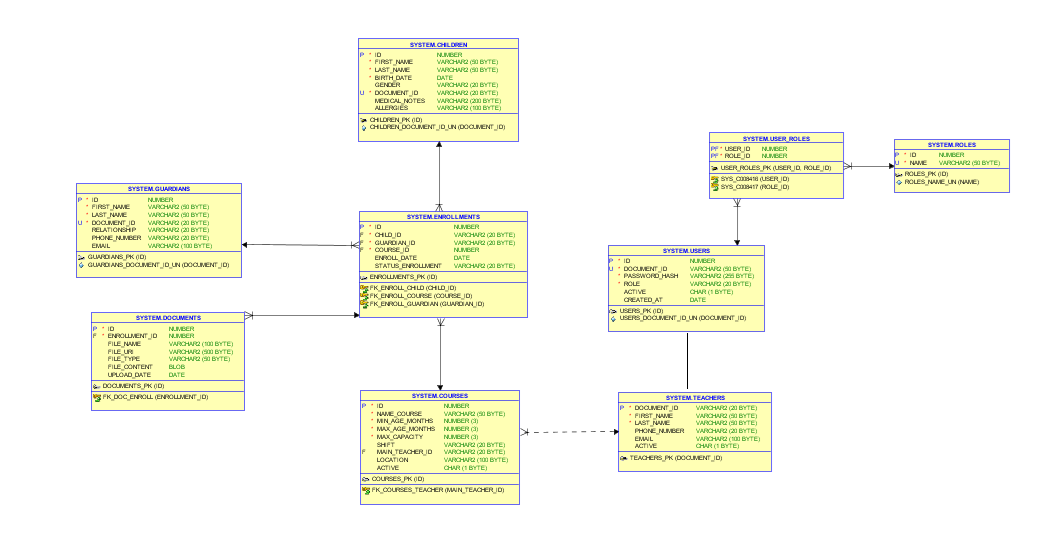
**Users (Usuarios):**

Registra los usuarios del sistema con credenciales y rol asignado (ADMIN, TEACHER, GUARDIAN, etc.), para la gestión de acceso.

**Roles y User\_Roles:**

Gestionan la asignación de roles a los usuarios mediante una relación muchos a muchos, permitiendo múltiples permisos por usuario.

**Diagrama ER** (También adjunto en la documentación)



**Detalles de API REST (Principales Endpoints)**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Método | Endpoint URI | Descripción | Requiere Auth | Parámetros |
| /authentication |  |  |  |  |
| POST | /authentication | Registrar un nuevo usuario | No | Body: RegisterAuth |
| POST | /authentication/login | Generar token JWT | No | Body: AuthCredentials |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Método | | Endpoint | Descripción | Requiere Auth | Parámetros |
| /api/enrollment |  | |  |  |  |
| GET | /api/enrollment | | Obtener todas las inscripciones | Si | — |
| GET | /api/enrollment/{id} | | Obtener inscripción por ID | Si | id: Long |
| POST | /api/enrollment | | Crear nueva inscripción | Si | Body: Enrollments |
| PUT | /api/enrollment/{id} | | Actualizar inscripción completa | Si | id, Body: Enrollments |
| PATCH | /api/enrollment/{id}/{status} | | Actualizar estado de inscripción | Si | id: Long, status: String |
| DELETE | /api/enrollment/{id} | | Eliminar inscripción | Si | id: Long |
| GET | /api/enrollment/{courseId}/students | | Obtener estudiantes aceptados por curso | Si | courseId: Long |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Método | Endpoint | Descripción | Requiere Auth | Parámetros |
| /api/course |  |  |  |  |
| GET | /api/course/active | Obtener cursos activos | SI |  |
| GET | /api/course/{id} | Obtener curso por ID | SI | id: Long |
| POST | /api/course | Crear un nuevo curso | SI | Body: Course |
| DELETE | /api/course/{id} | Eliminar curso | SI | id: Long |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Método | Endpoint | Descripción | Requiere Auth | Parámetros |
| /api/professor |  |  |  |  |
| GET | /api/professor | Obtener todos los profesores (paginado) | SI | Parámetros de paginación (Pageable) |
| GET | /api/professor/{id} | Obtener profesor por documento ID | SI | id: String |
| POST | /api/professor | Crear nuevo profesor | SI | Body: Professor |
| PUT | /api/professor/{id} | Actualizar profesor | SI | id, Body: Professor |
| DELETE | /api/professor/{id} | Eliminar profesor | SI | id: String |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Método | Endpoint | Descripción | Requiere Auth | Parámetros |
| /api/child |  |  |  |  |
| GET | /api/child | Obtener todos los niños | SI | — |
| GET | /api/child/{id} | Obtener niño por documento ID | SI | id: String |
| POST | /api/child | Crear nuevo niño | SI | Body: Child |
| PUT | /api/child/{id} | Actualizar información del niño | SI | id, Body: Child |
| DELETE | /api/child/{id} | Eliminar niño | SI | id: String |

**Flujo de Funcionamiento del Sistema**

El sistema de inscripción digital para la guardería opera bajo el siguiente flujo general:

1. Registro de Inscripción por Parte del Padre o Acudiente:

* El padre o acudiente accede al formulario de inscripción en línea.
* Completa la información personal del niño, incluyendo nombre, fecha de nacimiento, género, notas médicas y alergias.
* Selecciona el curso al que desea inscribir al niño, indicando la jornada correspondiente (mañana, tarde o jornada completa).
* Ingresa los datos del acudiente, como nombre, documento, parentesco, teléfono y correo electrónico.
* De manera opcional, puede adjuntar documentos relevantes (por ejemplo, documento de identidad, carné de vacunación, etc.).

1. Revisión y Gestión por Parte del Administrador:

* Un usuario con rol administrador accede al panel de gestión.
* Revisa las inscripciones pendientes y, según criterio, puede aprobar o rechazar las solicitudes.
* Tiene la capacidad de crear nuevos cursos, especificando parámetros como edad mínima/máxima, jornada, capacidad máxima, ubicación y profesor principal.
* Puede registrar profesores y asignarlos a los cursos disponibles, asegurando que cada grupo tenga un docente responsable.

1. Comunicación y Seguimiento:

* La información queda registrada en la base de datos Oracle, permitiendo su posterior consulta, auditoría o generación de reportes.

**Conclusión**

El desarrollo e implementación del sistema digital de inscripción para la guardería ubicada en el barrio Decepaz representa un avance significativo en la modernización de sus procesos administrativos. Al sustituir los formularios físicos por una solución web segura, accesible y eficiente, se logró reducir los errores de transcripción, optimizar los tiempos de gestión y mejorar la experiencia tanto para los padres como para el personal.

El uso de tecnologías modernas como Spring Boot 3 con seguridad basada en JWT, una base de datos Oracle robusta y un frontend dinámico construido en React con herramientas como MUI, React Hook Form, Redux y Axios, permite que el sistema sea escalable, mantenible y fácil de usar.

Este sistema sienta las bases para una transformación digital completa, y se encuentra abierto a futuras ampliaciones como notificaciones automáticas, reportes estadísticos o integración con plataformas educativas.

# Referencias