Prueba Técnica Full-Stack: Sistema de Inscripción Académica con Aulas

Objetivo: Diseñar un sistema backend que permita gestionar la inscripción de estudiantes a cursos, asegurando que no haya conflictos de horarios, con una interfaz frontend para facilitar la interacción del usuario. Los reportes deben poder exportarse en **PDF** y **Excel**.

Requerimientos funcionales

1. **Gestión de períodos académicos**:
    - Backend: CRUD de períodos académicos.
    - Frontend: Formulario para registrar, editar y eliminar períodos académicos.
        - Campos: Nombre del período, Fecha de inicio y fin.
2. **Gestión de cursos**:
    - Backend: CRUD de cursos con validación de aulas y horarios.
        - Campos: Código del curso, Nombre del curso, Docente asignado, Aula, Horario del curso (día, hora inicio y hora fin), Período académico.
        - **Validación**: Un curso no puede solaparse con otro curso en el mismo aula en el mismo horario.
    - Frontend: Formulario para registrar, editar y eliminar cursos.
        - Incluir selección de aula, horario, y docente.
3. **Gestión de estudiantes**:
    - Backend: CRUD de estudiantes.
        - Campos: Matrícula, Nombre completo, Correo electrónico.
    - Frontend: Formulario para registrar, editar y eliminar estudiantes.
        - Incluir campos para matrícula, nombre y correo electrónico.
4. **Inscripción de estudiantes a cursos**:
    - Backend: API para inscripciones, con validaciones.
        - Un estudiante no puede inscribirse en dos cursos con horarios que se solapen.
        - Limitar el número máximo de estudiantes por curso-aula.
    - Frontend: Vista donde los estudiantes puedan:
        - Consultar los cursos disponibles.
        - Ver los cursos ya inscritos y los horarios de los mismos.
        - Inscribirse a un curso (validando la disponibilidad y evitando los solapamientos).
        - **Formularios** para inscribirse a un curso, con validación en tiempo real.
5. **Reportes**:
    - **PDF**:
        - Generar un reporte por estudiante con:
            - Nombre y matrícula del estudiante.
            - Cursos inscritos con aula, horario, docente.
            - Total de cursos inscritos.
    - **Excel**:

        - Generar un reporte por curso-aula con:
            - Código, nombre del curso, aula, docente.
            - Lista de estudiantes inscritos con matrícula y nombre.
            - Estadísticas de inscripciones y lugares disponibles.
    - Backend: Endpoint para descargar los reportes en PDF y Excel.
    - Frontend: Interfaz para visualizar y descargar los reportes.

Requerimientos técnicos

1. **Backend (Laravel)**:
    - Crear una API RESTful con las siguientes rutas:
        - **Períodos Académicos**: CRUD (Registro, edición, eliminación).
        - **Cursos**: CRUD con validación de solapamientos de horarios y aula.
        - **Estudiantes**: CRUD.
        - **Inscripciones**: Registro, consulta de inscripciones por estudiante o curso-aula.
        - **Reportes**: Endpoint para generar y descargar reportes en PDF y Excel.
    - Validaciones:
        - Evitar cruces de horarios en cursos en el mismo aula.
        - Validar que no se inscriban más estudiantes de los permitidos en un curso-aula.
        - Evitar duplicados de inscripciones.
    - Implementación de relaciones y migraciones:
        - Crear relaciones entre **Estudiantes** y **Cursos** (a través de **Inscripciones**).
        - Crear datos iniciales (seeders).
    - Generación de reportes:
        - PDF con **DomPDF**.
        - Excel con **Maatwebsite\Excel**.
2. **Frontend (React.js)**:
    - Usar **React.js** para crear la interfaz de usuario interactiva.
        - Formularios para:
            - Registrar, editar y eliminar períodos académicos, cursos y estudiantes.
            - Inscripción de estudiantes a cursos.
        - Visualización de las inscripciones de un estudiante, con los cursos y horarios asociados.
        - Verificación de la disponibilidad de cupos y validación de solapamientos de horarios.
    - Integración con la API backend usando **Axios** o **Fetch**.
    - Página para mostrar los reportes generados, con opción para descargar en **PDF** y **Excel**.

3. **Extras opcionales (opcional, para puntos adicionales)**:
    - Implementar autenticación de usuarios con **JWT** para el frontend y backend.
    - Añadir roles (Administrador y Estudiante) para controlar el acceso a las funcionalidades del sistema.
    - Implementación de pruebas unitarias y funcionales con PHPUnit y Jest.
    - Implementar paginación para la visualización de estudiantes y cursos.
    - Añadir una función de búsqueda y filtrado para cursos y estudiantes.
    - Implementar un sistema de notificación por correo cuando un estudiante se inscriba en un curso.
    - Crear un reporte consolidado en Excel que contenga:
        - Resumen de cursos-aulas.
        - Detalles de los estudiantes inscritos.

Criterios de evaluación

1. **Backend**:
    - Correcta implementación de las validaciones para horarios y aulas.
    - Organización del código y uso adecuado de las relaciones entre modelos.
    - Funcionalidad completa de reportes en PDF y Excel.
2. **Frontend**:
    - Implementación correcta de las interfaces de usuario
    - Integración adecuada con la API backend.
    - Experiencia de usuario intuitiva y clara.
3. **General**:
    - Uso de Git con commits claros y frecuentes.
    - Documentación clara en el README:
        - Pasos de configuración.
        - Descripción de los endpoints de la API.
        - Descripción de la estructura del frontend.

Instrucciones

1. Crea un repositorio en **GitHub** para el proyecto.
2. Documenta el proyecto en el archivo **README.md**:
    - Instalación y configuración del entorno.
    - Descripción de los endpoints de la API y ejemplos de uso.
    - Descripción de la estructura de las vistas y cómo usar la interfaz frontend.
    - Incluye una colección de **Postman** para facilitar las pruebas de la API.
3. Comparte el enlace del repositorio antes de la fecha límite.

Tiempo estimado

5 días.