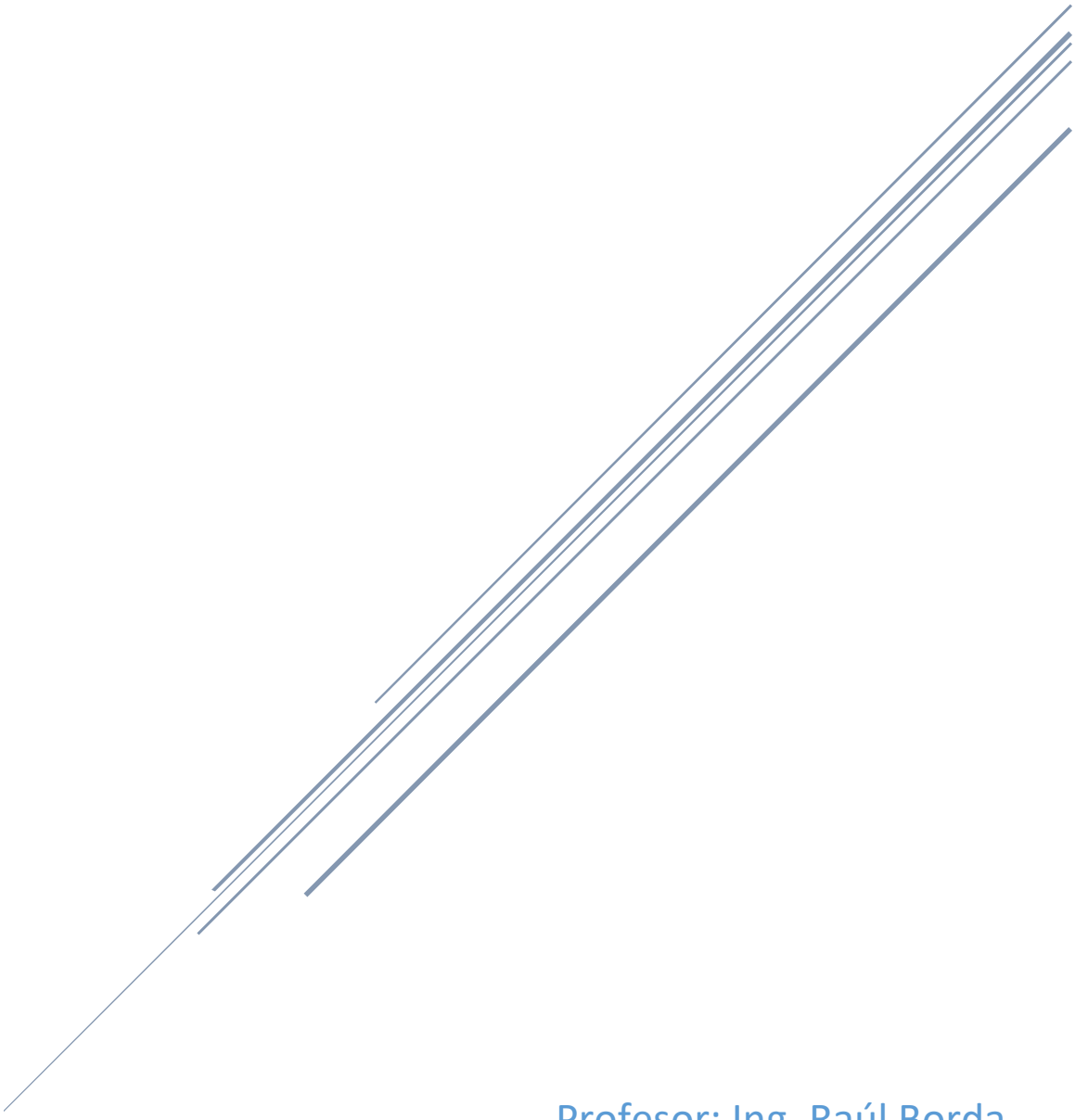
TP 4 - COLOQUIO REDES

Instituto Leibnitz



Profesor: Ing. Raúl Borda
Estudiante: Nievas Nicolas A.

TP 4 - Redes - NIEVAS Nicolas

Profesor: Ing. Raúl Alberto Borda

Sistema de generación de informes escolares utilizando IA:

Este sistema permite gestionar alumnos y generar informes de evolución detallados a partir de ciertas características que deberá proporcionar el maestro/a.

Luego, estos informes podrán ser descargados en formato PDF para su fácil almacenamiento e impresión.

Repositorio GitHub: nicoonievas/TP4-RedesNievasNicolas

https://github.com/nicoonievas/TP4-RedesNievasNicolas

BACKEND:

MANIPULACION DE DATOS

FRONTEND

CONCLUSION:

Este trabajo me permitió implementar la utilización de una IA en un proyecto propio. La manipulación de los datos no fue complicado ya que estamos muy familiarizados con el formato JSON.

Al utilizar MongoDB directo (sin Mongoose), encontré mucho mas simple la inserción de datos sin utilizar esquemas intermedios.

La generación del PDF fue lo mas complicado ya que tuve que implementar nuevas librerías y darle formato al texto que se utilizara en el documento final.

Tuve algunos inconvenientes ya que al principio, los PDF salían sin datos o en formato inaccesible. Lo pude corregir con ayuda de Chat GPT, quien me recomendó modificar el endpoint para que sea asincrónico, lo cual hace que la ejecución espere a que el PDF este completamente generado antes de ser enviado al frontend.

BACKEND:

Los endpoints utilizados son:

POST/api/alumno: Este endpoint permite agregar un nuevo alumno a la base de datos. Recibe los datos del alumno (`firstname`, `lastname`, `grado`, `instituto`) desde el frontend y los guarda en la colección `alumnos`.

Adicionalmente se agrega el mail del usuario logueado para luego mostrar en la tabla aquellos alumnos guardados con un usuario especifico.


```
_id: ObjectId('6731550266f0cd7d0e8b1ed1')
usuario : "nicolasnievas1@gmail.com"
firstname : "Florencia"
lastname : "Romano"
grado : "4"
instituto : "Inst. del Rosario"
```

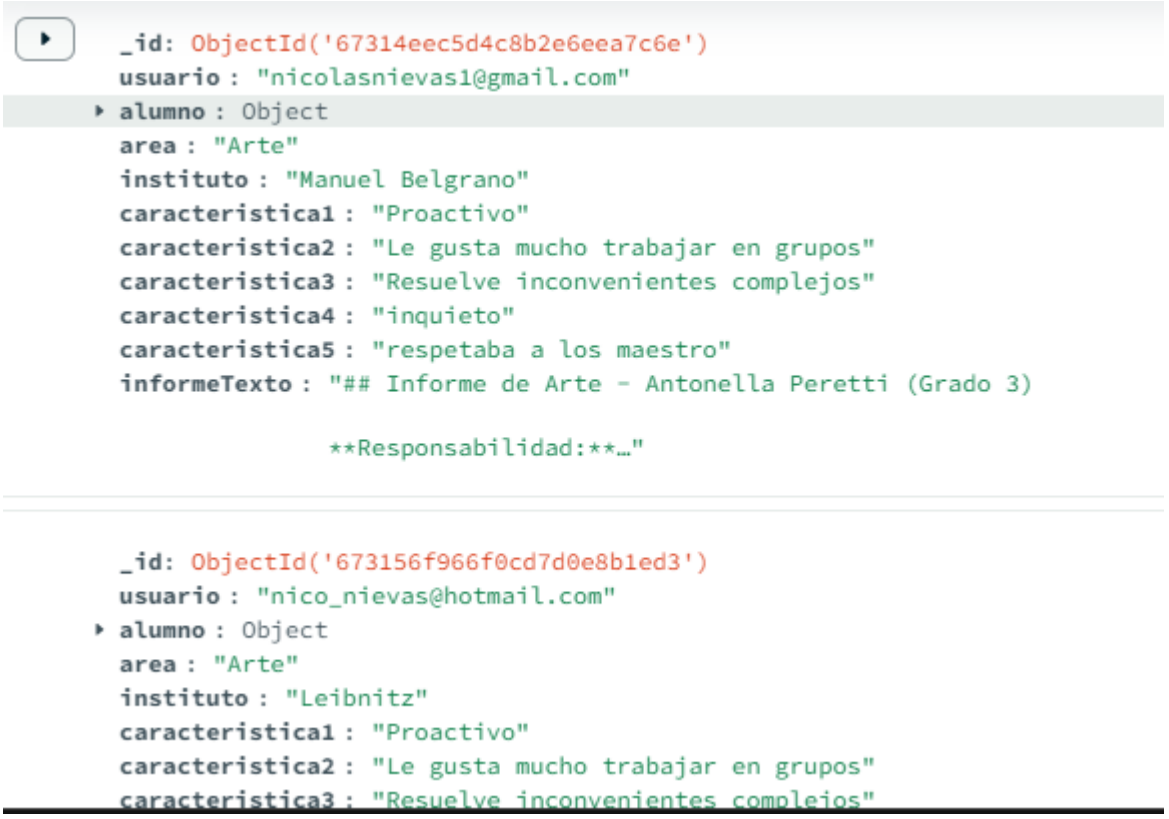
```
_id: ObjectId('673156db66f0cd7d0e8b1ed2')
usuario : "nico_nievas@hotmail.com"
firstname : "Facundo"
lastname : "Rabbia"
grado : "3"
instituto : "Leibnitz"
```


POST/api/informes: Este endpoint recibe los datos de un alumno y características específicas a través del frontend.

- Busca los datos del alumno en la base de datos MongoDB.
- Genera un prompt detallado usando los datos del alumno y las características proporcionadas.
- Envía este prompt a la API de Gemini para obtener un informe generado automáticamente.
- Almacena el informe junto con los detalles del alumno en la colección `informes` de la base de datos.

- Finalmente, responde al frontend con el texto del informe.

Adicionalmente se agrega el mail del usuario logueado para luego mostrar en la tabla aquellos informes generados por un usuario especifico.



GET /api/alumnos: Recupera y envía una lista de todos los alumnos guardados en la colección `alumnos` de MongoDB para cada usuario especifico. Esto permite que el frontend muestre la lista de alumnos disponibles.

GET /api/informes: Este endpoint envía una lista de todos los informes almacenados en la base de datos de cada usuario especifico al frontend, lo que permite a los usuarios visualizar los informes generados por ellos mismos.

GET /api/informePDF/: Este endpoint genera un PDF a partir de un informe ya guardado en la base de datos.

- Busca el informe usando el `informeId` proporcionado en la URL.

- Si encuentra el informe, usa la función `generarInformePDF` para crear un PDF con los datos del informe y lo envía al usuario.
- Después de descargar el PDF, el archivo temporal se elimina del servidor para liberar espacio.

PUT /api/alumno/:id: Este endpoint permite editar los datos del alumno previamente guardado en la Base de Datos.

- Los campos editables son Nombre, Apellido, Institución y Grado

PUT /api/informe/:id: Este endpoint permite editar el informe previamente guardado en la Base de Datos.

- El único campo editable es el informe generado por la IA, en caso que el usuario desee modificar la información.

DELETE/api/informe/:id: Este endpoint permite ELIMINAR el informe guardado en la Base de Datos.

DELETE/api/alumno/:id: Este endpoint permite ELIMINAR el alumno guardado en la Base de Datos.

MANIPULACION DE DATOS

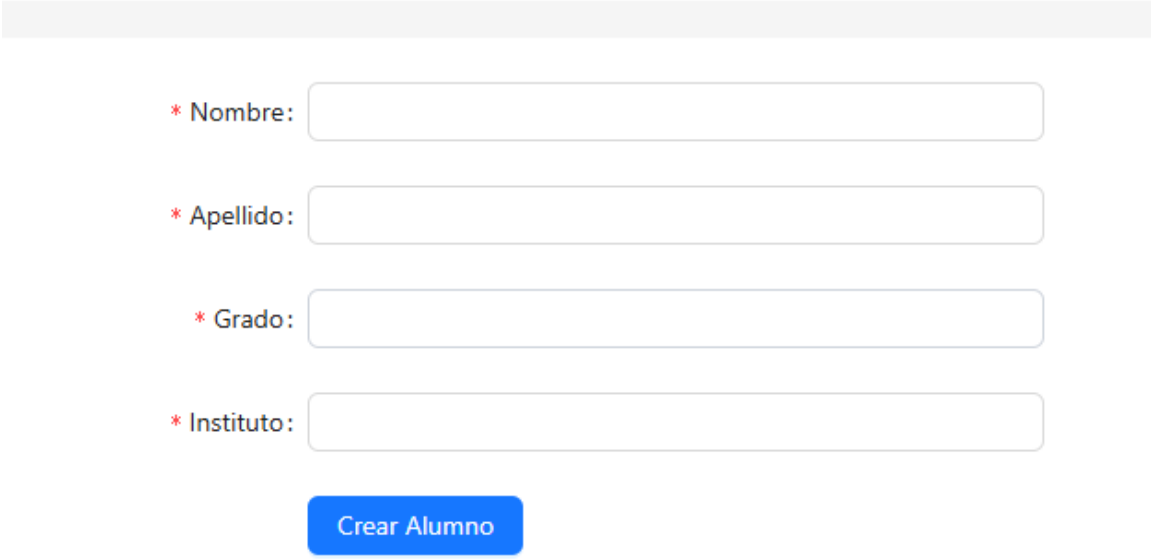
- **Conexión a la base de datos**: Se utiliza una función `connectDB` para establecer la conexión con MongoDB y realizar las inserciones y consultas en las colecciones `alumnos` e `informes`.
- **Generación de informes**: Se interactúa con la API de Gemini para recibir el contenido generado en base a un prompt específico.
- **Generación de PDF**: La función `generarInformePDF` convierte el texto del informe a un formato PDF y lo envía al front para que sea descargado por el maestro.

```
app.get('/api/informePDF/:informeId', async (req, res) => {
    const { informeId } = req.params;
    const { apellido, año, area } = req.query;

    try {
        const db = await connectDB();
        const collection = db.collection('informes');
        const informe = await collection.findOne({ _id: new ObjectId(informeId) });

        if (!informe) {
            return res.status(404).json({ error: 'Informe no encontrado' });
        }

     //utilizo apellido, año y area para colocar en el nombre del PDF
        const fileName = `Informe_${apellido}_${año}_${area}.pdf`;
        const outputPath = path.join(__dirname, fileName);

       //esperar a que este listo el PDF
        await generarInformePDF(informe, outputPath);

        res.download(outputPath, fileName, (err) => {
            if (err) {
                console.error('Error al enviar el PDF:', err);
                res.status(500).send('Error al enviar el PDF');
            }
```

FRONTEND

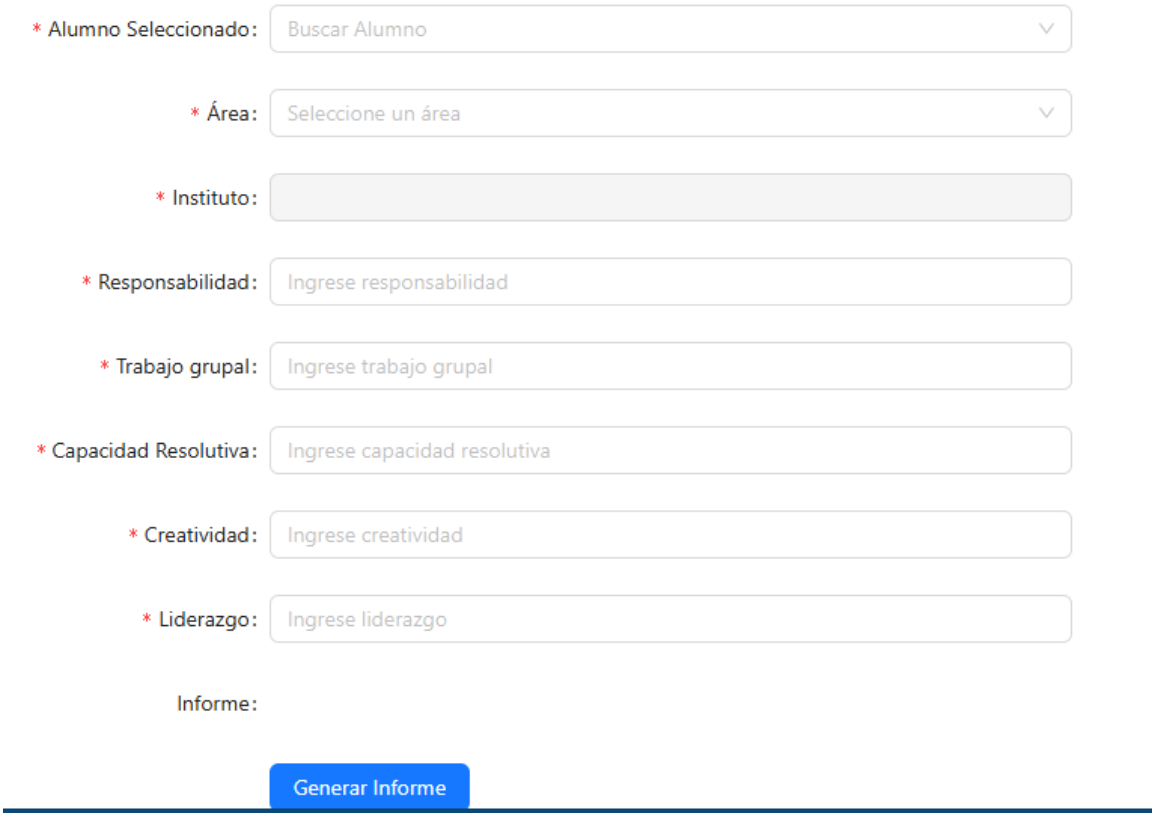
La tecnología utilizada para el Frontend es REACT - ANT Design

- Componentes principales
  - `CrearAlumno.js`: Formulario que permite agregar un nuevo alumno a la base de datos. Una vez enviado, los datos se envían al backend para ser almacenados.



  - `CrearInforme.js`: Formulario para generar un nuevo informe a partir de las características del alumno seleccionado. Este formulario envía una solicitud al backend, el cual usa la API de Gemini para generar el informe

Se agregó en el formulario, los distintos puntos a cumplimentar por el alumno para que el informe respete lo solicitado por la institucion

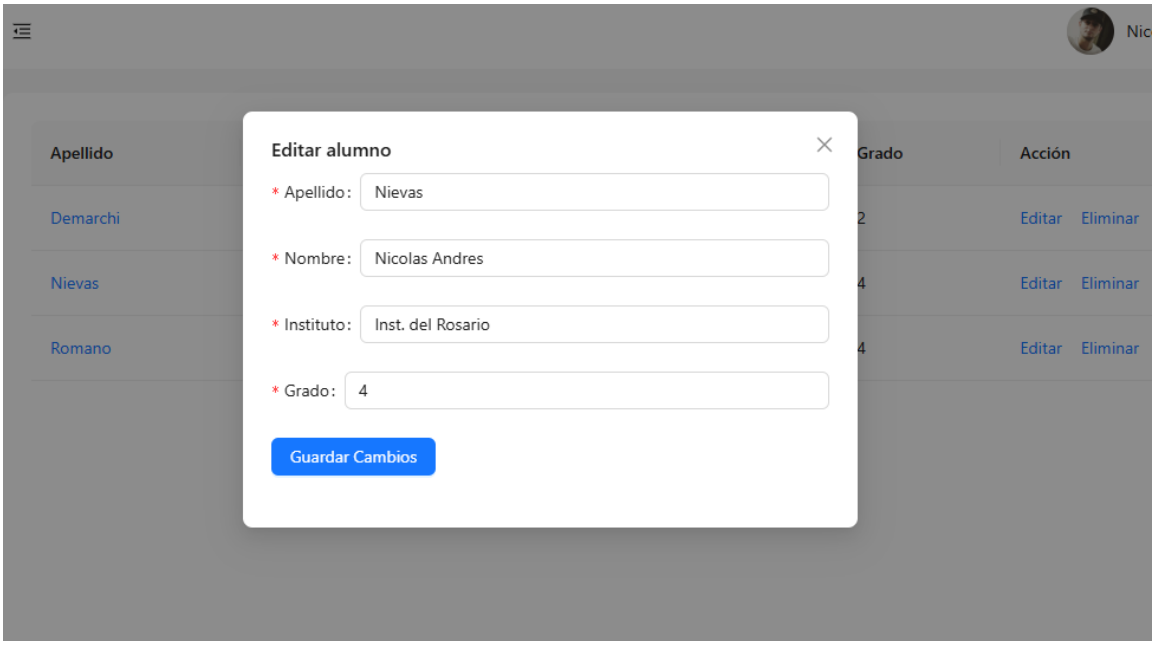


Los alumnos se traen al select de acuerdo al el email del usuario logueado, para que no se puedan crear informes utilizando alumnos de otro usuario.

- `TablaAlumnos.js`: Tabla con todos los alumnos registrados en la base de datos.

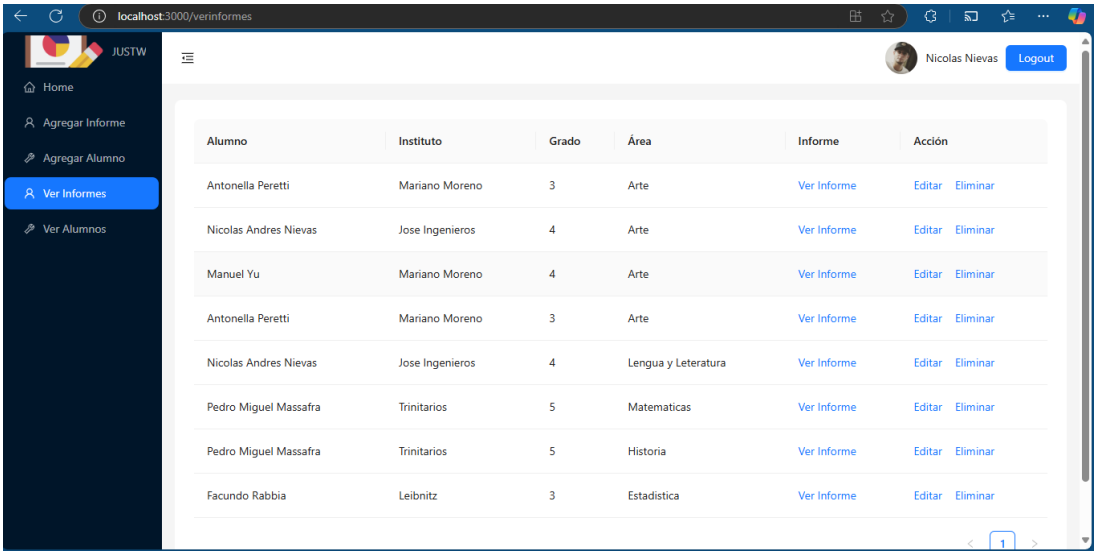
Los alumnos del listado, son aquellos que corresponden al usuario logueado.

Se podrá editar los siguientes datos:



- `TablaInformes.js`: Similar a `TablaAlumnos.js`, este componente muestra una lista de todos los informes generados, permitiendo a los usuarios visualizar los informes previamente guardados en la base de datos. Posteriormente se podrá editar el informe a gusto de cada usuario para que se pueda descargar con las ultimas modificaciones.

  La presente tabla cuenta con un botón en cada registro, el cual permite hacer una petición al backend para la generación y descarga del informe en formato PDF.



La edición del informe permite modificar el texto original generado por la IA en caso que no sea de agrado por el usuario: