Formato de Informe

1. Portada

- **Título del informe**: LABORATORIO 3.1: APLICACIONES (APIS, REST)
- **Asignatura**: Desarrollo web avanzado
- **Nombre del autor o autores**: Ariel Reyes, Anthony Villareal
- **Departamento**: Departamento de Ciencias de la Computación
- **Universidad:** Universidad de las Fuerzas Armadas ESPE
- **Fecha de entrega**: 21/02/2023

2. Índice

- Lista de los apartados y subapartados del informe con sus números de página.

3. Resumen

El propósito de este informe es documentar el desarrollo de una aplicación web en Angular, utilizando PrimeNG para la interfaz gráfica y consumiendo datos de una API en localhost. Se describen las herramientas empleadas, el proceso de implementación y los desafíos encontrados. A lo largo del desarrollo, se implementaron funcionalidades como la gestión de estudiantes y calificaciones, optimización del rendimiento y mejoras en la experiencia de usuario. Los resultados obtenidos muestran que la integración de PrimeNG mejora significativamente la interfaz y la usabilidad de la aplicación. Finalmente, se presentan conclusiones sobre el impacto del uso de Angular y PrimeNG en el desarrollo de aplicaciones web modernas.

4. Introducción

El objetivo principal de este informe es describir el proceso de desarrollo de una aplicación web que permite la gestión de estudiantes y notas mediante el uso de tecnologías modernas como Angular y PrimeNG. Este proyecto surge de la necesidad de contar con un sistema eficiente para almacenar y visualizar información académica de manera estructurada y accesible.

El contexto en el que se desarrolla este proyecto está relacionado con la digitalización de los procesos educativos y administrativos. En muchas instituciones educativas, la gestión de estudiantes y sus calificaciones aún se realiza manualmente o con herramientas que no son óptimas. Con esta aplicación, se busca automatizar estos procesos, mejorando la organización y facilitando el acceso a la información en tiempo real.

El alcance del informe se centra en la implementación de un sistema de gestión con operaciones CRUD para estudiantes y notas, utilizando PrimeNG para la interfaz gráfica. Sin embargo, no se abordan aspectos avanzados como autenticación de usuarios o reportes analíticos detallados, ya que el objetivo es presentar una solución funcional y escalable.

5. Objetivos

- **General**: Desarrollar una aplicación web en Angular utilizando PrimeNG para mejorar la experiencia de usuario en la gestión de estudiantes y sus calificaciones.
- **Específicos**:
  - Implementar una interfaz gráfica moderna utilizando componentes de PrimeNG.
  - Integrar la aplicación con una API en localhost para manejar datos de estudiantes y notas.
  - Optimizar el rendimiento de la aplicación mediante buenas prácticas en Angular.
  - Validar formularios para garantizar la integridad de los datos ingresados.
  - Mejorar la experiencia de usuario a través de un diseño responsivo y accesible.

6. Desarrollo o Cuerpo del Informe

La base del proyecto radica en la implementación de un sistema web que permite realizar operaciones CRUD sobre los estudiantes y sus calificaciones. En términos de herramientas de desarrollo, se ha utilizado Angular como framework principal para la construcción de la aplicación, PrimeNG para la interfaz gráfica y Bootstrap para mejorar el diseño. La API que proporciona los datos se encuentra alojada en localhost:9090 y gestiona las entidades Estudiante y Nota.

Estudiante.component,ts

```
import { Component, OnInit } from '@angular/core';
import { CommonModule } from '@angular/common';
import { FormsModule } from '@angular/forms';
import { EstudianteService } from '../../services/estudiante.service';
import { NotaService } from '../../services/nota.service';
import { Estudiante } from '../../models/estudiante';
import { Nota } from '../../models/nota';

@Component({
  selector: 'app-estudiante',
  standalone: true,
  imports: [CommonModule, FormsModule],
  templateUrl: './estudiante.component.html',
  styleUrls: ['./estudiante.component.css']
})
export class EstudianteComponent implements OnInit {
  estudiantes: Estudiante[] = [];
  estudianteSeleccionado: Estudiante | null = null;
  notas: Nota[] = [];
  nuevoEstudiante: Estudiante = new Estudiante(0, '', '', '', new Date(), []);
  nuevaNota: Nota = new Nota(0, '', 0, new Date(), 0);
```

```
  modoEdicion: boolean = false;

  constructor(
    private estudianteService: EstudianteService,
    private notaService: NotaService
  ) {}

  ngOnInit(): void {
    this.obtenerEstudiantes();
  }

  obtenerEstudiantes(): void {
    this.estudianteService.obtenerTodos().subscribe(data => {
      this.estudiantes = data;
    });
  }

  seleccionarEstudiante(estudiante: Estudiante): void {
    this.estudianteSeleccionado = estudiante;
    this.notaService.obtenerPorEstudianteId(estudiante.id).subscribe(data => {
      this.notas = data;
    });
  }

  agregarEstudiante(): void {
    this.estudianteService.agregarEstudiante(this.nuevoEstudiante).subscribe(data
=> {
      this.estudiantes.push(data);
      this.nuevoEstudiante = new Estudiante(0, '', '', '', new Date(), []);
    });
  }

  editarEstudiante(estudiante: Estudiante): void {
    this.nuevoEstudiante = { ...estudiante };
    this.modoEdicion = true;
  }

  actualizarEstudiante(): void {
    if (this.nuevoEstudiante.id) {
      this.estudianteService.actualizarEstudiante(this.nuevoEstudiante.id,
this.nuevoEstudiante).subscribe(data => {
        const index = this.estudiantes.findIndex(e => e.id === data.id);
        this.estudiantes[index] = data;
        this.cancelarEdicion();
      });
```

```
    }
  }

  eliminarEstudiante(id: number): void {
    this.estudianteService.eliminarEstudiante(id).subscribe(() => {
      this.estudiantes = this.estudiantes.filter(e => e.id !== id);
    });
  }

  agregarNota(): void {
    if (this.estudianteSeleccionado) {
      this.nuevaNota.estudianteId = this.estudianteSeleccionado.id;
      this.notaService.agregarNota(this.nuevaNota).subscribe(data => {
        this.notas.push(data);
        this.nuevaNota = new Nota(0, '', 0, new Date(), 0);
      });
    }
  }

  eliminarNota(id: number): void {
    this.notaService.eliminarNota(id).subscribe(() => {
      this.notas = this.notas.filter(nota => nota.id !== id);
    });
  }

  cancelarEdicion(): void {
    this.nuevoEstudiante = new Estudiante(0, '', '', '', new Date(), []);
    this.modoEdicion = false;
  }
}
```

Estudiante.component.html

```
<div>
    <h2>Estudiantes</h2>
    <form (ngSubmit)="modoEdicion ? actualizarEstudiante() :
agregarEstudiante()">
      <label for="nombre">Nombre:</label>
      <input id="nombre" [(ngModel)]="nuevoEstudiante.nombre" name="nombre"
required>

      <label for="apellido">Apellido:</label>
      <input id="apellido" [(ngModel)]="nuevoEstudiante.apellido" name="apellido"
required>
```

```
      <label for="email">Email:</label>
      <input id="email" [(ngModel)]="nuevoEstudiante.email" name="email"
required>

      <label for="fechaNacimiento">Fecha de Nacimiento:</label>
      <input id="fechaNacimiento" type="date"
[(ngModel)]="nuevoEstudiante.fechaNacimiento" name="fechaNacimiento" required>

      <button type="submit">{{ modoEdicion ? 'Actualizar' : 'Agregar' }}</button>
      <button type="button" *ngIf="modoEdicion"
(click)="cancelarEdicion()">Cancelar</button>
    </form>

    <table>
      <thead>
        <tr>
          <th>Nombre</th>
          <th>Apellido</th>
          <th>Email</th>
          <th>Fecha de Nacimiento</th>
          <th>Acciones</th>
        </tr>
      </thead>
      <tbody>
        <tr *ngFor="let estudiante of estudiantes">
          <td>{{ estudiante.nombre }}</td>
          <td>{{ estudiante.apellido }}</td>
          <td>{{ estudiante.email }}</td>
          <td>{{ estudiante.fechaNacimiento | date }}</td>
          <td>
            <button (click)="editarEstudiante(estudiante)">Editar</button>
            <button (click)="eliminarEstudiante(estudiante.id)">Eliminar</button>
            <button (click)="seleccionarEstudiante(estudiante)">Ver
Notas</button>
          </td>
        </tr>
      </tbody>
    </table>
  </div>

  <div *ngIf="estudianteSeleccionado">
    <h2>Notas de {{ estudianteSeleccionado.nombre }}</h2>
    <ul>
      <li *ngFor="let nota of notas">
        {{ nota.asignatura }} - {{ nota.nota }} ({{ nota.fechaRegistro | date }})
```

```
        <button (click)="eliminarNota(nota.id)">Eliminar</button>
      </li>
    </ul>

    <h3>Agregar Nota</h3>
    <form (ngSubmit)="agregarNota()">
      <label for="asignatura">Asignatura:</label>
      <input id="asignatura" [(ngModel)]="nuevaNota.asignatura" name="asignatura"
required>

      <label for="nota">Nota:</label>
      <input id="nota" type="number" [(ngModel)]="nuevaNota.nota" name="nota"
required>

      <button type="submit">Agregar Nota</button>
    </form>
  </div>
```

Estudiante.ts

```
import { Nota } from './nota';
export class Estudiante {
    id: number;
    nombre: string;
    apellido: string;
    email: string;
    fechaNacimiento: Date;
    notas: Nota[];

    constructor(id: number, nombre: string, apellido: string, email: string,
fechaNacimiento: Date, notas: Nota[]) {
      this.id = id;
      this.nombre = nombre;
      this.apellido = apellido;
      this.email = email;
      this.fechaNacimiento = fechaNacimiento;
      this.notas = notas;
    }
  }
```

Nota.ts

```
export class Nota {
    id: number;
    asignatura: string;
```

```
    nota: number;
    fechaRegistro: Date;
    estudianteId: number;

    constructor(id: number, asignatura: string, nota: number, fechaRegistro:
Date, estudianteId: number) {
      this.id = id;
      this.asignatura = asignatura;
      this.nota = nota;
      this.fechaRegistro = fechaRegistro;
      this.estudianteId = estudianteId;
    }
  }
```

Estudiante.service.ts

```
import { Injectable } from '@angular/core';
import { HttpClient } from '@angular/common/http';
import { Observable } from 'rxjs';
import { Estudiante } from '../models/estudiante';
import { catchError } from 'rxjs/operators'; // Importa catchError
import { throwError } from 'rxjs'; // Importa throwError


@Injectable({
  providedIn: 'root'
})
export class EstudianteService {
  private apiUrl = 'http://localhost:9090/api/estudiantes';

  constructor(private http: HttpClient) {}

  obtenerTodos(): Observable<Estudiante[]> {
    return this.http.get<Estudiante[]>(this.apiUrl);
  }

  obtenerPorId(id: number): Observable<Estudiante> {
    return this.http.get<Estudiante>(`${this.apiUrl}/${id}`);
  }

  agregarEstudiante(estudiante: Estudiante): Observable<Estudiante> {
    // Elimina el ID antes de enviar los datos si existe
    const { id, ...estudianteSinId } = estudiante;  // Desestructura y elimina el
ID
```

```
    console.log('📤 Datos enviados (sin ID):', JSON.stringify(estudianteSinId));

    return this.http.post<Estudiante>(this.apiUrl, estudianteSinId).pipe(
      catchError(error => {
        console.error('Error al agregar estudiante:', error);
        return throwError(() => new Error('Error al agregar estudiante'));
      })
    );
  }


  actualizarEstudiante(id: number, estudiante: Estudiante):
Observable<Estudiante> {
    return this.http.put<Estudiante>(`${this.apiUrl}/${id}`, estudiante);
  }

  eliminarEstudiante(id: number): Observable<void> {
    return this.http.delete<void>(`${this.apiUrl}/${id}`);
  }
}
```

Nota.service.ts

```
import { Injectable } from '@angular/core';
import { HttpClient } from '@angular/common/http';
import { Observable } from 'rxjs';
import { Nota } from '../models/nota';

@Injectable({
  providedIn: 'root'
})
export class NotaService {
  private apiUrl = 'http://localhost:9090/api/notas';

  constructor(private http: HttpClient) {}

  obtenerTodas(): Observable<Nota[]> {
    return this.http.get<Nota[]>(this.apiUrl);
  }

  obtenerPorEstudianteId(estudianteId: number): Observable<Nota[]> {
    return this.http.get<Nota[]>(`${this.apiUrl}?estudianteId=${estudianteId}`);
  }

  agregarNota(nota: Nota): Observable<Nota> {
```

```
    // Elimina el ID si existe
    const { id, ...notaSinId } = nota; // Si tienes un ID en el objeto, lo
eliminamos

    const notaAEnviar = {
      asignatura: notaSinId.asignatura,
      nota: notaSinId.nota,
      fechaRegistro: notaSinId.fechaRegistro,
      estudiante: {
        id: notaSinId.estudianteId // Asumimos que el objeto `estudiante` ya
tiene el `id`
      }
    };

    console.log('📤 Datos enviados:', JSON.stringify(notaAEnviar));

    return this.http.post<Nota>(this.apiUrl, notaAEnviar);
  }



  actualizarNota(id: number, nota: Nota): Observable<Nota> {
    return this.http.put<Nota>(`${this.apiUrl}/${id}`, nota);
  }

  eliminarNota(id: number): Observable<void> {
    return this.http.delete<void>(`${this.apiUrl}/${id}`);
  }
}
```

App.component.html

```
<h1>{{ title }}</h1>
<app-estudiante></app-estudiante>
```

App.component.ts

```
import { Component } from '@angular/core';
import { CommonModule } from '@angular/common';
import { RouterOutlet } from '@angular/router';
import { EstudianteComponent } from
'./components/estudiante/estudiante.component';

@Component({
  selector: 'app-root',
```

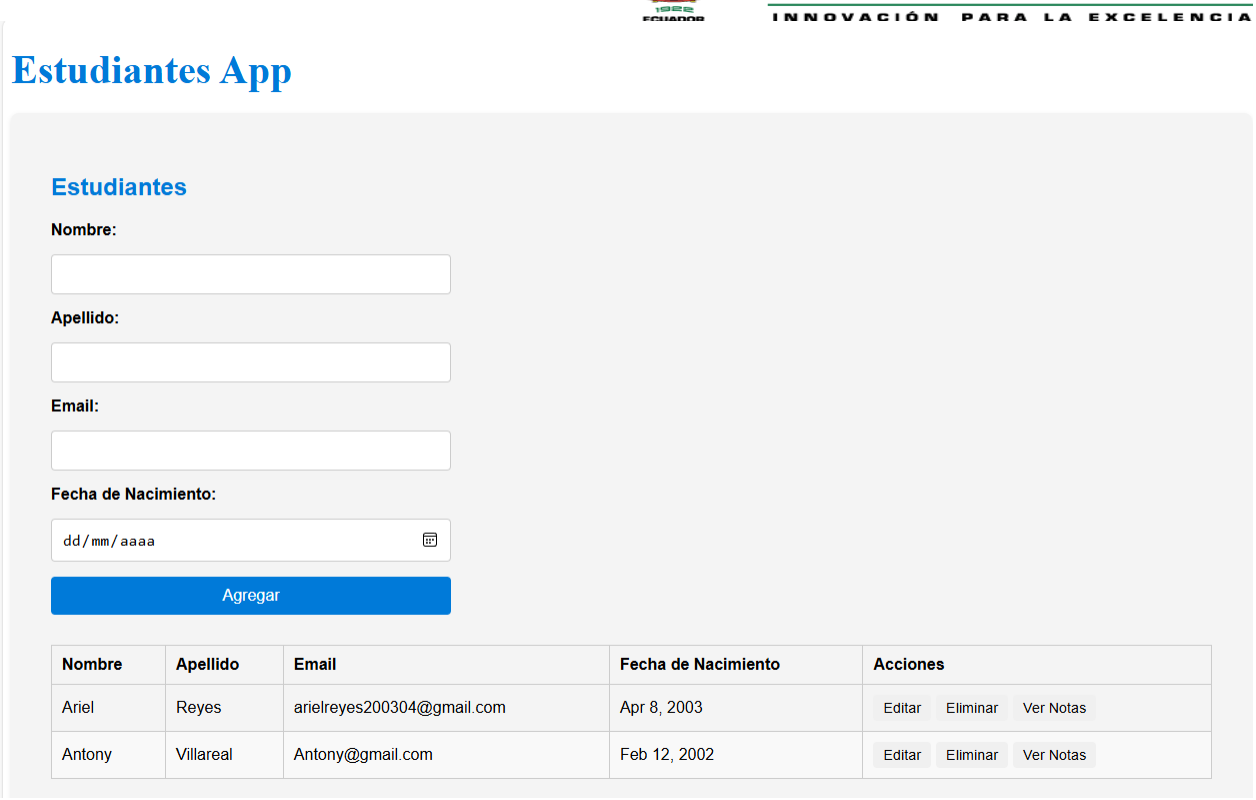
```
  standalone: true,
  imports: [CommonModule, RouterOutlet, EstudianteComponent],
  templateUrl: './app.component.html',
  styleUrls: ['./app.component.css']
})
export class AppComponent {
  title = 'Estudiantes App';
}
```

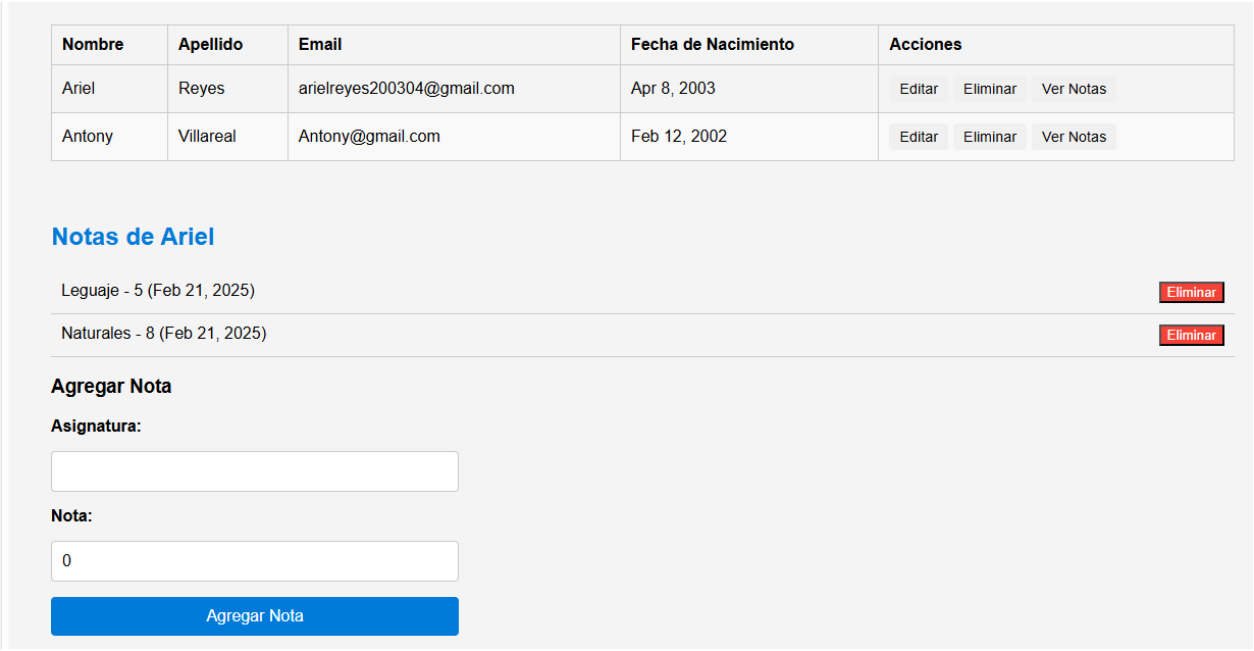
App.config.ts

```
import { ApplicationConfig, provideZoneChangeDetection } from '@angular/core';
import { provideRouter } from '@angular/router';
import { provideHttpClient } from '@angular/common/http';
import { routes } from './app.routes';
import { provideClientHydration, withEventReplay } from '@angular/platform-
browser';

export const appConfig: ApplicationConfig = {
  providers: [
    provideZoneChangeDetection({ eventCoalescing: true }),
    provideRouter(routes),
    provideClientHydration(withEventReplay()),
    provideHttpClient()
  ]
};
```

7. Resultados





8. Conclusiones

El desarrollo de esta aplicación web demuestra que Angular, en combinación con PrimeNG, es una excelente opción para la creación de interfaces modernas y eficientes. La facilidad de integración con APIs REST permite manejar datos de manera dinámica y optimizada.

Los objetivos planteados se cumplieron satisfactoriamente, logrando una aplicación funcional que permite la gestión de estudiantes y sus notas de manera eficiente. La implementación de componentes de PrimeNG mejoró significativamente la presentación visual de la aplicación y la usabilidad para los usuarios finales.

Sin embargo, se identificaron algunas áreas de mejora, como la posibilidad de agregar autenticación de usuarios y la optimización del rendimiento en consultas a la API. Estos aspectos podrían abordarse en futuras versiones del proyecto.

9. Recomendaciones

Para mejorar el sistema, se recomienda implementar autenticación y autorización de usuarios, permitiendo diferentes niveles de acceso según el rol del usuario. Esto garantizaría mayor seguridad y control sobre la información gestionada.

Asimismo, sería beneficioso optimizar las consultas a la API mediante técnicas de paginación y almacenamiento en caché, lo que reduciría la carga en el servidor y mejoraría la velocidad de respuesta de la aplicación.

Finalmente, se sugiere realizar pruebas de usuario con estudiantes y docentes para obtener retroalimentación sobre la experiencia de uso y realizar ajustes en la interfaz según sus necesidades y preferencias.

GitHub: https://github.com/ANTHONYNESTORVILLARREALMACIAS/Desarrollo-Web-Avanzado-AnthonyV-ArielR.git

10. Bibliografía o Referencias

- Angular Documentation. (2025). Getting Started with Angular. Retrieved from https://angular.io/docs
- PrimeNG Official Documentation. (2025). PrimeNG UI Components for Angular. Retrieved from https://www.primefaces.org/primeng/
- Mozilla Developer Network. (2025). Web APIs and Fetch Requests. Retrieved from https://developer.mozilla.org/en-US/