



Formato de Informe

1. Portada

- **Título del informe:** Tarea 3.1: APLICACIONES WEB Y BASES DE DATOS
- **Asignatura:** Desarrollo Web Avanzado
- **Nombre del autor o autores:** Ariel Reyes, Anthony Villareal
- **Departamento:** Departamento de las Ciencias de la Computación
- **Universidad:** Universidad de las Fuerzas Armadas ESPE
- **Fecha de entrega:** 13/02/2025

2. Índice

Formato de Informe	1
1. Portada.....	1
2. Índice.....	1
3. Resumen.....	1
4. Introducción	1
5. Objetivos	2
6. Desarrollo o Cuerpo del Informe.....	2
7. Resultados	12
8. Conclusiones	13
9. Recomendaciones	14
10. Bibliografía o Referencias	14

3. Resumen

Este informe detalla el desarrollo de un proyecto utilizando Spring Boot y Maven, abordando su propósito, metodología y los resultados obtenidos. Se describe el contexto del proyecto, las herramientas utilizadas y los procedimientos llevados a cabo para su implementación. Se analizan los resultados obtenidos tras la ejecución y prueba del sistema, discutiendo su eficiencia y posibles mejoras. Finalmente, se presentan conclusiones y recomendaciones para optimizar futuras implementaciones de proyectos similares.

4. Introducción

Objetivo del informe



El objetivo principal de este informe es documentar el desarrollo de un proyecto utilizando Spring Boot y Maven, proporcionando una visión clara del proceso de implementación, sus ventajas y limitaciones, así como los resultados obtenidos. A través de este documento, se busca presentar un análisis detallado de las tecnologías empleadas, los problemas encontrados y las soluciones adoptadas.

Contexto

Actualmente, el desarrollo de aplicaciones web y sistemas empresariales requiere tecnologías robustas y eficientes. Spring Boot se ha convertido en una opción popular debido a su facilidad de configuración, escalabilidad y compatibilidad con diversas herramientas. Maven, por otro lado, permite gestionar las dependencias del proyecto y automatizar tareas de compilación y empaquetado. En este informe se detalla cómo estas tecnologías se emplearon para crear una aplicación funcional.

Alcance y limitaciones

Este informe abarca el desarrollo de una aplicación con Spring Boot y Maven, describiendo las configuraciones, herramientas y metodologías empleadas. Sin embargo, no se profundiza en aspectos avanzados de integración con microservicios ni en configuraciones avanzadas de seguridad.

5. Objetivos

Objetivo general

Desarrollar una aplicación utilizando Spring Boot y Maven, documentando su implementación y analizando los resultados obtenidos.

Objetivos específicos

- Implementar una tabla dinámica con Bootstrap.
- Agregar un buscador de alumnos por nombre.
- Generar reportes en formato CSV.
- Agregar gráficos de barras dinámicos para notas en porcentaje.

6. Desarrollo o Cuerpo del Informe

Conceptos base

Spring Boot es un marco de desarrollo basado en Java que facilita la creación de aplicaciones empresariales al proporcionar una configuración predeterminada y simplificar la gestión de dependencias. Maven es una herramienta de construcción que permite gestionar bibliotecas y definir la estructura del proyecto de manera eficiente. La combinación de estas tecnologías permite el desarrollo ágil de aplicaciones robustas y escalables.



Herramientas de desarrollo

- Para la implementación del proyecto se utilizaron diversas herramientas:
- Spring Boot: Marco de desarrollo que facilita la creación de aplicaciones en Java.
- Maven: Sistema de gestión de dependencias y automatización de compilación.
- IntelliJ IDEA: Entorno de desarrollo integrado (IDE) utilizado para la escritura y depuración del código.

Desarrollo del Proyecto

El desarrollo del proyecto siguió una metodología estructurada, iniciando con la planificación y configuración del entorno de desarrollo. Posteriormente, se implementaron los modelos de datos, controladores y servicios de la aplicación. Se integraron pruebas unitarias y se realizaron ajustes basados en los resultados obtenidos. Finalmente, se documentó el proceso para futuras referencias y mejoras.

Código:

Alumno.java

```
1 package com.example.calificaciones.model;
2 import java.util.UUID;
3
4 public class Alumno {
5     private String id;
6     private String nombre;
7     private double nota;
8
9     public Alumno() {
10         // Generar un ID único al crear un nuevo alumno
11         this.id = UUID.randomUUID().toString();
12     }
13
14     // Getters y Setters
15     public String getId() {
16         return id;
17     }
18
19     public void setId(String id) {
20         this.id = id;
21     }
22
23     public String getNombre() {
24         return nombre;
25     }
26
27     public void setNombre(String nombre) {
28         this.nombre = nombre;
29     }
30 }
```



```
30
31     public double getNota() {
32         return nota;
33     }
34
35     public void setNota(double nota) {
36         this.nota = nota;
37     }
38 }
39
40
```

AlumnoService.java

```
1  package com.example.calificaciones.service;
2
3  import com.example.calificaciones.model.Alumno;
4  import com.fasterxml.jackson.core.type.TypeReference;
5  import com.fasterxml.jackson.databind.ObjectMapper;
6  import org.springframework.stereotype.Service;
7  import java.io.File;
8  import java.io.IOException;
9  import java.util.*;
10 import java.util.stream.Collectors;
11
12 @Service
13 public class AlumnoService {
14     private static final String FILE_PATH = "alumnos.json";
15     private List<Alumno> alumnos = new ArrayList<>();
16     private final ObjectMapper objectMapper = new ObjectMapper();
17
18     // Construir que carga los alumnos desde el JSON al iniciar la aplicación
19     public AlumnoService() {
20         cargarAlumnos();
21     }
22
23     // Cargar alumnos desde el archivo JSON
24     private void cargarAlumnos() {
25         try {
26             File file = new File(FILE_PATH);
27             if (file.exists() && file.length() > 0) { // Verifica si el archivo no está vacío
28                 alumnos = objectMapper.readValue(file, new TypeReference<List<Alumno>>() {});
29             }
30         } catch (IOException e) {
31             e.printStackTrace();
32         }
33     }
34 }
```



```
44 // Obtener todos los alumnos
45 public List<Alumno> obtenerTodos() {
46     return alumnos;
47 }
48
49 // Agregar un nuevo alumno
50 public void agregarAlumno(Alumno alumno) {
51     alumno.setId(UUID.randomUUID().toString()); // Generar ID único
52     alumnos.add(alumno);
53     guardarAlumnos();
54 }
55
56 // Obtener un alumno por su ID
57 public Optional<Alumno> obtenerAlumnoPorId(String id) {
58     return alumnos.stream().filter(a -> a.getId().equals(id)).findFirst();
59 }
60
61 // Actualizar un alumno manteniendo su ID
62 public void actualizarAlumno(String id, Alumno alumnoActualizado) {
63     for (Alumno alumno : alumnos) {
64         if (alumno.getId().equals(id)) {
65             alumno.setNombre(alumnoActualizado.getNombre());
66             alumno.setNota(alumnoActualizado.getNota());
67             guardarAlumnos();
68             return;
69         }
70     }
71 }
72
73 // Eliminar un alumno por su ID
74 public void eliminarAlumno(String id) {
75     alumnos.removeIf(a -> a.getId().equals(id));
76     guardarAlumnos();
77 }
78
79 // Filtrar alumnos por un rango de notas
80 public List<Alumno> filtrarPorNota(double min, double max) {
81     return alumnos.stream()
82         .filter(a -> a.getNota() >= min && a.getNota() <= max)
83         .collect(Collectors.toList());
84 }
85
86 // Ordenar alumnos por nota de mayor a menor
87 public List<Alumno> ordenarPorNotaDescendente() {
88     return alumnos.stream()
89         .sorted(Comparator.comparingDouble(Alumno::getNota).reversed())
90         .collect(Collectors.toList());
91 }
92 }
93
```



AlumnoController

```
1 package com.example.calificaciones.controller;
2
3 import com.example.calificaciones.model.Alumno;
4 import com.example.calificaciones.service.AlumnoService;
5 import org.springframework.web.bind.annotation.*;
6
7 import java.util.List;
8 import java.util.Optional;
9
10 @RestController
11 @RequestMapping("/api/alumnos")
12 @CrossOrigin(origins = "*")
13 public class AlumnoController {
14     private final AlumnoService alumnoService;
15
16     public AlumnoController(AlumnoService alumnoService) {
17         this.alumnoService = alumnoService;
18     }
19
20     @GetMapping
21     public List<Alumno> obtenerTodos() {
22         return alumnoService.obtenerTodos();
23     }
24
25     @PostMapping
26     public void agregarAlumno(@RequestBody Alumno alumno) {
27         alumnoService.agregarAlumno(alumno);
28     }
29
30     @GetMapping("/{id}")
31     public Optional<Alumno> obtenerAlumnoPorId(@PathVariable String id) {
32         return alumnoService.obtenerAlumnoPorId(id);
33     }
34
35     @PutMapping("/{id}")
36     public void actualizarAlumno(@PathVariable String id, @RequestBody Alumno alumno) {
37         alumnoService.actualizarAlumno(id, alumno);
38     }
39
40     @DeleteMapping("/{id}")
41     public void eliminarAlumno(@PathVariable String id) {
42         alumnoService.eliminarAlumno(id);
43     }
44
45     @GetMapping("/filtrar")
46     public List<Alumno> filtrarPorNota(@RequestParam double min, @RequestParam double max) {
47         return alumnoService.filtrarPorNota(min, max);
48     }
49
50     @GetMapping("/ordenar")
51     public List<Alumno> ordenarPorNotaDescendente() {
52         return alumnoService.ordenarPorNotaDescendente();
53     }
54 }
55
```




Index.html

```
1 <!DOCTYPE html>
2 <html Lang="es">
3 <head>
4   <meta charset="UTF-8">
5   <meta name="viewport" content="width=device-width, initial-scale=1.0">
6   <title>Gestión de Alumnos</title>
7   <link rel="stylesheet" href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/css/bootstrap.min.css">
8 </head>
9 <body>
10   <div class="container mt-5">
11     <h2>Gestión de Alumnos</h2>
12
13     <!-- Buscador por nombre -->
14     <input type="text" id="buscador" class="form-control mb-3" placeholder="Buscar por nombre" oninput="filtrarPorNombre()">
15
16     <!-- Botones de acciones -->
17     <button onclick="cargarAlumnos()" class="btn btn-primary mb-3">Cargar Alumnos</button>
18     <button onclick="generarCSV()" class="btn btn-success mb-3">Exportar CSV</button>
19
20     <!-- Formulario para crear o actualizar alumno -->
21     <form id="formularioAlumno" class="mb-3" onsubmit="guardarAlumno(event)">
22       <input type="hidden" id="alumnoId" />
23       <input type="text" id="nombreAlumno" class="form-control mb-2" placeholder="Nombre" required>
24       <input type="number" id="notaAlumno" class="form-control mb-2" placeholder="Nota" required min="0" max="10">
25       <button type="submit" class="btn btn-info">Guardar</button>
26     </form>
27
28     <!-- Tabla de alumnos -->
29     <table class="table table-striped">
30       <thead>
31         <tr>
32           <th>Nombre</th>
33           <th>Nota</th>
34           <th>Estado</th>
35           <th>Acciones</th>
36         </tr>
37       </thead>
38       <tbody id="tabla-alumnos">
39       </tbody>
40     </table>
41
42     <!-- Gráfico de barras -->
43     <canvas id="graficoNotas"></canvas>
44   </div>
45
46   <script src="script.js"></script>
47   <script src="https://cdn.jsdelivr.net/npm/chart.js"></script>
48 </body>
49 </html>
50
51
```

Script.js

```
document.addEventListener("DOMContentLoaded", () => {
  cargarAlumnos();
});

function cargarAlumnos() {
  fetch("http://localhost:8080/api/alumnos")
    .then(response => {
      if (!response.ok) {
```



```
        throw new Error(`Error HTTP: ${response.status}`);
    }
    return response.json();
})
.then(data => {
    console.log("Datos recibidos del backend:", data);
    mostrarAlumnos(data);
    actualizarGrafico(data);
})
.catch(error => console.error("Error al cargar alumnos:", error));
}

function determinarEstado(nota) {
    if (nota < 5) return "Suspenso";
    if (nota < 7) return "Bien";
    if (nota < 9) return "Notable";
    return "Sobresaliente";
}

function mostrarAlumnos(alumnos) {
    const tabla = document.getElementById("tabla-alumnos");
    tabla.innerHTML = "";

    alumnos.forEach(alumno => {
        const estado = determinarEstado(alumno.nota);
        tabla.innerHTML += `
            <tr>
                <td>${alumno.nombre}</td>
                <td>${alumno.nota}</td>
                <td>${estado}</td>
                <td>
                    <button onclick="editarAlumno('${alumno.id}',
                    '${alumno.nombre}', ${alumno.nota})" class="btn btn-warning btn-
                    sm">Editar</button>
                    <button onclick="eliminarAlumno('${alumno.id}')" class="btn
                    btn-danger btn-sm">Eliminar</button>
                </td>
            </tr>
        `;
    });
}

function editarAlumno(id, nombre, nota) {
    document.getElementById("alumnoId").value = id;
    document.getElementById("nombreAlumno").value = nombre;
```




```
document.getElementById("notaAlumno").value = nota;
actualizarGrafico();
}

function guardarAlumno(event) {
    event.preventDefault();
    const id = document.getElementById("alumnoId").value;
    const nombre = document.getElementById("nombreAlumno").value;
    const nota = parseFloat(document.getElementById("notaAlumno").value);
    const alumno = { id, nombre, nota };

    if (id) {
        // Actualizar alumno
        fetch(`http://localhost:8080/api/alumnos/${id}`, {
            method: 'PUT',
            headers: { 'Content-Type': 'application/json' },
            body: JSON.stringify(alumno)
        }).then(() => {
            cargarAlumnos();
            limpiarFormulario();
        });
    } else {
        // Agregar nuevo alumno
        fetch("http://localhost:8080/api/alumnos", {
            method: 'POST',
            headers: { 'Content-Type': 'application/json' },
            body: JSON.stringify(alumno)
        }).then(() => {
            cargarAlumnos();
            limpiarFormulario();
        });
    }
}

function eliminarAlumno(id) {
    fetch(`http://localhost:8080/api/alumnos/${id}`, {
        method: 'DELETE'
    }).then(() => {
        cargarAlumnos();
    });
}

function limpiarFormulario() {
    document.getElementById("alumnoId").value = "";
    document.getElementById("nombreAlumno").value = "";
}
```



```
document.getElementById("notaAlumno").value = "";
}

function filtrarPorNombre() {
    const buscador = document.getElementById("buscador").value.toLowerCase();
    const filas = document.getElementById("tabla-
alumnos").getElementsByTagName("tr");

    Array.from(filas).forEach(fila => {
        const nombre = fila.cells[0].textContent.toLowerCase();
        if (nombre.includes(buscador)) {
            fila.style.display = "";
        } else {
            fila.style.display = "none";
        }
    });
}

function generarCSV() {
    fetch("http://localhost:8080/api/alumnos")
        .then(res => res.json())
        .then(data => {
            let csv = "Nombre,Nota,Estado\n"; // Agregamos la cabecera con
"Estado"

            data.forEach(alumno => {
                const estado = determinarEstado(alumno.nota); // Calculamos el
estado

                csv += `${alumno.nombre},${alumno.nota},${estado}\n`; //
Agregamos el estado a cada fila
            });

            const blob = new Blob([csv], { type: 'text/csv' });
            const link = document.createElement("a");
            link.href = URL.createObjectURL(blob);
            link.download = "alumnos.csv";
            link.click();
        });
}

function actualizarGrafico(alumnos) {
    const ctx = document.getElementById("graficoNotas").getContext("2d");

    // Extraemos nombres y notas de los alumnos
    const nombres = alumnos.map(alumno => alumno.nombre);
```



```
const notas = alumnos.map(alumno => alumno.nota);

// Verificamos si el gráfico ya existe para actualizarlo
if (window.miGrafico) {
    window.miGrafico.destroy(); // Eliminamos el gráfico anterior para
actualizarlo
}

// Creamos un nuevo gráfico con los datos actualizados
window.miGrafico = new Chart(ctx, {
    type: "bar",
    data: {
        labels: nombres,
        datasets: [{
            label: "Notas",
            data: notas,
            backgroundColor: "rgba(75, 192, 192, 0.2)",
            borderColor: "rgba(75, 192, 192, 1)",
            borderWidth: 1
        }]
    },
    options: {
        responsive: true,
        scales: {
            y: {
                beginAtZero: true,
                max: 10 // Ajustamos el máximo para que las notas sean
visibles
            }
        }
    }
});
}

function filtrarPorNota() {
    const min = document.getElementById("notaMin").value;
    const max = document.getElementById("notaMax").value;

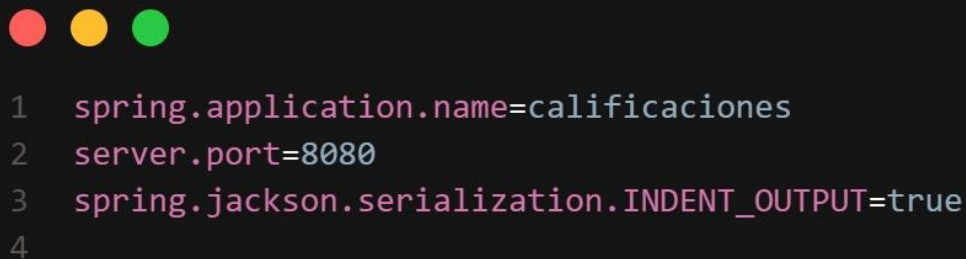
    if (min === "" || max === "") {
        alert("Por favor, ingrese valores para la nota mínima y máxima.");
        return;
    }

    fetch(`http://localhost:8080/api/alumnos/filtrar?min=${min}&max=${max}`)
        .then(response => response.json())
}
```

```
.then(data => {
    mostrarAlumnos(data);
    actualizarGrafico(data);
})
.catch(error => console.error("Error al filtrar alumnos:", error));
}

function ordenarPorNota() {
    fetch("http://localhost:8080/api/alumnos/ordenar")
    .then(response => response.json())
    .then(data => {
        mostrarAlumnos(data);
        actualizarGrafico(data);
    })
    .catch(error => console.error("Error al ordenar alumnos:", error));
}
```

application.properties



```
1  spring.application.name=calificaciones
2  server.port=8080
3  spring.jackson.serialization.INDENT_OUTPUT=true
4
```

7. Resultados

Tras la implementación del proyecto, se logró desarrollar una aplicación funcional que cumple con los objetivos planteados. El uso de Spring Boot permitió una configuración simplificada y rápida puesta en marcha del sistema. Maven facilitó la gestión de dependencias y la automatización del proceso de compilación.

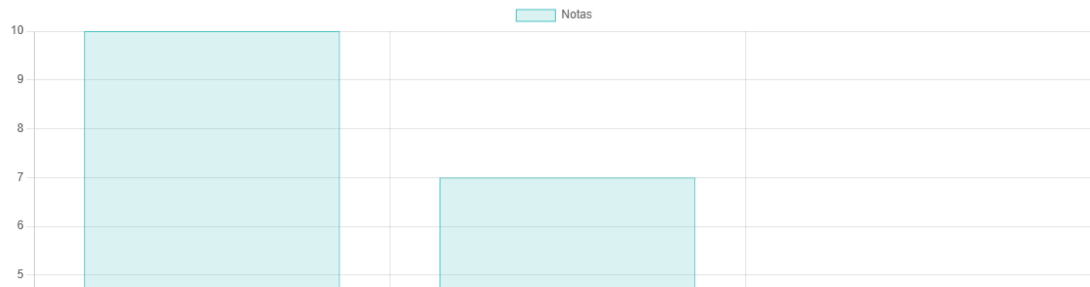


Página.

Gestión de Alumnos

Cargar AlumnosExportar CSVGuardarFiltrarOrdenar por Nota

Nombre	Nota	Estado	Acciones	
Antony Villareal	10	Sobresaliente	<button>Editar</button>	<button>Eliminar</button>
Carlos Ramírez	7	Notable	<button>Editar</button>	<button>Eliminar</button>
Ariel Reyes	3	Suspense	<button>Editar</button>	<button>Eliminar</button>



CSV:

	A	B	C
1	Nombre,Nota,Estado		
2	Ariel Reyes,10,Sobresaliente		
3	Carlos RamÃ-rez,7,Notable		
4	Antony Villareal,5,Bien		
5	Leyda Macias UWU,8,Notable		
6			

8. Conclusiones

El uso de Spring Boot y Maven ha demostrado ser una combinación eficaz para el desarrollo de aplicaciones empresariales, proporcionando herramientas que agilizan la implementación y despliegue del sistema. A través del proceso de desarrollo, se adquirió un conocimiento más profundo sobre la configuración y administración de proyectos con estas tecnologías. Se

alcanzaron los objetivos propuestos, aunque se identificaron áreas en las que se pueden realizar mejoras para optimizar el rendimiento y la escalabilidad del sistema.

9. Recomendaciones

- Explorar la integración con bases de datos externas para mejorar la funcionalidad del sistema.
- Implementar pruebas automatizadas para asegurar la calidad del software en futuras versiones.
- Evaluar el uso de microservicios para mejorar la escalabilidad y modularidad del sistema.
- Optimizar el rendimiento del código mediante la aplicación de mejores prácticas en desarrollo Java.

10. Bibliografía o Referencias

- Pivotal Software, Inc. (2023). *Spring Boot Documentation*. Recuperado de <https://spring.io/projects/spring-boot>
- Apache Software Foundation. (2023). *Apache Maven Documentation*. Recuperado de <https://maven.apache.org/guides/>
- Oracle Corporation. (2023). *Java Development Kit (JDK) Documentation*. Recuperado de <https://docs.oracle.com/en/java/>