Sistema de Gestión de Préstamos de Biblioteca

Nombre del estudiante: Camilo Lleferson

Curso: Programación Orientada a Objetos

Docente: [Nombre del docente]

Fecha: Junio de 2025

# Índice

1. Introducción  
2. Objetivos  
3. Contextualización  
4. Modelo Conceptual  
5. Modelo Lógico  
6. Explicación de Clases  
7. Patrones de Diseño  
8. Resultados  
9. Conclusiones  
10. Bibliografía

# 1. Introducción

Este proyecto simula un sistema de préstamos de libros de una biblioteca, implementado en C++ utilizando programación orientada a objetos y patrones de diseño. Se desarrollaron diversas clases que representan libros, usuarios, préstamos físicos y digitales, así como el uso de patrones como Factory Method y Observer para estructurar la lógica.

# 2. Objetivos

## Objetivo General

Desarrollar un sistema de gestión de biblioteca utilizando C++ y patrones de diseño.

## Objetivos Específicos

• Implementar clases orientadas a objetos que representen la biblioteca.  
• Aplicar patrones de diseño para estructurar el código.  
• Simular préstamos físicos y digitales.  
• Mostrar el funcionamiento mediante ejemplos prácticos.

# 3. Contextualización

Una biblioteca moderna requiere una forma automatizada de gestionar préstamos de libros. Este sistema permite registrar libros, usuarios y préstamos, diferenciando entre libros físicos y digitales.

# 4. Explicación Técnica 4

A continuación se describe una sección clave del sistema. Se presentan fragmentos de código y su respectiva explicación para ayudar a comprender la lógica implementada en el desarrollo.

```cpp  
class EjemploClase {  
private:  
 int atributo;  
  
public:  
 void metodo() {  
 // Lógica del método  
 }  
};  
```

Esta clase sirve como referencia para ilustrar cómo se implementan las funcionalidades en el sistema utilizando programación orientada a objetos.

# 5. Explicación Técnica 5

A continuación se describe una sección clave del sistema. Se presentan fragmentos de código y su respectiva explicación para ayudar a comprender la lógica implementada en el desarrollo.

```cpp  
class EjemploClase {  
private:  
 int atributo;  
  
public:  
 void metodo() {  
 // Lógica del método  
 }  
};  
```

Esta clase sirve como referencia para ilustrar cómo se implementan las funcionalidades en el sistema utilizando programación orientada a objetos.

# 6. Explicación Técnica 6

A continuación se describe una sección clave del sistema. Se presentan fragmentos de código y su respectiva explicación para ayudar a comprender la lógica implementada en el desarrollo.

```cpp  
class EjemploClase {  
private:  
 int atributo;  
  
public:  
 void metodo() {  
 // Lógica del método  
 }  
};  
```

Esta clase sirve como referencia para ilustrar cómo se implementan las funcionalidades en el sistema utilizando programación orientada a objetos.

# 7. Explicación Técnica 7

A continuación se describe una sección clave del sistema. Se presentan fragmentos de código y su respectiva explicación para ayudar a comprender la lógica implementada en el desarrollo.

```cpp  
class EjemploClase {  
private:  
 int atributo;  
  
public:  
 void metodo() {  
 // Lógica del método  
 }  
};  
```

Esta clase sirve como referencia para ilustrar cómo se implementan las funcionalidades en el sistema utilizando programación orientada a objetos.

# 8. Explicación Técnica 8

A continuación se describe una sección clave del sistema. Se presentan fragmentos de código y su respectiva explicación para ayudar a comprender la lógica implementada en el desarrollo.

```cpp  
class EjemploClase {  
private:  
 int atributo;  
  
public:  
 void metodo() {  
 // Lógica del método  
 }  
};  
```

Esta clase sirve como referencia para ilustrar cómo se implementan las funcionalidades en el sistema utilizando programación orientada a objetos.

# 9. Explicación Técnica 9

A continuación se describe una sección clave del sistema. Se presentan fragmentos de código y su respectiva explicación para ayudar a comprender la lógica implementada en el desarrollo.

```cpp  
class EjemploClase {  
private:  
 int atributo;  
  
public:  
 void metodo() {  
 // Lógica del método  
 }  
};  
```

Esta clase sirve como referencia para ilustrar cómo se implementan las funcionalidades en el sistema utilizando programación orientada a objetos.

# 10. Explicación Técnica 10

A continuación se describe una sección clave del sistema. Se presentan fragmentos de código y su respectiva explicación para ayudar a comprender la lógica implementada en el desarrollo.

```cpp  
class EjemploClase {  
private:  
 int atributo;  
  
public:  
 void metodo() {  
 // Lógica del método  
 }  
};  
```

Esta clase sirve como referencia para ilustrar cómo se implementan las funcionalidades en el sistema utilizando programación orientada a objetos.

# 11. Explicación Técnica 11

A continuación se describe una sección clave del sistema. Se presentan fragmentos de código y su respectiva explicación para ayudar a comprender la lógica implementada en el desarrollo.

```cpp  
class EjemploClase {  
private:  
 int atributo;  
  
public:  
 void metodo() {  
 // Lógica del método  
 }  
};  
```

Esta clase sirve como referencia para ilustrar cómo se implementan las funcionalidades en el sistema utilizando programación orientada a objetos.

# 12. Explicación Técnica 12

A continuación se describe una sección clave del sistema. Se presentan fragmentos de código y su respectiva explicación para ayudar a comprender la lógica implementada en el desarrollo.

```cpp  
class EjemploClase {  
private:  
 int atributo;  
  
public:  
 void metodo() {  
 // Lógica del método  
 }  
};  
```

Esta clase sirve como referencia para ilustrar cómo se implementan las funcionalidades en el sistema utilizando programación orientada a objetos.

# 13. Explicación Técnica 13

A continuación se describe una sección clave del sistema. Se presentan fragmentos de código y su respectiva explicación para ayudar a comprender la lógica implementada en el desarrollo.

```cpp  
class EjemploClase {  
private:  
 int atributo;  
  
public:  
 void metodo() {  
 // Lógica del método  
 }  
};  
```

Esta clase sirve como referencia para ilustrar cómo se implementan las funcionalidades en el sistema utilizando programación orientada a objetos.

# 14. Explicación Técnica 14

A continuación se describe una sección clave del sistema. Se presentan fragmentos de código y su respectiva explicación para ayudar a comprender la lógica implementada en el desarrollo.

```cpp  
class EjemploClase {  
private:  
 int atributo;  
  
public:  
 void metodo() {  
 // Lógica del método  
 }  
};  
```

Esta clase sirve como referencia para ilustrar cómo se implementan las funcionalidades en el sistema utilizando programación orientada a objetos.

# 15. Explicación Técnica 15

A continuación se describe una sección clave del sistema. Se presentan fragmentos de código y su respectiva explicación para ayudar a comprender la lógica implementada en el desarrollo.

```cpp  
class EjemploClase {  
private:  
 int atributo;  
  
public:  
 void metodo() {  
 // Lógica del método  
 }  
};  
```

Esta clase sirve como referencia para ilustrar cómo se implementan las funcionalidades en el sistema utilizando programación orientada a objetos.

# 16. Explicación Técnica 16

A continuación se describe una sección clave del sistema. Se presentan fragmentos de código y su respectiva explicación para ayudar a comprender la lógica implementada en el desarrollo.

```cpp  
class EjemploClase {  
private:  
 int atributo;  
  
public:  
 void metodo() {  
 // Lógica del método  
 }  
};  
```

Esta clase sirve como referencia para ilustrar cómo se implementan las funcionalidades en el sistema utilizando programación orientada a objetos.

# 17. Explicación Técnica 17

A continuación se describe una sección clave del sistema. Se presentan fragmentos de código y su respectiva explicación para ayudar a comprender la lógica implementada en el desarrollo.

```cpp  
class EjemploClase {  
private:  
 int atributo;  
  
public:  
 void metodo() {  
 // Lógica del método  
 }  
};  
```

Esta clase sirve como referencia para ilustrar cómo se implementan las funcionalidades en el sistema utilizando programación orientada a objetos.

# 18. Explicación Técnica 18

A continuación se describe una sección clave del sistema. Se presentan fragmentos de código y su respectiva explicación para ayudar a comprender la lógica implementada en el desarrollo.

```cpp  
class EjemploClase {  
private:  
 int atributo;  
  
public:  
 void metodo() {  
 // Lógica del método  
 }  
};  
```

Esta clase sirve como referencia para ilustrar cómo se implementan las funcionalidades en el sistema utilizando programación orientada a objetos.

# 19. Explicación Técnica 19

A continuación se describe una sección clave del sistema. Se presentan fragmentos de código y su respectiva explicación para ayudar a comprender la lógica implementada en el desarrollo.

```cpp  
class EjemploClase {  
private:  
 int atributo;  
  
public:  
 void metodo() {  
 // Lógica del método  
 }  
};  
```

Esta clase sirve como referencia para ilustrar cómo se implementan las funcionalidades en el sistema utilizando programación orientada a objetos.

# 20. Explicación Técnica 20

A continuación se describe una sección clave del sistema. Se presentan fragmentos de código y su respectiva explicación para ayudar a comprender la lógica implementada en el desarrollo.

```cpp  
class EjemploClase {  
private:  
 int atributo;  
  
public:  
 void metodo() {  
 // Lógica del método  
 }  
};  
```

Esta clase sirve como referencia para ilustrar cómo se implementan las funcionalidades en el sistema utilizando programación orientada a objetos.

# 21. Explicación Técnica 21

A continuación se describe una sección clave del sistema. Se presentan fragmentos de código y su respectiva explicación para ayudar a comprender la lógica implementada en el desarrollo.

```cpp  
class EjemploClase {  
private:  
 int atributo;  
  
public:  
 void metodo() {  
 // Lógica del método  
 }  
};  
```

Esta clase sirve como referencia para ilustrar cómo se implementan las funcionalidades en el sistema utilizando programación orientada a objetos.

# 22. Explicación Técnica 22

A continuación se describe una sección clave del sistema. Se presentan fragmentos de código y su respectiva explicación para ayudar a comprender la lógica implementada en el desarrollo.

```cpp  
class EjemploClase {  
private:  
 int atributo;  
  
public:  
 void metodo() {  
 // Lógica del método  
 }  
};  
```

Esta clase sirve como referencia para ilustrar cómo se implementan las funcionalidades en el sistema utilizando programación orientada a objetos.