PROJETO DE SOFTWARE ORIENTADO A OBJETOS PARA GESTÃO DE BIBLIOTECAS EM C++

Objetivos, escopo e benefícios práticos
Gestão de acervo, empréstimos, reservas e usuários com módulos: Catálogo, Usuários, Empréstimos, Multas, Notificações, Persistência, Infraestrutura

APLICAÇÕES TÉCNICAS DE MODELAGEM UML

e implementações em C++ com ênfase em modulação

CONTEXTO DO DOMÍNIO: PROCESSOS E ENTIDADES CHAVE

Resumo técnico dos principais fluxos e integrações da biblioteca

ENTIDADES PRINCIPAIS

Livro/Item, Exemplar, Usuário (Aluno/Professor/Cidadão), Empréstimo, Reserva, Multa.

1

PROCESSOS CENTRAIS

2

Cadastro de acervo, busca/catalogação, empréstimo, devolução, renovação, reserva, cobrança de multa.

REGRAS DE NEGÓCIO

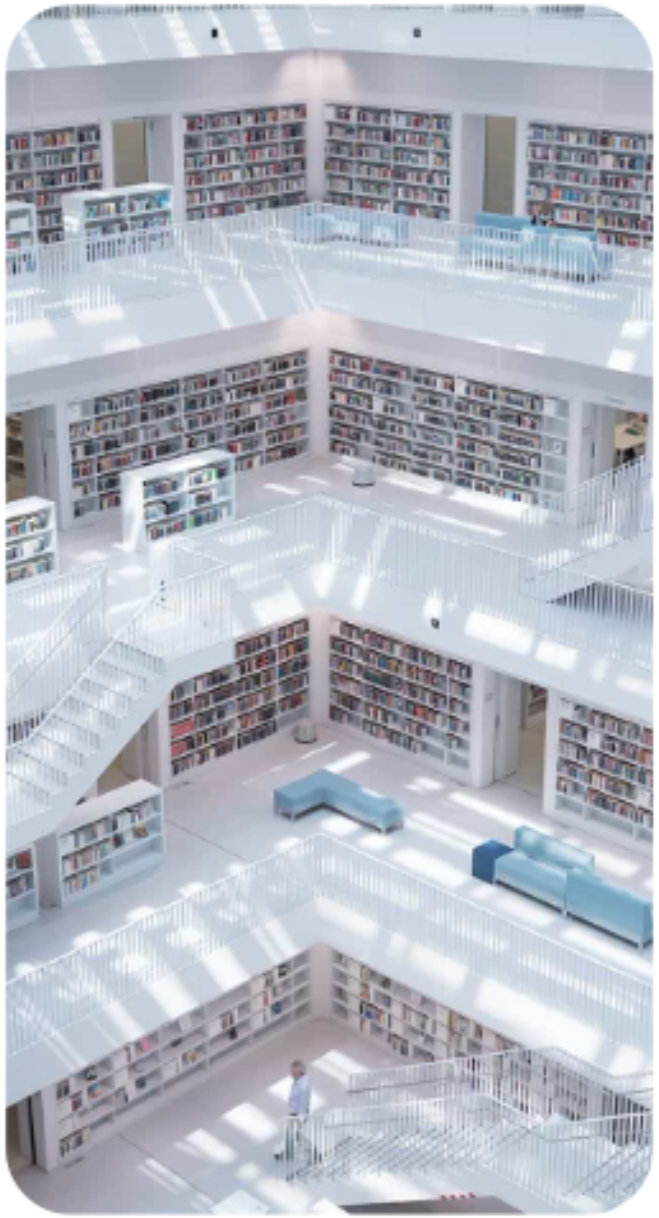
Limites por usuário, prazos por tipo de material, prioridades de reserva, penalidades por atraso.

3

INTEGRAÇÕES EXTERNAS

4

Notificações email/SMS, catálogo externo ISBN, sistema de autenticação institucional.



REQUISITOS ESSENCIAIS PARA SISTEMA DE BIBLIOTECA

Funcionais e não funcionais para arquitetura modular em C++

REQUISITOS FUNCIONAIS

- **GERENCIAR CATÁLOGO: CRUD DE LIVROS E EXEMPLARES**
- **EMPRÉSTIMOS: CRIAR, RENOVAR, DEVOLVER**
- **RESERVAS: FILA, PRIORIDADE, NOTIFICAÇÃO**
- **USUÁRIOS: AUTENTICAÇÃO, PERFIS, LIMITES**
- **MULTAS: CÁLCULO, REGISTRO, QUITAÇÃO**

REQUISITOS NÃO FUNCIONAIS

- **PERFORMANCE: RESPOSTAS UNDER 200MS PARA BUSCAS EM CATÁLOGO**
- **SEGURANÇA: AUTENTICAÇÃO E PROTEÇÃO CONTRA INJEÇÃO E ACESSO INDEVIDO**
- **ESCALABILIDADE: MODULARIDADE PARA SUPORTAR CATALOGAÇÕES MAIORES**
- **MANUTENIBILIDADE: BAIXO ACOPLAMENTO, ALTA COESÃO POR MÓDULO**

MODELAGEM DE DOMÍNIO OO PARA GESTÃO DE BIBLIOTECA

Classes, agregações e responsabilidades para implementação modular em C++

1 CLASSES NÚCLEO

- Livro: metadados ISBN, título, autor, categorias
- Exemplar: ID, estado, localização física
- Usuário: id, tipo, histórico de empréstimos
- Empréstimo: data saída, data prevista, status

2 RELACIONAMENTOS

- Agregação: Livro contém Exemplares
- Composição: Usuário possui Histórico de Empréstimos
- Associação: Reserva liga Usuário to Exemplar desejado

3 RESPONSABILIDADES

- Catálogo: indexação e busca eficiente, encapsular índices
- Gerência de prazos: cálculo de multas e regras por tipo
- Notificações: desacopladas via Observer and Events

ARQUITETURA MODULAR: SEPARAR RESPONSABILIDADES POR CAMADAS

Camadas claras e módulos independentes para um sistema de biblioteca em C++

APRESENTAÇÃO

CLI, GUI ou API REST que invoca casos de uso do domínio

1

2

APLICAÇÃO

Orquestra casos de uso, validações e transações

DOMÍNIO

Entidades, regras e serviços de domínio puro sem infraestrutura

3

4

INFRAESTRUTURA

Persistência, mensagens e integrações externas via adaptadores

MÓDULOS

Catálogo, Usuários, Empréstimos, Notificações como limites claros

5

APLICANDO SOLID E PADRÕES EM DOMÍNIO DE BIBLIOTECA

Arquitetura C++ orientada a modularidade e extensibilidade

Princípio	Aplicação no domínio
Responsabilidade única	Classes separadas: cálculo de multas isolado de persistência
Aberto/Fechado	Extensão via Strategy para políticas de empréstimo
Inversão de dependência	Repositórios dependem de interfaces, não de implementações
Padrões recomendados	Repository para acesso a dados; Factory para criação de objetos complexos
Observador	Notificações de vencimento via Observer desacoplado do domínio

PERSISTÊNCIA E INFRAESTRUTURA: REPOSITÓRIOS E SERIALIZAÇÃO

Arquitetura C++ modular para gestão de biblioteca

ESTRATÉGIA DE PERSISTÊNCIA

Interfaces de repositório; implementações SQLite, JSON e mocks para testes.

MAPEAMENTO E SERIALIZAÇÃO

Serializers para import/export JSON e CSV; uso de std::filesystem e nlohmann::json.

TRANSAÇÕES E CONSISTÊNCIA

Unidade de trabalho na camada de aplicação para operações atômicas.

GESTÃO DE ERROS

Exceções controladas, códigos de erro e logs estruturados para auditoria.

INFRA PARA NOTIFICAÇÕES

Adaptadores SMTP e HTTP; filas simples para retry de envio.

CASOS DE USO END-TO-END: FLUXO DE EMPRÉSTIMO E RESERVA

Comparação entre arquitetura monolítica e modular para gestão de biblioteca

ANTES (MONOLÍTICO)

- Busca, empréstimo e persistência fortemente acoplados
- Alteração no banco de dados exige recompilar grande parte do sistema
- Testes integrados complexos e lentos

DEPOIS (MODULAR)

- Fluxo: Controlador para Serviço de Aplicativo para domínio para repositório.
- Substituição de repositório sem impacto no domínio
- Testes unitários e de integração rápidos; deploy incremental

QUALIDADE, TESTES E ROADMAP: MÉTRICAS E PRÓXIMOS PASSOS

Implementação em C++ com foco em modularidade e entregas rápidas

T+0

- Configurar repositório e CI
- Estilo e análise: clang-format, clang-tidy; CI: GitHub Actions

T+2 SEMANAS

- Modelo de domínio e testes unitários
- Implementar domínio e testes com GoogleTest

T+6 SEMANAS

- Adaptação e testes de integração
- Adaptação de persistência com SQLite e testes de integração