



Gabriela Vieira dos Santos Vaz – BP3044513

Jonas Ribeiro da Rosa – BP304422X

Vinícius Pereira Costa – BP3044289

Iago Felipe Steigleder Bacci – BP3044246

## **Projeto Back-end & Projeto de Componentes**

Trabalho apresentado ao Curso de Análise e Desenvolvimento de Sistemas do Instituto Federal de Educação, Ciência e Tecnologia de São Paulo Campus Bragança Paulista, como requisito das disciplinas de Desenvolvimento de Componentes e Desenvolvimento Web Back-End sob orientação do Prof. Luiz Gustavo Diniz de Oliveira Vêras.

**Novembro, 2025**

# Documento de Especificação da API

## 1. INFORMAÇÕES GERAIS

### 1.1 Nome da Aplicação

Sistema de Gerenciamento de Livraria (Bookstore Platform)

### 1.2 Integrantes do Grupo

- Gabriela Vieira dos Santos Vaz – BP3044513
- Jonas Ribeiro da Rosa – BP304422X
- Vinícius Pereira Costa – BP3044289
- Iago Felipe Steigleder Bacci – BP3044246

## 2. TEMA DO PROJETO

### 2.1 Descrição do Tema

O projeto consiste no desenvolvimento de uma API REST para gerenciamento de uma livraria, permitindo o controle e administração de livros, editoras, clientes e pedidos realizados pelos usuários.

O sistema foi planejado para simular um ambiente real de e-commerce de livraria, sendo capaz de:

- Cadastrar clientes
- Cadastrar editoras e livros
- Associar livros a autores e categorias
- Criar pedidos de compra
- Gerenciar itens dentro de pedidos
- Consultar informações por meio de endpoints REST
- Aplicar arquitetura baseada em componentes e boas práticas de projeto

## 3. ARQUITETURA DO SISTEMA

### 3.1 Tipo de Arquitetura

O sistema foi construído seguindo os princípios de Clean Architecture e Domain-Driven Design (DDD), dividido em módulos independentes (Componentes) que são integrados pela aplicação principal.

```
bookstore-platform/  
├─ book-domain/      # Componente de Gestão de Livros (Catálogo/Estoque)  
├─ customer-domain/  # Componente de Gestão de Clientes  
├─ order-domain/     # Componente de Gestão de Pedidos (Core Business)  
├─ common-domain/    # Componente de Infraestrutura e Serviços Compartilhados (Frete/Email)  
├─ api-rest/         # Aplicação Spring Boot (API Gateway/Controller)  
└─ pom.xml           # Parent POM (Gerenciamento de Dependências)
```

### 3.2 Tecnologias Utilizadas

- **Linguagem:** Java 17
- **Framework:** Spring Boot 3.1.5
- **Persistência:** Spring Data JPA / Hibernate
- **Banco de Dados:**
  - Produção: MySQL 8.0 (Driver 8.4.0)
  - Testes: H2 Database (Em memória)
- **Serviços e Integrações Externas:**
  - ViaCEP: Utilizado para consulta de endereços e cálculo lógico de frete por região.
  - MailHog: Ambiente local para captura e inspeção de e-mails enviados.
  - Spring Mail: Estrutura configurada para envio de notificações, como alertas de estoque.
- **Testes:** JUnit 5 (Jupiter), Mockito, AssertJ
- **Build:** Maven
- **Documentação:** SpringDoc OpenAPI (Swagger UI)

## 4. ENTIDADES DO SISTEMA

### 4.1 Book-Domain (Catálogo)

Responsável pelas regras de negócio dos produtos.

- **Entidades:** Livro (Abstrata), Autor, Editora, Categoria, Funcionário.
- **Padrões Aplicados:**
  - **Polimorfismo/Template Method (RN02):** Cálculo de preço dinâmico nas subclasses LivroCapaDura, LivroBrochura e LivroDigital.
  - **Rich Domain Model:** Lógica de validação de estoque (decrementarEstoque, verificarEstoqueMinimo) encapsulada na entidade.
- **Funcionalidades:** CRUD de Livros, Baixa de Estoque.

#### 4.1.1 Livro

Representa os livros disponíveis para venda.

Atributos:

- ID
- Título
- Preço
- ISBN
- Status
- Quantidade
- Resumo
- Ano Publicação
- Número de Páginas
- Editora
- Categoria
- Autores

#### 4.1.2 Editora

Empresa responsável pela publicação dos livros.

Atributos:

- ID
- Nome
- CNPJ

- Telefone
- Email

#### 4.1.3 Funcionário

Atributos:

- ID
- Nome
- Matrícula

#### 4.1.4 Categoria

Atributos:

- ID
- Nome

#### 4.1.5 Autor

Atributos:

- ID
- Nome
- Data de Nascimento
- Nacionalidade

### 5.1 Customer-Domain (Cliente)

Responsável pela gestão dos usuários.

- **Entidades:** Cliente (Aggregate Root), Endereço.
- **Funcionalidades:** Cadastro com validação de unicidade (CPF/Email), Busca por Email.

#### 5.1.1 Cliente

Representa o usuário que efetua compras na livraria.

Atributos principais:

- ID
- Nome
- CPF
- Data Nascimento
- Email

- Telefone
- Endereço
- Senha

### 5.1.2 Endereço

Atributos principais:

- ID
- CEP
- Rua
- Número
- Complemento
- Bairro
- Cidade
- Estado

## 6.1 Order-Domain (Pedidos)

Responsável pelas compras efetuadas pelos clientes.

- **Entidades:** Pedido, ItemPedido.
- **Padrões Aplicados:**
  - **Strategy Pattern (RN04):** Hierarquia Pagamento → PagamentoPix (8% desconto) e PagamentoCartao (3% à vista).
  - **Factory Pattern:** Classe PagamentoFactory para decidir qual estratégia de pagamento instanciar.
  - **Snapshot Pattern:** ItemPedido congela o preço do livro no momento da compra.
- **Funcionalidades:** Efetuar Pedido (Transacional), Cálculo de Total, Histórico.

### 6.1.1 Pedido

Representa uma compra realizada por um cliente.

Atributos principais:

- ID
- Data do Pedido
- Status
- Valor Total
- Valor Frete
- Prazo Entrega

- Endereço Entrega

### 6.1.2 Item Pedido

Atributos principais:

- ID
- Quantidade
- Preço Unitário

## 5. RELACIONAMENTO ENTRE ENTIDADES

O sistema foi modelado com base em relacionamentos entre entidades que representam com fidelidade o funcionamento real de uma livraria:

Entidade A	Relacionamento	Entidade B
Cliente	1:N	Pedido
Pedido	1:N	ItemPedido
Livro	N:M	Autor
Livro	N:M	Categoria
Livro	1:N	Editora

- **Cliente 1:N Pedido**

- Indica que um cliente pode realizar vários pedidos, porém cada pedido pertence a apenas um cliente.

- **Pedido 1:N ItemPedido**

- Indica que um pedido pode conter vários itens, e cada item está associado a um único pedido.

- **Livro M:N Autor**

- Indica que um livro pode ter vários autores, e vários autores podem ter escrito vários livros.

- **Livro M:N Categoria**

- Indica que um livro pode estar em várias categorias, e uma categoria pode agrupar vários livros.

- **Livro 1:N Editora**

- Indica que um livro pode pertencer a apenas uma editora, e uma editora pode estar associada a vários livros.

## 6. DOCUMENTAÇÃO DA API

- **Cliente-Controller**

Verbo HTTP	Path	Body de Requisição	Body de Retorno	Status Sucesso	Status Erro	Observações
GET	/api/clientes	—	Lista de clientes	200 OK	500	Retorna todos os clientes
GET	/api/clientes/{id}	—	Cliente	200 OK	404	Busca por ID
GET	/api/clientes/email/{email}	—	Cliente	200 OK	404	Busca por e-mail
POST	/api/clientes	JSON com dados do cliente	Cliente criado	201 Created	400	Criação
PUT	/api/clientes/{id}	JSON atualizado	Cliente	200 OK	404	Atualização
DELETE	/api/clientes/{id}	—	—	204 No Content	404	Exclusão

- **Livro-Controller**

Verbo HTTP	Path	Body de Requisição	Body de Retorno	Status Sucesso	Status Erro
GET	/api/livros	—	Lista de livros	200	500
GET	/api/livros/{id}	—	Livro detalhado	200	404
POST	/api/livros	JSON	Livro	201	400
PUT	/api/livros/{id}	JSON	Livro	200	404
DELETE	/api/livros/{id}	—	—	204	404



- **Editora-Controller**

Verbo HTTP	Path	Body de Requisição	Body de Retorno	Status Sucesso	Status Erro
GET	/api/editoras	—	Lista de editoras	200	500
GET	/api/editoras/{id}	—	Editora	200	404
POST	/api/editoras	JSON	Editora	201	400
PUT	/api/editoras/{id}	JSON	Editora	200	404
DELETE	/api/editoras/{id}	—	—	204	404

- **Pedido-Controller**

Verbo HTTP	Path	Body de Requisição	Body de Retorno	Status HTTP de Sucesso	Status HTTP de Erro	Observações
GET	/api/pedidos	—	Lista de pedidos	200 OK	500	Retorna todos os pedidos
GET	/api/pedidos/{id}	—	Pedido	200 OK	404	Busca por ID
GET	/api/pedidos/cliente/{clienteId}	—	Lista	200 OK	404	Pedidos de um cliente
POST	/api/pedidos	JSON do pedido	Pedido criado	201 Created	400	Cria pedido
PUT	/api/pedidos/{id}	JSON atualizado	Pedido	200 OK	404	Atualiza
DELETE	/api/pedidos/{id}	—	—	204 No Content	404	Remove

## 7. Implantação da API com NGINX

A implantação da API foi realizada utilizando duas instâncias do projeto Spring Boot e um servidor NGINX configurado como reverse proxy para habilitar o balanceamento de carga.

As instâncias da API foram iniciadas com o comando:

- `mvn spring-boot:run`

Executadas nos diretórios correspondentes:

- `C:\Users\User\bookstore-platform\api-rest\`

Duas imagens foram registradas mostrando:

- As portas **8081** e **8082** sendo iniciadas no terminal
- Ambas as instâncias acessíveis no navegador, confirmando o funcionamento

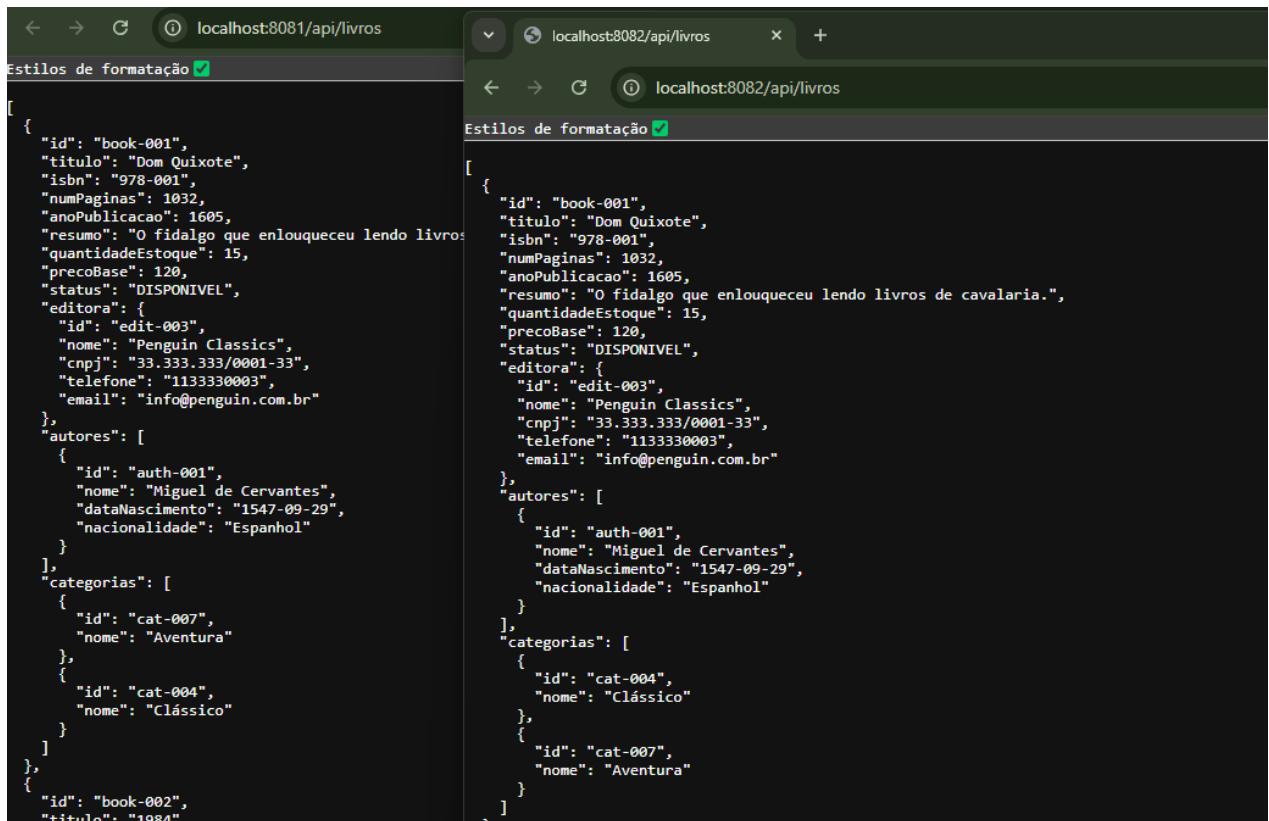
```
C:\Users\User\bookstore-platform\api-rest>mvn spring-boot:run -Dspring-boot.run.arguments="--server.port=8081"
[INFO] Scanning for projects...
[WARNING]
[WARNING] Some problems were encountered while building the effective model for rosa.ribeiro.jonas:api-rest:jar:1.0-SNAPSHOT
[WARNING] 'build.plugins.plugin.version' for org.springframework.boot:spring-boot-maven-plugin is missing. @ line 63, column 21
[WARNING]
[WARNING] It is highly recommended to fix these problems because they threaten the stability of your build.
[WARNING]
[WARNING] For this reason, future Maven versions might no longer support building such malformed projects.
[WARNING]
[INFO]
[INFO] -----< rosa.ribeiro.jonas:api-rest >-----
[INFO] Building api-rest 1.0-SNAPSHOT
[INFO] from pom.xml
[INFO] -----[ jar ]-----
[INFO]
[INFO] >>> spring-boot:4.0.0:run (default-cli) > test-compile @ api-rest >>>
[INFO]
[INFO] --- resources:3.3.1:resources (default-resources) @ api-rest ---
[INFO] Copying 2 resources from src\main\resources to target\classes
[INFO]
[INFO] --- compiler:3.13.0:compile (default-compile) @ api-rest ---
[INFO] Nothing to compile - all classes are up to date.
[INFO]
[INFO] --- resources:3.3.1:testResources (default-testResources) @ api-rest ---
[INFO] skip non existing resourceDirectory C:\Users\User\bookstore-platform\api-rest\src\test\resources
[INFO]
[INFO] --- compiler:3.13.0:testCompile (default-testCompile) @ api-rest ---
[INFO] No sources to compile
[INFO]
[INFO] <<< spring-boot:4.0.0:run (default-cli) < test-compile @ api-rest <<<
[INFO]
[INFO]
[INFO] --- spring-boot:4.0.0:run (default-cli) @ api-rest ---
[INFO] Attaching agents: []
```

```
PROBLEMAS 46 SAÍDA CONSOLE DE DEBURAÇÃO TERMINAL PORTAS
2025-11-29T22:24:52.837-03:00 INFO 19088 --- [main] o.apache.catalina.core.StandardEngine : Starting Servlet engine: [Apache Tomcat/10.1.15]
2025-11-29T22:24:52.935-03:00 INFO 19088 --- [main] o.a.c.c.C.[Tomcat].[localhost].[/] : Initializing Spring embedded WebApplicationContext
2025-11-29T22:24:52.935-03:00 INFO 19088 --- [main] w.s.c.ServletWebServerApplicationContext : Root WebApplicationContext: initialization completed in 1586 ms
2025-11-29T22:24:53.086-03:00 INFO 19088 --- [main] o.hibernate.jpa.internal.util.LogHelper : ###000204: Processing PersistenceUnitInfo [name: default]
2025-11-29T22:24:53.134-03:00 INFO 19088 --- [main] org.hibernate.Version : ###000412: Hibernate ORM core version 6.2.13.Final
2025-11-29T22:24:53.134-03:00 INFO 19088 --- [main] org.hibernate.cfg.Environment : ###000406: Using bytecode reflection optimizer
2025-11-29T22:24:53.380-03:00 INFO 19088 --- [main] o.s.o.j.p.SpringPersistenceUnitInfo : No LoadTimeWeaver setup: ignoring JPA class transformer
2025-11-29T22:24:53.412-03:00 INFO 19088 --- [main] com.zaxxer.hikari.HikariDataSource : HikariPool-1 - Starting...
2025-11-29T22:24:53.653-03:00 INFO 19088 --- [main] com.zaxxer.hikari.pool.HikariPool : HikariPool-1 - Added connection com.mysql.cj.jdbc.ConnectionImpl@54463380
2025-11-29T22:24:53.653-03:00 INFO 19088 --- [main] com.zaxxer.hikari.HikariDataSource : HikariPool-1 - Start completed.
2025-11-29T22:24:53.720-03:00 WARN 19088 --- [main] org.hibernate.orm.deprecation : ###0000025: MySQLDialect does not need to be specified explicitly using 'hibernate.dialect' (remove the property setting and it will be selected by default)
2025-11-29T22:24:54.769-03:00 INFO 19088 --- [main] o.h.e.t.j.p.i.JtaPlatformInitiator : ###000489: No JTA platform available (set 'hibernate.transaction.jta.platform' to enable JTA platform integration)
2025-11-29T22:24:54.986-03:00 INFO 19088 --- [main] j.LocalContainerEntityManagerFactoryBean : Initialized JPA EntityManagerFactory for persistence unit 'default'
2025-11-29T22:24:55.222-03:00 WARN 19088 --- [main] ocalVariableTableParameterNameDiscoverer : Using deprecated '-debug' fallback for parameter name resolution. Com
pile the affected code with '-parameters' instead or avoid its introspection: rosa.ribeiro.jonas.customerdomain.repository.ClienteRepository
2025-11-29T22:24:55.394-03:00 WARN 19088 --- [main] ocalVariableTableParameterNameDiscoverer : Using deprecated '-debug' fallback for parameter name resolution. Com
pile the affected code with '-parameters' instead or avoid its introspection: rosa.ribeiro.jonas.bookdomain.repository.LivroRepository
2025-11-29T22:24:55.422-03:00 INFO 19088 --- [main] o.s.d.j.r.query.QueryEnhancerFactory : Hibernate is in classpath; If applicable, HQL parser will be used.
2025-11-29T22:24:55.863-03:00 WARN 19088 --- [main] ocalVariableTableParameterNameDiscoverer : Using deprecated '-debug' fallback for parameter name resolution. Com
pile the affected code with '-parameters' instead or avoid its introspection: rosa.ribeiro.jonas.orderdomain.repository.PedidoRepository
2025-11-29T22:24:56.071-03:00 WARN 19088 --- [main] JpaBaseConfiguration$JpaWebConfiguration : spring.jpa.open-in-view is enabled by default. Therefore, database qu
eries may be performed during view rendering. Explicitly configure spring.jpa.open-in-view to disable this warning
2025-11-29T22:24:56.463-03:00 INFO 19088 --- [main] o.s.b.w.embedded.tomcat.TomcatWebServer : Tomcat started on port(s): 8081 (http) with context path ''
2025-11-29T22:24:56.469-03:00 INFO 19088 --- [main] r.r.j.a.bookdomain.BookstoreApplication : Started BookstoreApplication in 5.555 seconds (process running for 5.967)
[]
```

```
C:\Users\User\bookstore-platform\api-rest>mvn spring-boot:run -Dspring-boot.run.arguments="--server.port=8082"
[INFO] Scanning for projects...
[WARNING]
[WARNING] Some problems were encountered while building the effective model for rosa.ribeiro.jonas:api-rest:jar:1.0-SNAPSHOT
[WARNING] 'build.plugins.plugin.version' for org.springframework.boot:spring-boot-maven-plugin is missing. @ line 63, column 21
[WARNING]
[WARNING] It is highly recommended to fix these problems because they threaten the stability of your build.
[WARNING]
[WARNING] For this reason, future Maven versions might no longer support building such malformed projects.
[WARNING]
[INFO]
[INFO] -----< rosa.ribeiro.jonas:api-rest >-----
[INFO] Building api-rest 1.0-SNAPSHOT
[INFO] from pom.xml
[INFO] -----[ jar]-----
[INFO]
[INFO] >>> spring-boot:4.0.0:run (default-cli) > test-compile @ api-rest >>>
[INFO]
[INFO] --- resources:3.3.1:resources (default-resources) @ api-rest ---
[INFO] Copying 2 resources from src/main/resources to target/classes
[INFO]
[INFO] --- compiler:3.13.0:compile (default-compile) @ api-rest ---
[INFO] Nothing to compile - all classes are up to date.
[INFO]
[INFO] --- resources:3.3.1:testResources (default-testResources) @ api-rest ---
[INFO] skip non existing resourceDirectory C:\Users\User\bookstore-platform\api-rest\src\test\resources
[INFO]
[INFO] --- compiler:3.13.0:testCompile (default-testCompile) @ api-rest ---
[INFO] No sources to compile
[INFO]
[INFO] <<< spring-boot:4.0.0:run (default-cli) < test-compile @ api-rest <<<
[INFO]
[INFO]
[INFO] --- spring-boot:4.0.0:run (default-cli) @ api-rest ---
[INFO] Attaching agents: []
```

```
PROBLEMAS 46 SAÍDA CONSOLE DE DEBURAÇÃO TERMINAL PORTAS
2025-11-29T22:21:43.694-03:00 INFO 33828 --- [main] org.hibernate.cfg.Environment : ###000406: Using bytecode reflection optimizer
2025-11-29T22:21:43.954-03:00 INFO 33828 --- [main] o.s.o.j.p.SpringPersistenceUnitInfo : No LoadTimeWeaver setup: ignoring JPA class transformer
2025-11-29T22:21:43.980-03:00 INFO 33828 --- [main] com.zaxxer.hikari.HikariDataSource : HikariPool-1 - Starting...
2025-11-29T22:21:44.183-03:00 INFO 33828 --- [main] com.zaxxer.hikari.pool.HikariPool : HikariPool-1 - Added connection com.mysql.cj.jdbc.ConnectionImpl@7b7068d8
2025-11-29T22:21:44.184-03:00 INFO 33828 --- [main] com.zaxxer.hikari.HikariDataSource : HikariPool-1 - Start completed.
2025-11-29T22:21:44.303-03:00 WARN 33828 --- [main] org.hibernate.orm.deprecation : ###0000025: MySQLDialect does not need to be specified explicitly using 'hibernate.dialect' (remove the property setting and it will be selected by default)
2025-11-29T22:21:45.407-03:00 INFO 33828 --- [main] o.h.e.t.j.p.i.JtaPlatformInitiator : ###000489: No JTA platform available (set 'hibernate.transaction.jta.platform' to enable JTA platform integration)
2025-11-29T22:21:45.725-03:00 INFO 33828 --- [main] j.LocalContainerEntityManagerFactoryBean : Initialized JPA EntityManagerFactory for persistence unit 'default'
2025-11-29T22:21:45.966-03:00 WARN 33828 --- [main] ocalVariableTableParameterNameDiscoverer : Using deprecated '-debug' fallback for parameter name resolution. Com
pile the affected code with '-parameters' instead or avoid its introspection: rosa.ribeiro.jonas.customerdomain.repository.ClienteRepository
2025-11-29T22:21:46.133-03:00 WARN 33828 --- [main] ocalVariableTableParameterNameDiscoverer : Using deprecated '-debug' fallback for parameter name resolution. Com
pile the affected code with '-parameters' instead or avoid its introspection: rosa.ribeiro.jonas.bookdomain.repository.LivroRepository
2025-11-29T22:21:46.155-03:00 INFO 33828 --- [main] o.s.d.j.r.query.QueryEnhancerFactory : Hibernate is in classpath; If applicable, HQL parser will be used.
2025-11-29T22:21:46.607-03:00 WARN 33828 --- [main] ocalVariableTableParameterNameDiscoverer : Using deprecated '-debug' fallback for parameter name resolution. Com
pile the affected code with '-parameters' instead or avoid its introspection: rosa.ribeiro.jonas.orderdomain.repository.PedidoRepository
2025-11-29T22:21:46.792-03:00 WARN 33828 --- [main] JpaBaseConfiguration$JpaWebConfiguration : spring.jpa.open-in-view is enabled by default. Therefore, database qu
eries may be performed during view rendering. Explicitly configure spring.jpa.open-in-view to disable this warning
2025-11-29T22:21:47.304-03:00 INFO 33828 --- [main] o.s.b.w.embedded.tomcat.TomcatWebServer : Tomcat started on port(s): 8082 (http) with context path ''
2025-11-29T22:21:47.326-03:00 INFO 33828 --- [main] r.r.j.a.bookdomain.BookstoreApplication : Started BookstoreApplication in 5.922 seconds (process running for 6.368)
[]
```

*Dois backends Spring Boot executando nas portas 8081 e 8082 simultaneamente.*



Ambos os serviços foram testados diretamente via navegador nas portas 8081 e 8082, retornando corretamente a lista de livros.

Na sequência, foi inserida a configuração utilizada no arquivo **nginx.conf**, acompanhada de imagens demonstrando:

- O upstream configurado para distribuir as requisições
- O NGINX em execução
- As portas 8081, 8082 e 8080 ativas simultaneamente (via netstat)
- O endpoint funcionando através da porta 8080 no navegador
- <http://localhost:8080/api/livros>

Com isso, o balanceamento de carga foi implementado com sucesso.

```
nginx.conf X
conf > nginx.conf
1  worker_processes 1;
2
3  events {
4      worker_connections 1024;
5  }
6
7  http {
8      upstream backend {
9          server 127.0.0.1:8081;
10         server 127.0.0.1:8082;
11     }
12
13     server {
14         listen 8080;
15
16         location / {
17             proxy_pass http://backend;
18             proxy_set_header Host $host;
19             proxy_set_header X-Real-IP $remote_addr;
20         }
21     }
22 }
```

```
C:\nginx>netstat -ano | findstr ":8080"
TCP    0.0.0.0:8080      0.0.0.0:0        LISTENING      20484
```

```
C:\nginx>netstat -ano | findstr ":8081"
TCP    0.0.0.0:8081      0.0.0.0:0        LISTENING      19088
TCP    [::]:8081        [::]:0           LISTENING      19088
TCP    [::1]:62474      [::1]:8081       TIME_WAIT      0
TCP    [::1]:63829      [::1]:8081       TIME_WAIT      0
```

```
C:\nginx>netstat -ano | findstr ":8082"
TCP    0.0.0.0:8082      0.0.0.0:0        LISTENING      33828
TCP    [::]:8082        [::]:0           LISTENING      33828
TCP    [::1]:57842      [::1]:8082       TIME_WAIT      0
TCP    [::1]:59517      [::1]:8082       TIME_WAIT      0
```

NGINX executando na porta 8080 e aplicações Spring Boot nas portas 8081 e 8082.

```
C:\nginx>nginx -s reload

C:\nginx>curl http://localhost:8080/api/livros
[{"id":"book-001","titulo":"Dom Quixote","isbn":"978-001","numPaginas":1032,"anoPublicacao":1605,"resumo":"O fidalgo que enlouqueceu lendo livros de cavalaria.", "quantidadeEstoque":15,"precoBase":120.00,"status":"DISPONIVEL","editora":{"id":"edit-003","nome":"Penguin Classics","cnpj":"33.333.333/0001-33","telefone":"1133330003","email":"info@penguin.com.br"},"autores":[{"id":"auth-001","nome":"Miguel de Cervantes","dataNascimento":"1547-09-29","nacionalidade":"Espanhol"}],"categorias":[{"id":"cat-007","nome":"Aventura"}, {"id":"cat-004","nome":"Clássico"}]}, {"id":"book-002","titulo":"1984","isbn":"978-002","numPaginas":416,"anoPublicacao":1949,"resumo":"O Grande Irmão está de olho em você.", "quantidadeEstoque":50,"precoBase":45.00,"status":"DISPONIVEL","editora":{"id":"edit-001","nome":"Companhia das Letras","cnpj":"11.111.111/0001-11","telefone":"1133330001","email":"contato@cialetras.com.br"},"autores":[{"id":"auth-002","nome":"George Orwell","dataNascimento":"1903-06-25","nacionalidade":"Britânico"}],"categorias":[{"id":"cat-004","nome":"Clássico"}, {"id":"cat-006","nome":"Distopia"}]}, {"id":"book-003","titulo":"O Senhor dos Anéis","isbn":"978-003","numPaginas":1200,"anoPublicacao":1954,"resumo":"Uma jornada épica pela Terra Média.", "quantidadeEstoque":1000,"precoBase":39.90,"status":"DISPONIVEL","editora":{"id":"edit-002","nome":"HarperCollins","cnpj":"22.222.222/0001-22","telefone":"1133330002","email":"sac@harpercollins.com.br"},"autores":[{"id":"auth-003","nome":"J.R.R. Tolkien","dataNascimento":"1892-01-03","nacionalidade":"Britânico"}],"categorias":[{"id":"cat-007","nome":"Aventura"}, {"id":"cat-003","nome":"Fantasia"}]}, {"id":"book-004","titulo":"A Metamorfose","isbn":"978-004","numPaginas":96,"anoPublicacao":1915,"resumo":"Gregor Samsa acorda transformado em um inseto.", "quantidadeEstoque":20,"precoBase":55.00,"status":"DISPONIVEL","editora":{"id":"edit-001","nome":"Companhia das Letras","cnpj":"11.111.111/0001-11","telefone":"1133330001","email":"contato@cialetras.com.br"},"autores":[{"id":"auth-004","nome":"Franz Kafka","dataNascimento":"1883-07-03","nacionalidade":"Tcheco"}],"categorias":[{"id":"cat-008","nome":"Drama"}, {"id":"cat-004","nome":"Clássico"}]}, {"id":"book-005","titulo":"Orgulho e Preconceito","isbn":"978-005","numPaginas":424,"anoPublicacao":1813,"resumo":"O romance entre Elizabeth Bennet e Mr. Darcy.", "quantidadeEstoque":30,"precoBase":29.90,"status":"DISPONIVEL","editora":{"id":"edit-003","nome":"Penguin Classics","cnpj":"33.333.333/0001-33","telefone":"1133330003","email":"info@penguin.com.br"},"autores":[{"id":"auth-001","nome":"Jane Austen","dataNascimento":"1775-01-01","nacionalidade":"Britânica"}],"categorias":[{"id":"cat-004","nome":"Clássico"}]}]
```

```
← → ↻ 🕒 localhost:8080/api/livros
Estilos de formatação ☒

[
  {
    "id": "book-001",
    "titulo": "Dom Quixote",
    "isbn": "978-001",
    "numPaginas": 1032,
    "anoPublicacao": 1605,
    "resumo": "O fidalgo que enlouqueceu lendo livros de cavalaria.",
    "quantidadeEstoque": 15,
    "precoBase": 120,
    "status": "DISPONIVEL",
    "editora": {
      "id": "edit-003",
      "nome": "Penguin Classics",
      "cnpj": "33.333.333/0001-33",
      "telefone": "1133330003",
      "email": "info@penguin.com.br"
    },
    "autores": [
      {
        "id": "auth-001",
        "nome": "Miguel de Cervantes",
        "dataNascimento": "1547-09-29",
        "nacionalidade": "Espanhol"
      }
    ],
    "categorias": [
      {
        "id": "cat-007",
        "nome": "Aventura"
      },
      {
        "id": "cat-004",
        "nome": "Clássico"
      }
    ]
  },
  {
    "id": "book-002",
    "titulo": "1984",
    "isbn": "978-002",
    "numPaginas": 416,
    "anoPublicacao": 1949,
    "resumo": "O Grande Irmão está de olho em você.",
    "quantidadeEstoque": 50,
    "precoBase": 45,
    "status": "DISPONIVEL",
    "editora": {
      "id": "edit-001",
      "nome": "Companhia das Letras",
      "cnpj": "11.111.111/0001-11",
      "telefone": "1133330001",
      "email": "contato@cialetras.com.br"
    },
    "autores": [
      {
        "id": "auth-002",
        "nome": "George Orwell",
        "dataNascimento": "1903-06-25",
        "nacionalidade": "Britânico"
      }
    ],
    "categorias": [
      {
        "id": "cat-004",
        "nome": "Clássico"
      },
      {
        "id": "cat-006",
        "nome": "Distopia"
      }
    ]
  },
  {
    "id": "book-003",
    "titulo": "O Senhor dos Anéis",
    "isbn": "978-003",
    "numPaginas": 1200,
    "anoPublicacao": 1954,
    "resumo": "Uma jornada épica pela Terra Média.",
    "quantidadeEstoque": 1000,
    "precoBase": 39.9,
    "status": "DISPONIVEL",
    "editora": {
      "id": "edit-002",
      "nome": "HarperCollins",
      "cnpj": "22.222.222/0001-22",
      "telefone": "1133330002",
      "email": "sac@harpercollins.com.br"
    },
    "autores": [
      {
        "id": "auth-003",
        "nome": "J.R.R. Tolkien",
        "dataNascimento": "1892-01-03",
        "nacionalidade": "Britânico"
      }
    ],
    "categorias": [
      {
        "id": "cat-007",
        "nome": "Aventura"
      },
      {
        "id": "cat-003",
        "nome": "Fantasia"
      }
    ]
  },
  {
    "id": "book-004",
    "titulo": "A Metamorfose",
    "isbn": "978-004",
    "numPaginas": 96,
    "anoPublicacao": 1915,
    "resumo": "Gregor Samsa acorda transformado em um inseto.",
    "quantidadeEstoque": 20,
    "precoBase": 55,
    "status": "DISPONIVEL",
    "editora": {
      "id": "edit-001",
      "nome": "Companhia das Letras",
      "cnpj": "11.111.111/0001-11",
      "telefone": "1133330001",
      "email": "contato@cialetras.com.br"
    },
    "autores": [
      {
        "id": "auth-004",
        "nome": "Franz Kafka",
        "dataNascimento": "1883-07-03",
        "nacionalidade": "Tcheco"
      }
    ],
    "categorias": [
      {
        "id": "cat-008",
        "nome": "Drama"
      },
      {
        "id": "cat-004",
        "nome": "Clássico"
      }
    ]
  },
  {
    "id": "book-005",
    "titulo": "Orgulho e Preconceito",
    "isbn": "978-005",
    "numPaginas": 424,
    "anoPublicacao": 1813,
    "resumo": "O romance entre Elizabeth Bennet e Mr. Darcy.",
    "quantidadeEstoque": 30,
    "precoBase": 29.9,
    "status": "DISPONIVEL",
    "editora": {
      "id": "edit-003",
      "nome": "Penguin Classics",
      "cnpj": "33.333.333/0001-33",
      "telefone": "1133330003",
      "email": "info@penguin.com.br"
    },
    "autores": [
      {
        "id": "auth-001",
        "nome": "Jane Austen",
        "dataNascimento": "1775-01-01",
        "nacionalidade": "Britânica"
      }
    ],
    "categorias": [
      {
        "id": "cat-004",
        "nome": "Clássico"
      }
    ]
  }
]
```

NGINX via navegador na porta 8080, retornando corretamente a lista de livros.

## 8. Relatório de testes de performance com JMeter

Os testes de desempenho foram realizados com o Apache JMeter, avaliando dois cenários:

- Sem balanceamento (acessando diretamente a porta 8081)
- Com balanceamento (acessando a API pela porta 8080 via NGINX)

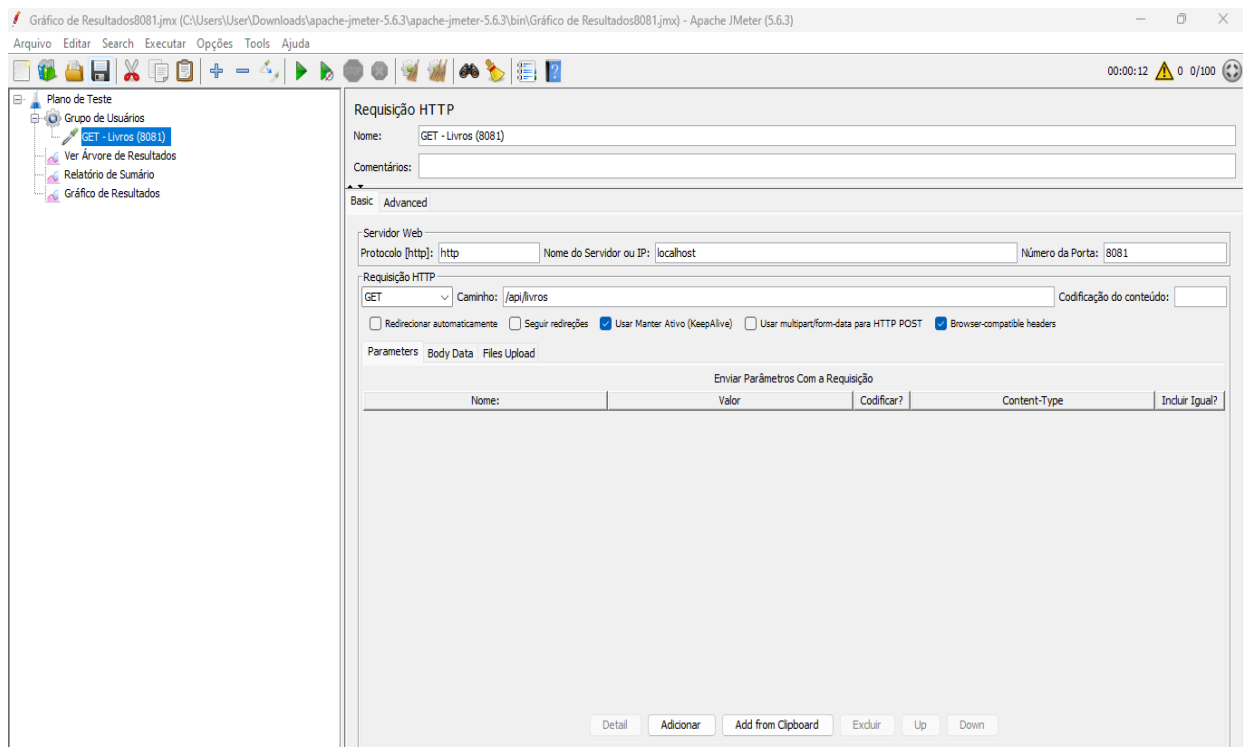
Cada teste foi configurado com 100 usuários virtuais, Ramp-Up de 10 segundos e 5 iterações, totalizando 500 requisições no primeiro cenário e 1000 no segundo.

- **Configurando o primeiro teste (SEM balanceamento)**

Para este cenário, apenas a instância da API na porta **8081** permaneceu ativa.

As imagens registram:

- A configuração da requisição HTTP no JMeter com:
  - Servidor: **localhost**
  - Porta: **8081**
  - Caminho: **/api/livros**



- Prova visual de que:
  - O **NGINX** estava desligado
  - A porta **8082** estava desligada
  - Apenas a porta **8081** estava em execução

```
PROBLEMAS          SAÍDA          CONSOLE DE DEPURACÃO          TERMINAL          PORTAS
```

```
"A Rússia durante as guerras napoleônicas.", "quantidadeEstoque":2, "precoBase":115.00, "status":"DISPONIVEL", "editora":{"id":"edit-001", "nome":"Companhia das Letras", "cnpj":"11.111.111/0001-11", "telefone":"1133330001", "email":"contato@cialetras.com.br"}, "autores":[{"id":"auth-019", "nome":"Liev Tolstói", "dataNascimento":"1828-09-09", "nacionalidade":"Russo"}], "categorias":[{"id":"cat-008", "nome":"Drama"}, {"id":"cat-004", "nome":"Clássico"}]]}]]]
C:\nginx>

C:\nginx>nginx -s stop

C:\nginx>

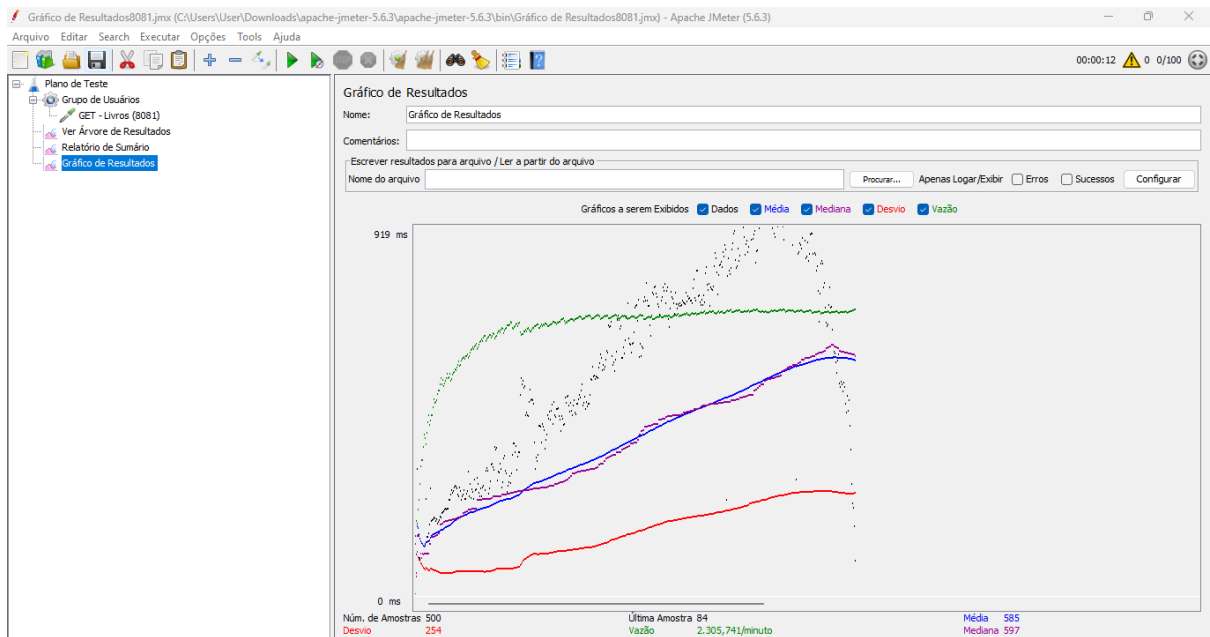
C:\nginx>cd 0.livro id=?
2025-11-30T15:58:14.859-03:00 INFO 33828 --- [ionShutdownHook] j.LocalContainerEntityManagerFactoryBean : Closing JPA EntityManagerFactory for persistence unit 'default'
2025-11-30T15:58:15.061-03:00 INFO 33828 --- [ionShutdownHook] com.zaxxer.hikari.HikariDataSource       : HikariPool-1 - Shutdown initiated...
2025-11-30T15:58:15.118-03:00 INFO 33828 --- [ionShutdownHook] com.zaxxer.hikari.HikariDataSource       : HikariPool-1 - Shutdown completed.
[INFO] -----
[INFO] BUILD SUCCESS
[INFO] -----
[INFO] Total time: 17:36 h
[INFO] Finished at: 2025-11-30T15:58:18-03:00
[INFO] -----
Deseja finalizar o arquivo em lotes (S/N)? s
```

- Resultados do teste:
  - Gráfico de resultados
  - Relatório de Sumário
  - Árvore de Resultados
  - Resultados em Tabela

Esses registros validam o comportamento da API com apenas uma instância ativa.







## Ver Resultados em Tabela

Nome: Ver Resultados em Tabela

Comentários:

Escrever resultados para arquivo / Ler a partir do arquivo

Nome do arquivo

Procurar...

Apenas Logar/Exibir ☐ Erros ☐ Sucessos

Configurar

Amostra #	Tempo de início	Nome do Usuári...	Rótulo	Tempo da amos...	Estado	Bytes	Sent Bytes	Latency	Connect Time(ms)
1	17:52:44.685	Grupo de Usuári...	GET - Livros (808...	70	✓	11038	126	70	1
2	17:52:44.757	Grupo de Usuári...	GET - Livros (808...	49	✓	11038	126	49	0
3	17:52:44.786	Grupo de Usuári...	GET - Livros (808...	50	✓	11038	126	50	1
4	17:52:44.806	Grupo de Usuári...	GET - Livros (808...	47	✓	11038	126	47	0
5	17:52:44.836	Grupo de Usuári...	GET - Livros (808...	47	✓	11038	126	47	0
6	17:52:44.853	Grupo de Usuári...	GET - Livros (808...	56	✓	11038	126	56	0
7	17:52:44.883	Grupo de Usuári...	GET - Livros (808...	79	✓	11038	126	79	0
8	17:52:44.886	Grupo de Usuári...	GET - Livros (808...	82	✓	11038	126	82	0
9	17:52:44.909	Grupo de Usuári...	GET - Livros (808...	90	✓	11038	126	90	0
10	17:52:44.962	Grupo de Usuári...	GET - Livros (808...	101	✓	11038	126	101	0
11	17:52:44.968	Grupo de Usuári...	GET - Livros (808...	100	✓	11038	126	100	0
12	17:52:44.986	Grupo de Usuári...	GET - Livros (808...	99	✓	11038	126	99	1
13	17:52:45.063	Grupo de Usuári...	GET - Livros (808...	97	✓	11038	126	97	0
14	17:52:45.069	Grupo de Usuári...	GET - Livros (808...	100	✓	11038	126	99	0
15	17:52:45.085	Grupo de Usuári...	GET - Livros (808...	94	✓	11038	126	94	0
16	17:52:45.087	Grupo de Usuári...	GET - Livros (808...	93	✓	11038	126	93	0
17	17:52:45.168	Grupo de Usuári...	GET - Livros (808...	83	✓	11038	126	83	0
18	17:52:45.179	Grupo de Usuári...	GET - Livros (808...	83	✓	11038	126	83	0
19	17:52:45.180	Grupo de Usuári...	GET - Livros (808...	87	✓	11038	126	87	0
20	17:52:45.185	Grupo de Usuári...	GET - Livros (808...	90	✓	11038	126	90	1
21	17:52:45.251	Grupo de Usuári...	GET - Livros (808...	106	✓	11038	126	106	0
22	17:52:45.263	Grupo de Usuári...	GET - Livros (808...	107	✓	11038	126	107	0
23	17:52:45.268	Grupo de Usuári...	GET - Livros (808...	106	✓	11038	126	106	0
24	17:52:45.275	Grupo de Usuári...	GET - Livros (808...	99	✓	11038	126	99	0
25	17:52:45.285	Grupo de Usuári...	GET - Livros (808...	108	✓	11038	126	108	1
26	17:52:45.370	Grupo de Usuári...	GET - Livros (808...	129	✓	11038	126	129	0
27	17:52:45.375	Grupo de Usuári...	GET - Livros (808...	129	✓	11038	126	129	0

## • Resumo dos Resultados — (8081) Sem Balanceamento

- **Amostras:** 500
- **Média:** 585 ms
- **Min:** 55 ms
- **Max:** 1481 ms
- **Desvio Padrão:** 254,82
- **% Erro:** 0%
- **Vazão (Throughput):** 38,4 requisições/seg
- **KB/s:** 414,24
- **Média de bytes:** 11038 bytes

Esses valores indicam uma operação estável, com nenhuma taxa de erro e boa taxa de processamento, considerando que apenas uma instância estava respondendo as requisições.

## • Configurando o segundo teste (COM balanceamento)

Nesse cenário, as três portas foram ativadas simultaneamente:

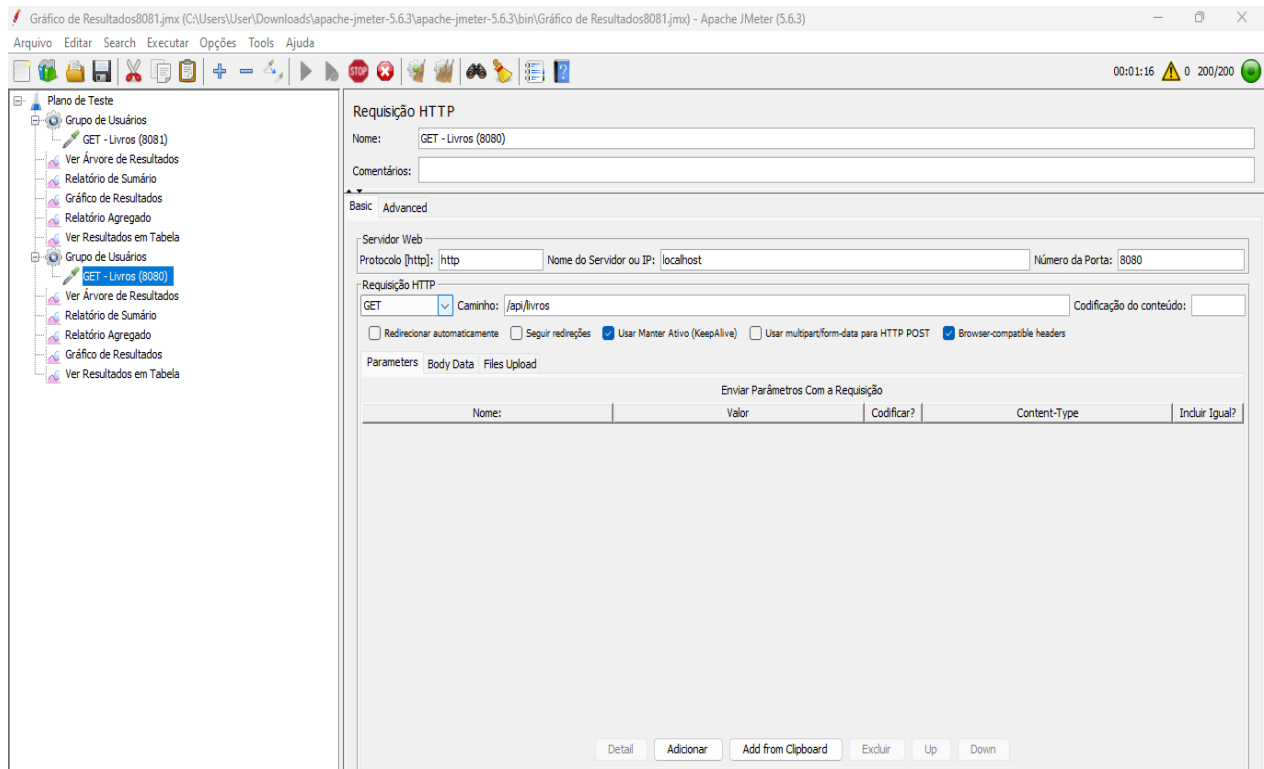
- 8081 e 8082: Instâncias da API
- 8080: NGINX realizando o balanceamento

As imagens utilizadas demonstram:

- O NGINX e ambas as instâncias da API ativos

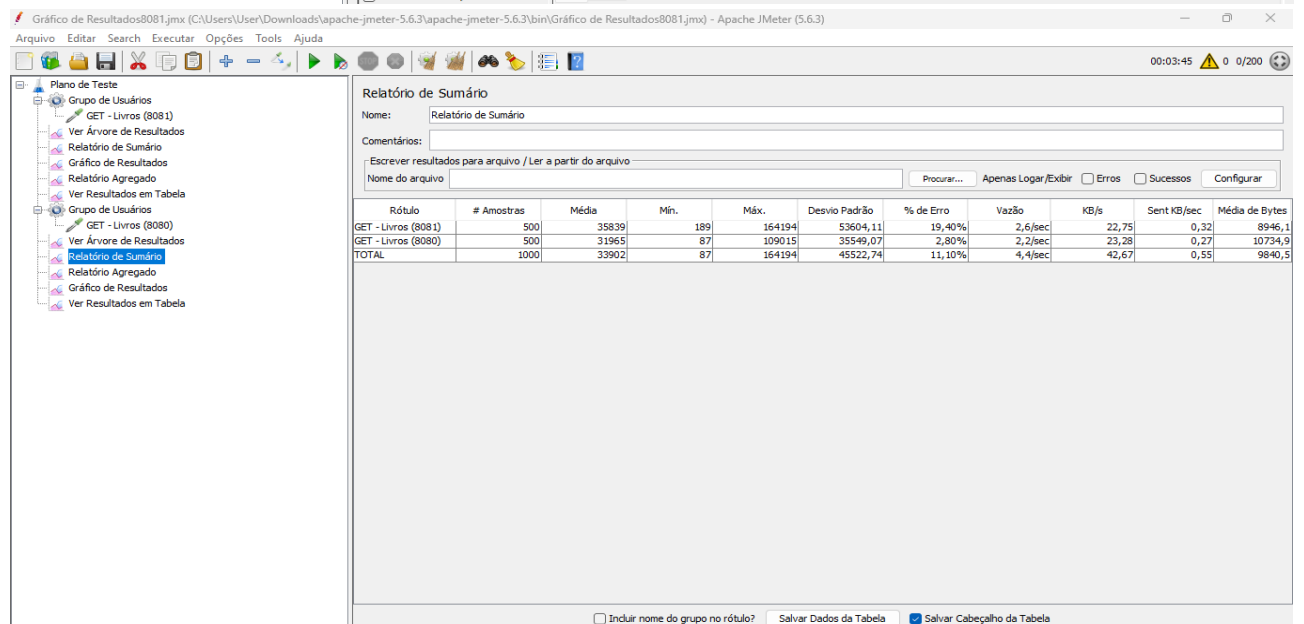
```
C:\nginx>netstat -ano | findstr /R "8080 8081 8082"
TCP    0.0.0.0:8080        0.0.0.0:0          LISTENING        22456
TCP    0.0.0.0:8081        0.0.0.0:0          LISTENING        22608
TCP    0.0.0.0:8082        0.0.0.0:0          LISTENING        34048
TCP    [::]:8081          [::]:0              LISTENING        22608
TCP    [::]:8082          [::]:0              LISTENING        34048
```

## Configuração da requisição HTTP no JMeter apontando para a porta **8080**



- Resultados do teste:
  - Gráfico de resultados
  - Relatório de Sumário
  - Árvore de Resultados
  - Relatório Agregado
  - Resultados em Tabela

Esses dados comprovam o funcionamento correto do balanceamento de carga e permitem comparação com o cenário sem balanceamento.



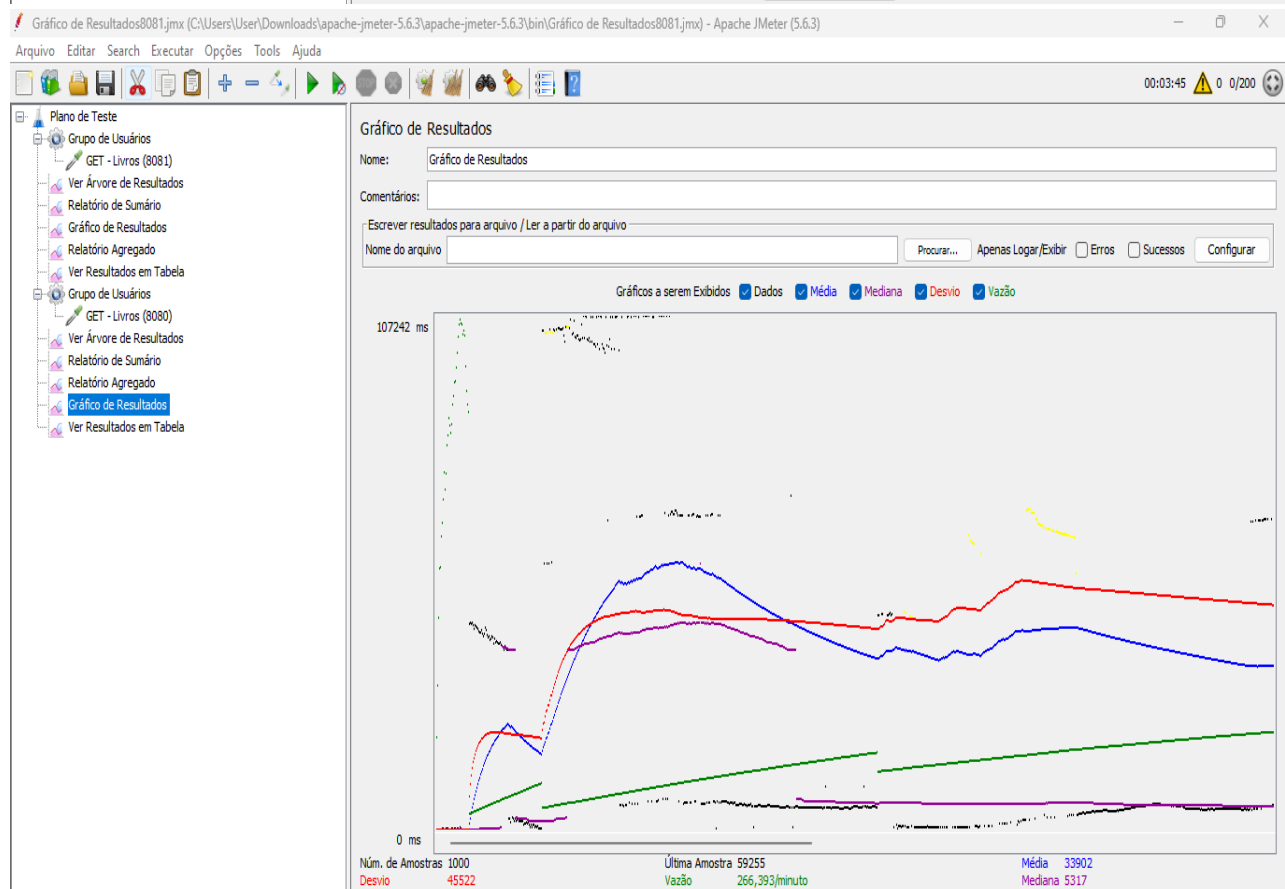
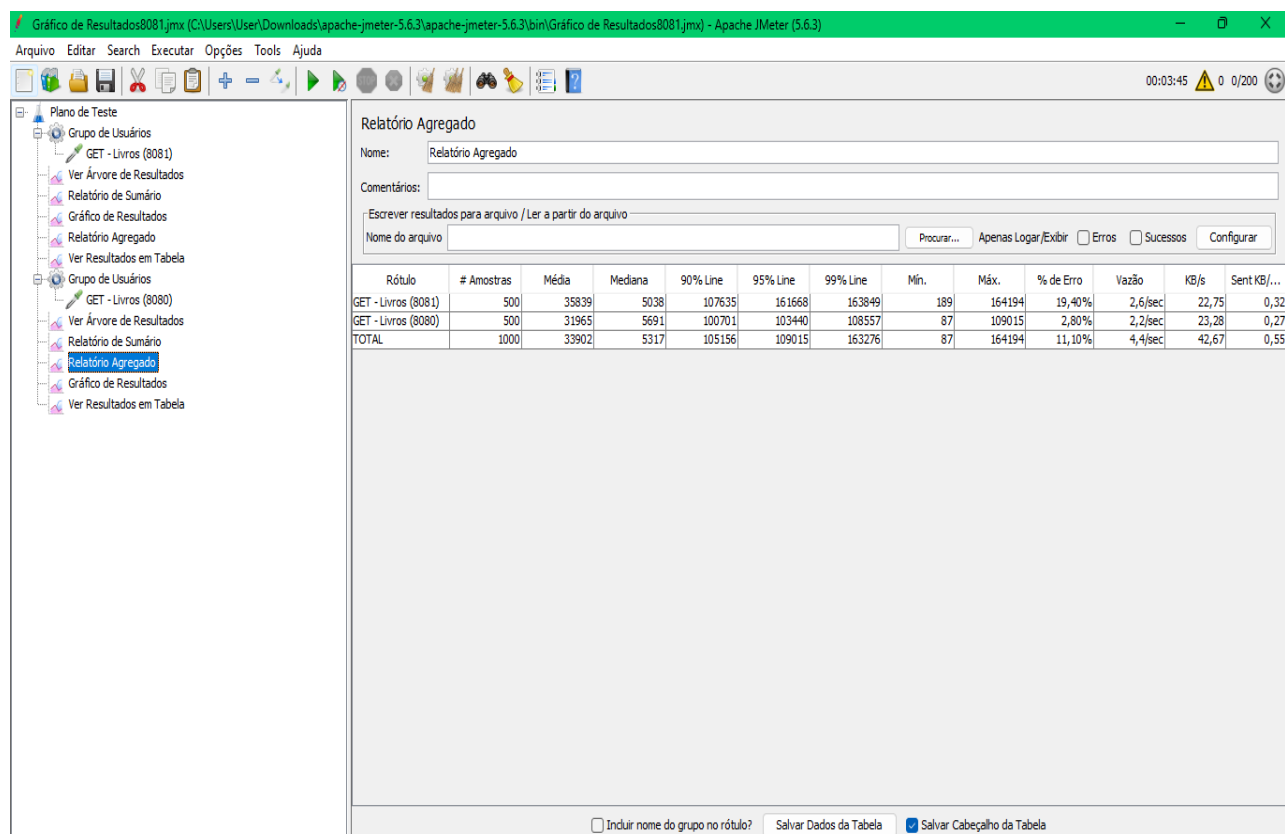


Gráfico de Resultados8081.jmx (C:\Users\User\Downloads\apache-jmeter-5.6.3\apache-jmeter-5.6.3\bin\Gráfico de Resultados8081.jmx) - Apache JMeter (5.6.3)

Arquivo Editar Search Executar Opções Tools Ajuda

00:03:45 0 0/200

Plano de Teste

- Grupo de Usuários
  - GET - Livros (8081)
    - Ver Árvore de Resultados
    - Relatório de Sumário
    - Gráfico de Resultados
    - Relatório Agregado
    - Ver Resultados em Tabela
  - Grupo de Usuários
    - GET - Livros (8080)
      - Ver Árvore de Resultados
      - Relatório de Sumário
      - Relatório Agregado
      - Gráfico de Resultados
      - Ver Resultados em Tabela

Ver Resultados em Tabela

Nome: Ver Resultados em Tabela

Comentários:

Escrever resultados para arquivo / Ler a partir do arquivo

Nome do arquivo: Procurar... Apenas Logar/Exibir ☐ Erros ☐ Sucessos Configurar

Amostra #	Tempo de início	Nome do Usuário...	Rótulo	Tempo da amostra...	Estado	Bytes	Sent Bytes	Latency	Connect Time(ms)
1	18:45:44.357	Grupo de Usuário...	GET - Livros (8081)	221	✓	11038	126	221	8
2	18:45:44.443	Grupo de Usuário...	GET - Livros (8080)	196	✓	11036	126	196	0
3	18:45:44.456	Grupo de Usuário...	GET - Livros (8081)	189	✓	11038	126	188	0
4	18:45:44.543	Grupo de Usuário...	GET - Livros (8080)	126	✓	11036	126	125	1
5	18:45:44.639	Grupo de Usuário...	GET - Livros (8080)	107	✓	11036	126	106	0
6	18:45:44.669	Grupo de Usuário...	GET - Livros (8080)	106	✓	11036	126	106	0
7	18:45:44.556	Grupo de Usuário...	GET - Livros (8081)	266	✓	11038	126	266	1
8	18:45:44.578	Grupo de Usuário...	GET - Livros (8081)	273	✓	11038	126	272	0
9	18:45:44.746	Grupo de Usuário...	GET - Livros (8080)	126	✓	11036	126	126	0
10	18:45:44.842	Grupo de Usuário...	GET - Livros (8080)	116	✓	11036	126	115	1
11	18:45:44.645	Grupo de Usuário...	GET - Livros (8081)	344	✓	11038	126	344	0
12	18:45:44.643	Grupo de Usuário...	GET - Livros (8080)	351	✓	11036	126	351	1
13	18:45:44.660	Grupo de Usuário...	GET - Livros (8081)	370	✓	11038	126	370	0
14	18:45:44.942	Grupo de Usuário...	GET - Livros (8080)	110	✓	11036	126	110	1
15	18:45:44.995	Grupo de Usuário...	GET - Livros (8080)	120	✓	11036	126	120	0
16	18:45:45.060	Grupo de Usuário...	GET - Livros (8080)	122	✓	11036	126	122	1
17	18:45:44.744	Grupo de Usuário...	GET - Livros (8080)	469	✓	11036	126	469	1
18	18:45:44.770	Grupo de Usuário...	GET - Livros (8081)	452	✓	11038	126	452	0
19	18:45:45.143	Grupo de Usuário...	GET - Livros (8080)	109	✓	11036	126	108	1
20	18:45:44.776	Grupo de Usuário...	GET - Livros (8080)	495	✓	11036	126	494	0
21	18:45:44.823	Grupo de Usuário...	GET - Livros (8081)	492	✓	11038	126	492	0
22	18:45:45.214	Grupo de Usuário...	GET - Livros (8080)	116	✓	11036	126	115	0
23	18:45:44.849	Grupo de Usuário...	GET - Livros (8081)	494	✓	11038	126	494	1
24	18:45:44.851	Grupo de Usuário...	GET - Livros (8081)	501	✓	11038	126	501	0
25	18:45:45.252	Grupo de Usuário...	GET - Livros (8080)	117	✓	11036	126	117	0
26	18:45:44.872	Grupo de Usuário...	GET - Livros (8080)	526	✓	11036	126	526	0
27	18:45:45.330	Grupo de Usuário...	GET - Livros (8080)	117	✓	11036	126	117	0

☐ Scroll automatically? ☐ Child samples? Núm. de Amostras 1000 Última Amostra 59255 Média 33902 Desvio 45522

## ● Resumo dos Resultados — (8080) Com Balanceamento

- Amostras: 1000
- Média: 33.902 ms
- Min: 87 ms
- Max: 164.194 ms
- Desvio Padrão: 45.522,74
- % Erro: 11,10%
- Vazão (Throughput): 4,4 requisições/seg
- KB/s: 42,67
- Bytes médios: 9840,5

Este cenário apresentou uma taxa de erro maior e um tempo médio elevado, indicando sobrecarga durante a execução do teste com distribuição de carga. Isso pode ocorrer devido a configuração local, concorrência alta em duas instâncias e processamento simultâneo pelo NGINX.

- **Comparação entre os Cenários**

Métrica	Sem Balanceamento (8081)	Com Balanceamento (8080)
Amostras	500	1000
Média (ms)	585 ms	33.902 ms
Máximo (ms)	1481 ms	164.194 ms
% Erro	0%	11,10%
Throughput	38,4 req/s	4,4 req/s

- **Análise**

- O cenário **sem balanceamento** apresentou desempenho melhor, com throughput alto, zero erros e baixa latência média.
- Já o cenário **com balanceamento** obteve latência muito alta e porcentagem de erros significativa. Isso é comum em ambientes *locais*, onde múltiplas instâncias competem pelos mesmos recursos (CPU, RAM, disco).
- Apesar disso, o teste **comprova o funcionamento do balanceamento**, pois o NGINX distribuiu requisições entre 8081 e 8082 corretamente.



## 1. Geração dos Artefatos (.JAR) dos Componentes

Conforme exigido pela especificação de componentes (BRADECO), cada módulo do projeto foi empacotado individualmente como um arquivo .jar.

A geração dos artefatos foi realizada através do Maven com o comando:

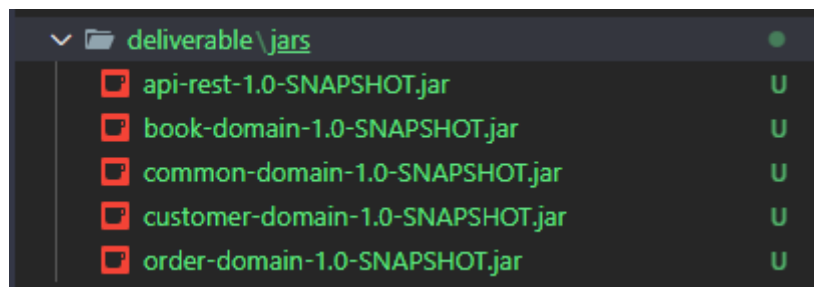
- `mvn clean package -DskipTests`

Após o build, os arquivos .jar gerados foram organizados na pasta:

- `deliverable/jars/`

Os componentes empacotados são:

- **book-domain.jar** – Regras de negócio relacionadas ao catálogo de livros
- **customer-domain.jar** – Gestão de clientes e seus endereços
- **order-domain.jar** – Processamento de pedidos
- **common-domain.jar** – Utilitários e elementos compartilhados
- **api-rest.jar** – Camada de apresentação (controllers REST) que integra os componentes



## 2. Aplicação dos Princípios SOLID

Aluno	RA	Componente
Vinícius Pereira Costa	BP3044289	book-domain
Jonas Ribeiro da Rosa	BP304422X	order-domain
Gabriela Vieira dos Santos Vaz	BP3044513	customer-domain
Iago Felipe Steigleder Bacci	BP3044246	common-domain

## Parte Individual – Vinícius

### Componente: book-domain

O componente book-domain é responsável por toda a lógica de negócio relacionada ao catálogo de livros da plataforma.

#### 1. Princípio da Responsabilidade Única

Definição: Uma classe deve ter apenas um motivo para mudar, ou seja, deve ter apenas uma responsabilidade.

##### *Livro (Classe Abstrata)*

A classe Livro é responsável apenas por manter os dados e a lógica de estoque (decrementar/incrementar). A responsabilidade de cálculo de preço é delegada aos seus subtipos (SRP aplicado por delegação de responsabilidade).

#### 2. Princípio do Aberto/Fechado

Definição: Entidades de software (classes, módulos, funções, etc.) devem ser abertas para extensão, mas fechadas para modificação.

##### *Livro e suas subclasses (LivroCapaDura, LivroBrochura, LivroDigital)*

O cálculo de preço está aberto para extensão. Para adicionar um novo formato de livro com uma regra de preço diferente, basta criar uma nova subclasse de Livro e implementar o método calcularPreco(). O código da classe base Livro permanece fechado para modificação.

#### 3. Princípio da Substituição de Liskov

Definição: Se S é um subtipo de T, então objetos do tipo T podem ser substituídos por objetos do tipo S sem alterar as propriedades corretas do programa.

Qualquer objeto do tipo LivroCapaDura, LivroBrochura ou LivroDigital pode ser usado no lugar de um Livro sem quebrar o comportamento esperado.

#### 4. Princípio da Segregação de Interface

Definição: “Clientes não devem ser forçados a depender de interfaces que não utilizam.”

O módulo book-domain lida com múltiplas entidades: Livro, Autor, Editora e Categoria. Em vez de criar uma única interface de persistência genérica (ex: CatalogoRepository) que conteria métodos para salvar, buscar e deletar todas essas entidades, o projeto utiliza interfaces de repositório separadas, dessa forma, cada cliente depende apenas das interfaces que realmente utiliza.

#### 5. Princípio da Inversão de Dependência

Definição: Módulos de alto nível não devem depender de módulos de baixo nível. Ambos devem depender de abstrações.

O *ManterLivroService* e o *ManterEditoraService* utilizam apenas as interfaces *LivroRepository* e *EditoraRepository*, sem conhecer detalhes de persistência. As implementações reais desses repositórios são definidas externamente (na camada api-rest), fazendo com que os detalhes dependam das abstrações, e não o contrário.

### Parte Individual – Jonas

#### Componente: Order-domain

É responsável por toda a lógica de negócio relacionada a pedidos, incluindo cálculo de totais, atualização de estoque, mudanças de status e aplicação das regras de pagamento.

##### 1. Princípio da Responsabilidade Única

Definição: Uma classe deve ter apenas um motivo para mudar, ou seja, deve ter apenas uma responsabilidade.

A classe *EfetuarPedidoService* tem a responsabilidade única de orquestrar a transação de criação de pedido (validar, calcular total, persistir). A lógica de cálculo de desconto é isolada nas classes de pagamento (*PagamentoPix*, *PagamentoCartao*), e a lógica de persistência é isolada nos repositórios.

## 2. Princípio do Aberto/Fechado

Definição: Entidades de software (classes, módulos, funções, etc.) devem ser abertas para extensão, mas fechadas para modificação.

A classe abstrata *Pagamento* define o contrato (*processarDesconto()*), e as subclasses (*PagamentoPix*, *PagamentoCartao*) implementam a regra. Para adicionar uma nova forma de pagamento (ex: *PagamentoBoleto*), basta criar uma nova subclasse de *Pagamento*, sem modificar o código de criação de pedido.

## 3. Princípio da Substituição de Liskov

Definição: Se S é um subtipo de T, então objetos do tipo T podem ser substituídos por objetos do tipo S sem alterar as propriedades corretas do programa.

*Pagamento* (Classe Abstrata) e suas subclasses *PagamentoFactory* pode retornar qualquer subtipo de *Pagamento* (*PagamentoPix* ou *PagamentoCartao*). O código cliente pode chamar *processarDesconto()* sem se preocupar com o tipo concreto, pois o comportamento é garantido pelo contrato da classe base.

## 4. Princípio da Segregação de Interface

Definição: “Clientes não devem ser forçados a depender de interfaces que não utilizam.”

O serviço que precisa apenas manipular pedidos não é forçado a depender de métodos de persistência de pagamento, mantendo as interfaces pequenas e focadas.

## 5. Princípio da Inversão de Dependência

Definição: Módulos de alto nível não devem depender de módulos de baixo nível. Ambos devem depender de abstrações.

*EfetuarPedidoService* (módulo de alto nível) utiliza apenas as interfaces de repositório (*PedidoRepository*, *PagamentoRepository*), sem conhecer os detalhes de persistência. As implementações reais desses repositórios são definidas externamente (pelo Spring Data JPA), fazendo com que os detalhes dependam das abstrações, e não o contrário.

## Parte Individual – Gabriela

### Componente: Customer-domain

O componente customer-domain é responsável por toda a lógica de negócio relacionada à manutenção e validação dos dados dos clientes.

#### 1. Princípio da Responsabilidade Única

Definição: Uma classe deve ter apenas um motivo para mudar, ou seja, deve ter apenas uma responsabilidade.

A classe *ManterClienteService* tem a responsabilidade única de aplicar as regras de negócio e organizar o CRUD de clientes. A entidade *Cliente* foca apenas em seus dados e regras internas. A responsabilidade de persistência é delegada à interface *ClienteRepository*.

#### 2. Princípio do Aberto/Fechado

Definição: Entidades de software (classes, módulos, funções, etc.) devem ser abertas para extensão, mas fechadas para modificação.

Para expandir o cadastro com novas informações (ex: RG, histórico, preferências), basta estender o modelo com novas classes ou novos campos sem alterar o funcionamento interno do *Cliente*.

#### 3. Princípio da Substituição de Liskov

Definição: Se S é um subtipo de T, então objetos do tipo T podem ser substituídos por objetos do tipo S sem alterar as propriedades corretas do programa.

Qualquer implementação de: *ClienteRepository* e *EnderecoRepository* pode ser substituída por uma implementação alternativa sem quebrar a lógica de cadastro do cliente.

#### 4. Princípio da Segregação de Interface

Definição: “Clientes não devem ser forçados a depender de interfaces que não utilizam.”

O módulo define interfaces de repositório coesas e específicas (*ClienteRepository*). O serviço que precisa apenas manipular clientes não é forçado a depender de interfaces de outros domínios.

## 5. Princípio da Inversão de Dependência

Definição: Módulos de alto nível não devem depender de módulos de baixo nível. Ambos devem depender de abstrações.

O *ManterClienteService* (módulo de alto nível) utiliza apenas a interface *ClienteRepository*, sem conhecer os detalhes de persistência.

## Parte Individual – Iago

### Componente: Common-domain

O componente common-domain é um módulo de baixo nível que contém classes e interfaces compartilhadas por outros domínios, como classes base e utilitários.

#### 1. Princípio da Responsabilidade Única

Definição: Uma classe deve ter apenas um motivo para mudar, ou seja, deve ter apenas uma responsabilidade.

O módulo é responsável apenas por definir estruturas de dados e utilitários que são comuns a todos os domínios (ex: classes base para entidades). Ele não contém lógica de negócio específica de Livro, Pedido ou Cliente.

#### 2. Princípio do Aberto/Fechado

Definição: Entidades de software (classes, módulos, funções, etc.) devem ser abertas para extensão, mas fechadas para modificação.

O módulo é aberto para a adição de novas classes utilitárias que podem ser usados por outros domínios, mas fechado para modificação em suas classes base, garantindo a estabilidade e a não quebra dos módulos que o utilizam.

#### 3. Princípio da Substituição de Liskov

Definição: Se S é um subtipo de T, então objetos do tipo T podem ser substituídos por objetos do tipo S sem alterar as propriedades corretas do programa.

Quaisquer classes base ou interfaces definidas neste módulo devem garantir que seus subtipos sejam substituíveis sem quebrar o comportamento dos módulos que os utilizam.

#### 4. Princípio da Segregação de Interface

Definição: “Clientes não devem ser forçados a depender de interfaces que não utilizam.”

Cada classe utilitária representa uma responsabilidade específica, e nenhuma interface força dependências desnecessárias.

#### 5. Princípio da Inversão de Dependência

Definição: Módulos de alto nível não devem depender de módulos de baixo nível. Ambos devem depender de abstrações.

Atua como a camada de abstração (baixo nível) para a qual os módulos de negócio (alto nível) dependem. Ao depender de classes e interfaces definidas no common-domain, os módulos de negócio evitam depender uns dos outros, promovendo a inversão de dependência e o baixo acoplamento entre os domínios.

### 3. Integração entre Componentes

#### • 3.1 Princípios de Coesão

**Princípio REP:** O projeto organiza o código em quatro módulos independentes (*book-domain*, *order-domain*, *customer-domain*, *common-domain*), garantindo que cada componente represente uma unidade de liberação e reutilização. Cada módulo possui lógica própria, bem delimitada, e pode ser versionado ou distribuído como um .jar independentemente dos demais.

**Princípio CRP:** Cada módulo só contém classes que devem ser reutilizadas juntas.

Exemplos:

- *book-domain* reúne apenas classes relacionadas ao catálogo.
- *order-domain* contém somente classes referentes a pedidos e seu ciclo de vida.
- *customer-domain* agrupa todas as classes relacionadas ao cliente.

Nenhum domínio importa código que não utiliza, evitando dependências desnecessárias.

**Princípio CCP:** Classes que mudam pelos mesmos motivos estão agrupadas no mesmo módulo.

Exemplos:

- Alterações nas regras de preço ou estoque afetam apenas o *book-domain*.
- Alterações nas regras de pedido afetam apenas o *order-domain*.
- Alterações cadastrais afetam apenas o *customer-domain*.

### • 3.2 Princípios de Acoplamento

**Princípio ADP:** A arquitetura modular do projeto é estritamente hierárquica e unidirecional: *api-rest* depende dos módulos de domínio, e os módulos de domínio dependem do *common-domain*. Não há dependências circulares (ex: *book-domain* dependendo de *customer-domain* e *customer-domain* dependendo de *book-domain*), o que garante que o projeto possa ser construído e testado em ordem.

**Princípio SDP:** O módulo *common-domain* é o mais estável (poucas razões para mudar e alta dependência de outros módulos). Os módulos de domínio (*book-domain*, etc.) dependem dele. O módulo *api-rest* (o mais volátil, pois lida com a interface externa) depende dos módulos de domínio. A dependência flui do volátil para o estável.

**Princípio SAP:** O módulo mais estável, o *common-domain*, é composto principalmente por abstrações (classes base, interfaces) que são estendidas pelos módulos de domínio. Isso garante que, embora o *common-domain* seja difícil de mudar (por ser estável), ele é fácil de estender, pois os módulos clientes podem implementar suas abstrações.

## 4. Implementação Funcional dos Casos de Uso

### CSU01 – Pesquisar Livro

Implementado no *book-domain* + exposto no *api-rest*.

Permite:

- listar livros



- buscar por ID
- consultar dados completos do livro

Funciona via *ManterLivroService* e *LivroController*.

### **CSU02 – Efetuar Pedido**

Implementado no *order-domain*.

O pedido:

- recebe os itens
- calcula total
- valida estoque (via book-domain)
- atualiza estoque
- muda status para “EM PROCESSAMENTO, PAGAMENTO PENDENTE e CONFIRMADO”
- finaliza caso pagamento aprovado

Toda lógica central está em *Pedido*, *ItemPedido* e *ManterPedidoService*.

### **CSU03 – Manter Cliente**

Implementado no *customer-domain*.

Permite:

- cadastrar cliente
- atualizar dados
- consultar informações
- manter endereço

Expõe operações via *ManterClienteService*.

### **CSU04 – Visualizar Pedido**

No *order-domain*, permite consultar:

- pedido por ID
- status atual
- itens associados

Expõe as operações por *PedidoController* na API.

## CSU05 – Manter Livro

Implementado no *book-domain*.

Permite:

- cadastrar livro
- atualizar livro
- excluir livro
- atualizar estoque

Tudo feito via *ManterLivroService*.

## 5. Diagrama de Componentes e Diagrama de Arquitetura de Software

