

Casos de Uso

ID:	UC001
Título:	Adicionar Livro
Descrição:	<p>Permitir que o operador adicione um livro no banco de dados da biblioteca.</p> <p>Este caso de uso descreve a operação de adicionar um novo livro ao banco de dados utilizando o método POST. O controlador recebe os dados do livro (como título, autor, ano de publicação) no corpo da requisição (req.body), cria um novo registro no banco de dados com esses dados usando o Sequelize e retorna a resposta ao cliente.</p>
Ator Primário:	Usuário ou sistema (ex: administrador) que envia a requisição para criar um novo livro.
Pré-condição:	<ul style="list-style-type: none">O livro deve ter um título.O livro deve ter um autor.
Pós-condição:	<ul style="list-style-type: none">O livro e seus dados ficam registrados no banco de dados.
Diagrama:	<pre>graph TD; Inicio([Início]) --> Input[/título, autor, genero(opcional), ano_publicado(opcional)/]; Input --> Process[Chamar Livro.create(req.body)]; Process --> Decision{Erro durante a requisição?}; Decision -- Não --> Success[Retornar status 201 e novo Livro]; Success --> Fim1([Fim]); Decision -- Sim --> Error[Retornar status 500 e logar erro no console]; Error --> Fim2([Fim]);</pre>
Fluxo Principal:	<p>1. Requisição:</p> <ul style="list-style-type: none">o O usuário ou sistema envia uma requisição POST para o endpoint destinado a adicionar livros, passando os dados necessários no corpo da requisição (ex: título, autor, ano de publicação). <p>2. Processamento:</p> <ul style="list-style-type: none">o O controlador adicionarLivro() recebe a requisição.o Os dados do livro são extraídos de req.body.o O método Livro.create(req.body) é chamado, que cria um novo registro de livro no banco de dados.

	<p>3. Sucesso:</p> <p>o Se a criação for bem-sucedida, a aplicação retorna: Status HTTP 201 (Criado). Os dados do livro recém-criado, incluindo um ID gerado pelo banco de dados.</p>
<p>Fluxo Alternativo:</p>	<p>Fluxo Alternativo (Erro):</p> <p>1. Se ocorrer um erro no processo de criação (por exemplo, falha de validação ou problemas no banco de dados), o código entra no bloco catch.</p> <p>2. O sistema retorna uma resposta com:</p> <p>o Status HTTP 500.</p> <p>o Uma mensagem de erro: "Erro ao adicionar novo livro."</p> <p>3. O erro é logado no console para análise do desenvolvedor</p>
<p>Tratamento de exceções:</p>	<p>Retorna a mensagem de erro “Ocorreu um erro ao registrar empréstimo” com código de status 500, e uma mensagem é mostrada no console, informando mais detalhes do erro.</p>

ID:	UC002
Título:	Listar Livros
Descrição:	Este caso de uso descreve a operação de recuperar todos os livros do banco de dados. A requisição GET é feita para um endpoint que retorna todos os livros armazenados. O código usa o Sequelize para buscar todos os registros da tabela Livro e retorna esses dados no formato JSON.
Ator Primário:	Usuário ou sistema (ex: administrador) que solicita a lista de livros.
Pré-condição:	<ul style="list-style-type: none">• A tabela Livro já deve existir e conter dados no banco de dados.• A API está rodando e acessível para receber requisições HTTP.• O banco de dados deve estar funcionando corretamente.
Pós-condição:	<ul style="list-style-type: none">• A lista de livros cadastrados é retornada ao cliente em formato JSON.
Diagrama:	<pre>graph TD; Inicio([Início]) --> Call[Chamar Livro.findAll().]; Call --> Decision{Erro Durante a Busca?}; Decision -- Não --> Return200[Retornar status 200 e lista de livros.]; Return200 --> Fim1([Fim]); Decision -- Sim --> Return500[Retornar status 500 e logar erro no console]; Return500 --> Fim2([Fim]);</pre>
Fluxo Principal:	<p>1. Requisição:</p> <ul style="list-style-type: none">o O ator (usuário ou sistema) envia uma requisição GET para o endpoint responsável por listar todos os livros. <p>2. Processamento:</p> <ul style="list-style-type: none">o O controlador listarLivros é invocado para processar a requisição.o O método Livro.findAll() do Sequelize é chamado, que consulta todos os registros da tabela Livro no banco de dados. <p>3. Sucesso:</p> <ul style="list-style-type: none">o Se a consulta for bem-sucedida, a aplicação

	retorna: Status HTTP 200 (OK). Uma lista de objetos de livros no formato JSON, contendo todos os livros cadastrados no banco.
Fluxo Alternativo:	<ol style="list-style-type: none">1. Se a consulta falhar (por exemplo, o banco de dados estiver inacessível), o código entra no bloco catch.2. O sistema retorna uma resposta com o Status HTTP 500 e uma mensagem de erro no formato JSON: "Erro ao buscar livros."
Tratamento de exceções:	<ul style="list-style-type: none">• Retorna a mensagem de erro "Erro ao buscar livros" com código de status 500.• Loga o erro no console, incluindo detalhes da exceção para facilitar a depuração.

ID:	UC003
Título:	Buscar Livro
Descrição:	Permitir que um usuário ou sistema busque um livro específico no banco de dados usando seu ID
Ator Primário:	Usuário ou sistema (ex: administrador) que solicita os detalhes de um livro específico.
Pré-condição:	<ul style="list-style-type: none">• O ID do livro deve ser fornecido como parâmetro na URL.• O banco de dados deve estar conectado corretamente.• O livro com o ID fornecido deve existir no banco de dados (caso contrário, será retornada uma mensagem de erro).
Pós-condição:	<ul style="list-style-type: none">• O cliente recebe os detalhes do livro correspondente ao ID fornecido, ou uma mensagem de erro caso o livro não seja encontrado.
Diagrama:	
Fluxo Principal:	<ol style="list-style-type: none">1. O usuário envia uma requisição GET para o endpoint/livros/:id, incluindo o ID do livro.2. O controlador busca o livro no banco usando Livro.findByPk(id).3. Se encontrado: o Retorna status 200 com os dados do livro em formato JSON.
Fluxo Alternativo:	Fluxo Alternativo 1 (Livro Não Encontrado):

	<div>1. Se o ID fornecido não corresponder a nenhum registro na tabela <code>livros</code>:<ul style="list-style-type: none">○ O sistema retorna uma resposta com:<ul style="list-style-type: none">■ Status HTTP 404 (Não Encontrado).■ Uma mensagem de erro: "Livro não encontrado."</div> <div>Fluxo Alternativo 2 (Erro no Processamento):</div> <div><div>1. Se ocorrer um erro durante a consulta (por exemplo, falha de conexão com o banco de dados):<ul style="list-style-type: none">○ O código entra no bloco <code>catch</code>.</div><div>2. O sistema retorna uma resposta com:<ul style="list-style-type: none">○ Status HTTP 500 (Erro Interno do Servidor).○ Uma mensagem de erro: "Erro ao buscar livro."</div><div>3. O erro é logado no console para análise do desenvolvedor.</div></div>
Tratamento de exceções:	<ul style="list-style-type: none">● Retorna a mensagem de erro "Erro ao buscar livro" com código de status 500.● Loga o erro no console, incluindo detalhes da exceção para facilitar a depuração.

ID:	UC004
Título:	Editar Livro
Descrição:	Permitir que um usuário ou sistema atualize as informações de um livro específico no banco de dados, utilizando o ID do livro.
Ator Primário:	Usuário ou sistema (ex: administrador) que solicita a atualização dos dados de um livro específico.
Pré-condição:	<ul style="list-style-type: none">• O ID do livro deve ser fornecido como parâmetro na URL.• O livro com o ID fornecido deve existir no banco de dados.• Os dados enviados no corpo da requisição devem ser válidos e compatíveis com o modelo Livro.
Pós-condição:	<ul style="list-style-type: none">• O registro do livro no banco de dados é atualizado com os novos dados fornecidos.• Os dados atualizados do livro são retornados ao cliente.
Diagrama:	
Fluxo Principal:	<ol style="list-style-type: none">1. O ator envia uma requisição PUT ou PATCH para o endpoint/livros/:id com os novos dados no corpo.2. O sistema tenta localizar o livro pelo ID com Livro.findByPk(id).3. Se o livro for encontrado:<ul style="list-style-type: none">> A atualização é feita com livro.update(req.body).> Retorna status 200 com os dados do livro atualizado.
Fluxo Alternativo:	Fluxo Alternativo 1 (Livro Não Encontrado): <ol style="list-style-type: none">1. Se o ID fornecido não corresponder a nenhum registro na tabela livros:

	<ul style="list-style-type: none">○ O sistema retorna uma resposta com:<ul style="list-style-type: none">■ Status HTTP 404 (Não Encontrado).■ Uma mensagem de erro: "Livro não encontrado." <p>Fluxo Alternativo 2 (Erro no Processamento):</p> <ol style="list-style-type: none">1. Se ocorrer um erro durante o processo de atualização (por exemplo, falha de validação ou problemas no banco de dados):<ul style="list-style-type: none">○ O código entra no bloco catch.2. O sistema retorna uma resposta com:<ul style="list-style-type: none">○ Status HTTP 500 (Erro Interno do Servidor).○ Uma mensagem de erro: "Erro ao editar livro."3. O erro é logado no console para análise do desenvolvedor.
Tratamento de exceções:	<ul style="list-style-type: none">● Retorna a mensagem de erro "Erro ao editar livro" com código de status 500.● Loga o erro no console, incluindo detalhes da exceção para facilitar a depuração.

ID:	UC005
Título:	Excluir/Deletar Livro
Descrição:	Permitir a exclusão de um livro específico do banco de dados usando seu ID.
Ator Primário:	Usuário ou sistema (ex: administrador) que solicita a exclusão de um livro específico.
Pré-condição:	<ul style="list-style-type: none">• O ID do livro deve ser fornecido como parâmetro na URL.• O livro com o ID fornecido deve existir no banco de dados.
Pós-condição:	<ul style="list-style-type: none">• O registro do livro é removido do banco de dados.• O cliente recebe uma mensagem de confirmação da exclusão.
Diagrama:	
Fluxo Principal:	1. O ator envia uma requisição DELETE para o endpoint /livros/:id com o ID do livro.

	<div>2. O sistema busca o livro com <code>Livro.findByPk(id)</code>.</div> <div>3. Se encontrado, o livro é excluído com <code>livro.destroy()</code>, e retorna status 200 com a mensagem "Livro excluído com sucesso!".</div>
Fluxo Alternativo:	<div>Fluxo Alternativo 1 (Livro Não Encontrado):</div> <div><div>1. Se o ID fornecido não corresponder a nenhum registro na tabela <code>livros</code>:<ul style="list-style-type: none">O sistema retorna uma resposta com:<ul style="list-style-type: none">Status HTTP 404 (Não Encontrado).Uma mensagem de erro: "Livro não encontrado."</div><div>Fluxo Alternativo 2 (Erro no Processamento):</div><div><div>1. Se ocorrer um erro durante o processo de exclusão (por exemplo, falha de conexão com o banco de dados):<ul style="list-style-type: none">O código entra no bloco <code>catch</code>.</div><div>2. O sistema retorna uma resposta com:<ul style="list-style-type: none">Status HTTP 500 (Erro Interno do Servidor).Uma mensagem de erro: "Erro ao excluir livro."</div><div>3. O erro é logado no console para análise do desenvolvedor.</div></div></div>
Tratamento de exceções:	<div><ul style="list-style-type: none">Retorna a mensagem de erro "Erro ao excluir livro" com código de status 500.Loga o erro no console, incluindo detalhes da exceção para facilitar a depuração.</div>

ID:	UC006
Título:	Adicionar Usuário
Descrição:	Permitir a criação de um novo usuário no sistema
Ator Primário:	Usuário ou sistema (ex: administrador) que solicita a criação de um novo usuário.
Pré-condição:	<ul style="list-style-type: none">Os dados do novo usuário (nome, endereço, email, telefone) devem ser fornecidos no corpo da requisição.O campo de email, se fornecido, deve ser único no banco de dados.O banco de dados deve estar conectado corretamente.
Pós-condição:	<ul style="list-style-type: none">O novo usuário é registrado no banco de dados.Os detalhes do usuário criado, incluindo o ID gerado, são retornados ao cliente.
Diagrama:	<pre>graph TD; Inicio([Início]) --> Input[/nome, endereço, e-mail(opcional), telefone/]; Input --> Process[chamar Usuário.create(req.body)]; Process --> Decision{Erro durante requisição?}; Decision -- Não --> Success[Retornar status 201 e dados do novo usuário]; Decision -- Sim --> Error[Retornar status 500, "erro ao criar usuário" e logar erro no console];</pre>
Fluxo Principal:	O ator envia uma requisição POST para /usuarios com os dados do novo usuário no corpo. 2. O sistema cria o usuário usando Usuario.create(req.body). 3. Se bem-sucedido, retorna status 201 com os dados do novo usuário.
Fluxo Alternativo:	Fluxo Alternativo 1 (Erro no Processamento): 1. Se ocorrer um erro durante o processo de criação (por exemplo, falha de validação ou problemas no banco de dados): <ul style="list-style-type: none">O código entra no bloco catch. 2. O sistema retorna uma resposta com: <ul style="list-style-type: none">Status HTTP 500 (Erro Interno do Servidor).Uma mensagem de erro: "Erro ao criar usuário." 3. O erro é logado no console para análise do desenvolvedor.
Tratamento de exceções:	<ul style="list-style-type: none">Retorna a mensagem de erro "Erro ao criar usuário" com código de status 500.

	<ul style="list-style-type: none">Loga o erro no console, incluindo detalhes da exceção para facilitar a depuração.
--	---

ID:	UC007
Título:	Listar todos os usuários
Descrição:	Permitir a listagem de todos os usuários registrados no sistema, retornando suas informações para visualização ou outras operações de gestão.
Ator Primário:	Usuário com permissão para visualizar todos os usuários (geralmente um administrador ou sistema interno).
Pré-condição:	O sistema está funcionando corretamente. O banco de dados contém usuários registrados. O ator possui permissão para acessar a listagem de usuários (por exemplo, um administrador).
Pós-condição:	
Diagrama:	
Fluxo Principal:	<div>1. Requisição:</div> <div><ul style="list-style-type: none">O usuário ou sistema envia uma requisição GET para o endpoint destinado à listagem de usuários (ex: /usuarios).</div> <div>2. Processamento:</div> <div><ul style="list-style-type: none">O controlador <code>listarUsuarios()</code> recebe a requisição.O método <code>Usuario.findAll()</code> é chamado para recuperar todos os registros da tabela <code>usuarios</code>.</div> <div>3. Sucesso:</div> <div><ul style="list-style-type: none">Se a recuperação for bem-sucedida, a aplicação:<ul style="list-style-type: none">Retorna:<ul style="list-style-type: none">Status HTTP 200 (OK).A lista de usuários no formato JSON.</div>
Fluxo Alternativo:	<div>. Se ocorrer um erro ao buscar os usuários (como falha na conexão com o banco de dados ou outro problema técnico), o sistema</div>

ID:	UC007
Título:	Listar todos os usuários
Descrição:	Permitir a listagem de todos os usuários registrados no sistema, retornando suas informações para visualização ou outras operações de gestão.
Ator Primário:	Usuário com permissão para visualizar todos os usuários (geralmente um administrador ou sistema interno).
Pré-condição:	O sistema está funcionando corretamente. O banco de dados contém usuários registrados. O ator possui permissão para acessar a listagem de usuários (por exemplo, um administrador).
	retornará: o Status HTTP 500. o Mensagem de erro no formato JSON: "Erro ao buscar usuários". O erro será logado no console para facilitar a depuração.
Tratamento de exceções:	Retorna a mensagem de erro “Ocorreu um erro ao registrar emp

ID:	UC008
Título:	Buscar usuário
Descrição:	Permitir a busca e leitura das informações de um usuário específico no sistema, com base no seu identificador único (ID).
Ator Primário:	Usuário ou sistema (ex: administrador) que solicita os dados de um usuário específico.
Pré-condição:	<ul style="list-style-type: none">• O ID do usuário deve ser fornecido como parâmetro na URL.• O banco de dados deve estar conectado corretamente.
Pós-condição:	<ul style="list-style-type: none">• Os dados do usuário solicitado são retornados ao cliente.• Caso o usuário não seja encontrado, o cliente é informado com uma mensagem de erro.
Diagrama:	<pre>graph TD; Inicio([Início]) --> Requisição[/Requisição com id/]; Requisição --> Chamar[Chamar Usuario.findByPk(req.params.id)]; Chamar --> Encontrado{Usuário encontrado?}; Encontrado -- Não --> Status404[Retornar status 404 e "Usuário não encontrado."]; Encontrado -- Sim --> ErroExecução{Erro durante a execução?}; ErroExecução -- Não --> RetornarDados[retornar dados do usuário especificado.]; ErroExecução -- Sim --> Status500[Retornar status 500 logar erro no console]; Status404 --> Fim([Fim]); RetornarDados --> Fim; Status500 --> Fim;</pre>
Fluxo Principal:	O ator envia uma requisição GET para o endpoint /usuarios/:id, onde :id é o ID do usuário a ser consultado. 2. O sistema busca o usuário no banco de dados utilizando o Sequelize com o método findByPk(id). 3. Se o usuário for encontrado: o Retorna status 200 (OK). o Envia os dados do usuário em formato JSON.
Fluxo Alternativo:	Fluxo Alternativo 1 (Usuário Não Encontrado): 1. Se o ID fornecido não corresponder a nenhum registro na tabela usuarios: <ul style="list-style-type: none">○ O sistema retorna uma resposta com:<ul style="list-style-type: none">■ Status HTTP 404 (Não Encontrado).■ Uma mensagem de erro: "Usuário não encontrado."

	<p>Fluxo Alternativo 2 (Erro no Processamento):</p> <ol style="list-style-type: none">1. Se ocorrer um erro durante o processo de consulta (por exemplo, problemas de conexão com o banco de dados):<ul style="list-style-type: none">○ O código entra no bloco catch.2. O sistema retorna uma resposta com:<ul style="list-style-type: none">○ Status HTTP 500 (Erro Interno do Servidor).○ Uma mensagem de erro: "Erro ao buscar usuário."3. O erro é logado no console para análise do desenvolvedor.
<p>Tratamento de exceções:</p>	<ul style="list-style-type: none">● Retorna a mensagem de erro "Erro ao buscar usuário" com código de status 500.● Loga o erro no console, incluindo detalhes da exceção para facilitar a depuração.

ID:	UC009
Título:	Editar usuário
Descrição:	Permitir a atualização das informações de um usuário específico no sistema, utilizando seu identificador único (ID)
Ator Primário:	Usuário ou sistema (ex: administrador) que solicita a atualização de um usuário específico.
Pré-condição:	<ul style="list-style-type: none">• O ID do usuário deve ser fornecido como parâmetro na URL.• Os novos dados devem ser fornecidos no corpo da requisição.• O banco de dados deve estar conectado corretamente.
Pós-condição:	<ul style="list-style-type: none">• Os dados do usuário são atualizados no banco de dados.• O cliente recebe os dados atualizados do usuário.
Diagrama:	
Fluxo Principal:	<div><div>1. Requisição:</div><div><ul style="list-style-type: none">○ O usuário ou sistema envia uma requisição PUT para o endpoint destinado à atualização de usuários, incluindo o ID do usuário na URL (ex: /usuarios/:id) e os novos dados no corpo da requisição.</div><div>2. Processamento:</div><div><ul style="list-style-type: none">○ O controlador atualizarUsuario() recebe a requisição.○ O método Usuario.findByPk(req.params.id) é chamado para buscar o registro do usuário correspondente ao ID fornecido.○ Se o usuário for encontrado, o método usuario.update(req.body) é chamado para atualizar os dados do registro no banco de dados.</div><div>3. Sucesso:</div><div><ul style="list-style-type: none">○ Se a atualização for bem-sucedida, a aplicação:<ul style="list-style-type: none">■ Retorna:</div></div>

	<ul style="list-style-type: none">■ Status HTTP 200 (OK).■ Os dados atualizados do usuário no formato JSON.
Fluxo Alternativo:	<p>1. Se o usuário não for encontrado com o ID fornecido, o sistema retornará: o Status HTTP 404. o Mensagem: "Usuário não encontrado".</p> <p>2. Se ocorrer um erro ao atualizar o usuário (por exemplo, falha no banco de dados), o sistema retornará: o Status HTTP 500. o Mensagem de erro: "Erro ao atualizar usuário".</p>
Tratamento de exceções:	

ID:	UC010
Título:	Excluir usuário
Descrição:	Permitir a exclusão de um usuário específico do sistema, identificado pelo seu ID.
Ator Primário:	Usuário;
Pré-condição:	Usuário ou sistema (ex: administrador) que solicita a exclusão de um usuário específico.
Pós-condição:	<ul style="list-style-type: none">• O ID do usuário deve ser fornecido como parâmetro na URL.• O banco de dados deve estar conectado corretamente.
Diagrama:	<pre>graph TD; Inicio([Início]) --> Req[/Requisição com id/]; Req --> Chamar[Chamar Usuario.findByPk(req.params.id)]; Chamar --> Encontrado{Usuário encontrado?}; Encontrado -- Não --> Status404[Retornar status 404 e "Usuário não encontrado."]; Encontrado -- Sim --> ErroExecucao{Erro durante a execução?}; ErroExecucao -- Não --> Destroy[chamar usuario.destroy() e retornar mensagem "usuário excluído com sucesso."]; ErroExecucao -- Sim --> Status500[Retornar status 500 logar erro no consó]; Status404 --> Fim([Fim]); Destroy --> Fim; Status500 --> Fim;</pre>
Fluxo Principal:	<ol style="list-style-type: none">1. O ator envia uma requisição DELETE para o endpoint /usuarios/:id com o ID do usuário a ser excluído.2. O sistema busca o usuário no banco de dados.3. Se o usuário for encontrado, o sistema o exclui e retorna uma mensagem de sucesso com status 200.
Fluxo Alternativo:	<p>Fluxo Alternativo 1 (Usuário Não Encontrado):</p> <ol style="list-style-type: none">1. Se o ID fornecido não corresponder a nenhum registro na tabela usuarios:<ul style="list-style-type: none">○ O sistema retorna uma resposta com:<ul style="list-style-type: none">■ Status HTTP 404 (Não Encontrado).■ Uma mensagem de erro: "Usuário não encontrado."

	<p>Fluxo Alternativo 2 (Erro no Processamento):</p> <ol style="list-style-type: none">1. Se ocorrer um erro durante o processo de exclusão (por exemplo, problemas de conexão com o banco de dados):<ul style="list-style-type: none">○ O código entra no bloco catch.2. O sistema retorna uma resposta com:<ul style="list-style-type: none">○ Status HTTP 500 (Erro Interno do Servidor).○ Uma mensagem de erro: "Erro ao excluir usuário."3. O erro é logado no console para análise do desenvolvedor.
<p>Tratamento de exceções:</p>	<ul style="list-style-type: none">• Retorna a mensagem de erro "Erro ao excluir usuário" com código de status 500.• Loga o erro no console, incluindo detalhes da exceção para facilitar a depuração.

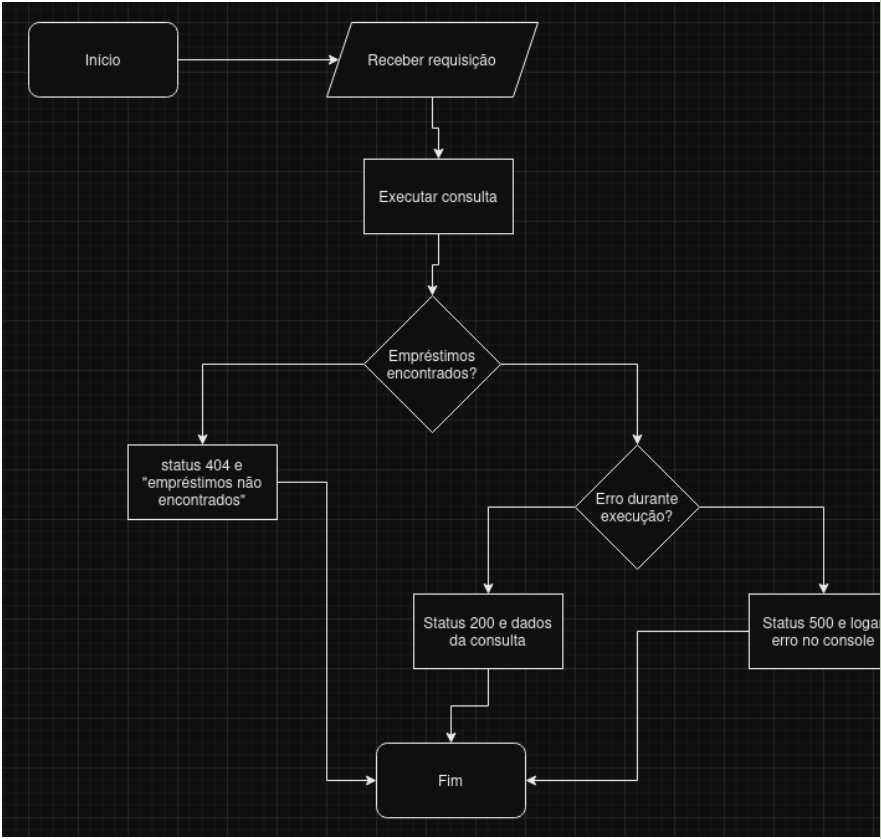
ID:	UC011
Título:	Registrar novo empréstimo
Descrição:	Registrar um novo empréstimo de livro para um usuário, verificando se ele ainda pode realizar mais empréstimos.
Ator Primário:	Usuário ou sistema (ex: bibliotecário ou administrador) que solicita o registro do empréstimo.
Pré-condição:	<ul style="list-style-type: none">• O ID do usuário e do livro devem ser fornecidos no corpo da requisição.• O usuário não deve ultrapassar o limite de 3 empréstimos ativos.• O banco de dados deve estar conectado corretamente.
Pós-condição:	<ul style="list-style-type: none">• O registro do empréstimo é adicionado ao banco de dados.• O cliente recebe uma mensagem de confirmação do registro.
Diagrama:	<pre>graph TD; Inicio([Início]) --> Input[/usuarioId, livroId/]; Input --> Count[Contar empréstimos pendentes do usuário.]; Count --> Limit{Usuário excedeu o limite(3)?}; Limit -- Sim --> Error400[Retornar status 400 e "usuário excedeu o limite de empréstimos"]; Limit -- Não --> Execution{Erro durante execução?}; Execution -- Sim --> Error500[Retornar status 500 logar erro no console]; Execution -- Não --> Create[Criar novo empréstimo com Empréstimo.create({ usuarioId, livroId })]; Create --> Success[Retornar status 201, "empréstimo registrado com sucesso!" e dados do empréstimo];</pre>
Fluxo Principal:	<ol style="list-style-type: none">Requisição:<ul style="list-style-type: none">○ O usuário ou sistema envia uma requisição POST para o endpoint destinado ao registro de empréstimos, incluindo os IDs do usuário e do livro no corpo da requisição (ex: usuarioId e livroId).Processamento:<ul style="list-style-type: none">○ O controlador registrarEmprestimo() recebe a requisição.○ O método Emprestimo.count() é chamado para verificar a quantidade de empréstimos ativos do usuário.○ Se o usuário tiver menos de 3 empréstimos ativos, o método Emprestimo.create() é chamado para registrar o empréstimo no banco de dados.Sucesso:<ul style="list-style-type: none">○ Se o registro for bem-sucedido, a aplicação:<ul style="list-style-type: none">■ Retorna:

	<ul style="list-style-type: none">■ Status HTTP 201 (Criado).■ Uma mensagem de sucesso: "Empréstimo registrado com sucesso!"■ Os detalhes do empréstimo recém-criado.
Fluxo Alternativo:	<p>Fluxo Alternativo 1 (Usuário Excedeu o Limite de Empréstimos):</p> <ol style="list-style-type: none">1. Se o usuário já tiver 3 empréstimos ativos no sistema:<ul style="list-style-type: none">○ O sistema retorna uma resposta com:<ul style="list-style-type: none">■ Status HTTP 400 (Requisição Inválida).■ Uma mensagem de erro: "Usuário atingiu o limite de empréstimos." <p>Fluxo Alternativo 2 (Erro no Processamento):</p> <ol style="list-style-type: none">1. Se ocorrer um erro durante o processo de registro (por exemplo, problemas de validação ou conexão com o banco de dados):<ul style="list-style-type: none">○ O código entra no bloco catch.2. O sistema retorna uma resposta com:<ul style="list-style-type: none">○ Status HTTP 500 (Erro Interno do Servidor).○ Uma mensagem de erro: "Erro ao registrar empréstimo."3. O erro é logado no console para análise do desenvolvedor.
Tratamento de exceções:	<ul style="list-style-type: none">● Retorna a mensagem de erro "Erro ao registrar empréstimo" com código de status 500.● Loga o erro no console, incluindo detalhes da exceção para facilitar a depuração.

ID:	UC012
Título:	Registrar devolução
Descrição:	Registrar a devolução de um livro emprestado, atualizando o status do empréstimo para "devolvido".
Ator Primário:	Usuário ou sistema (ex: bibliotecário ou administrador) que solicita o registro da devolução de um livro.
Pré-condição:	<ul style="list-style-type: none">• O ID do empréstimo deve ser fornecido como parâmetro na URL.• O banco de dados deve estar conectado corretamente.• O empréstimo deve existir no sistema.
Pós-condição:	<ul style="list-style-type: none">• O status do empréstimo é alterado para devolvido.• A data de devolução é registrada.• O cliente recebe uma mensagem de confirmação da devolução.
Diagrama:	<pre>graph TD; Inicio([Início]) --> Requisicao[/requisição com id do empréstimo/]; Requisicao --> Buscar[Buscar empréstimo pelo id]; Buscar --> Existe{Empréstimo existe?}; Existe --> Status404[retornar status 404 e "empréstimo não encontrado"]; Existe --> AtualizarCampos[atualizar campos, Devolvido = true, Data_devolução = data atual]; AtualizarCampos --> AtualizarDados[Atualizar dados no banco de dados. Retornar status 200 com mensagem "livro devolvido com sucesso"]; AtualizarDados --> ErroExecucao{Erro durante execução?}; ErroExecucao --> Fim([Fim]); ErroExecucao --> Status500[Retornar status 500 e Logar erro no console.]; Status500 --> Fim;</pre>
Fluxo Principal:	<ol style="list-style-type: none">1. O ator envia uma requisição POST ou PUT para o endpoint /devolucoes/:id, onde :id é o ID do empréstimo a ser devolvido.2. sistema busca o empréstimo no banco de dados utilizando o ID fornecido (Emprestimo.findByPk(id)).
Fluxo Alternativo:	<ol style="list-style-type: none">1. Se o empréstimo com o ID fornecido não for encontrado, o sistema retorna: o Status HTTP 404. o A mensagem de erro: "Empréstimo não encontrado".2. Se ocorrer um erro ao atualizar o empréstimo (por

	<p>exemplo, falha ao salvar no banco de dados), o sistema retorna:</p> <ul style="list-style-type: none">o Status HTTP 500 (Erro Interno).o A mensagem de erro: "Erro ao registrar devolução"
Tratamento de exceções:	<ul style="list-style-type: none">• Retorna a mensagem de erro "Erro ao registrar devolução" com código de status 500.• Loga o erro no console, incluindo detalhes da exceção para facilitar a depuração.

ID:	UC013
Título:	Listar pendências de usuário
Descrição:	Permitir que o sistema liste todos os empréstimos pendentes (não devolvidos) de um usuário especificado, incluindo informações sobre o usuário e os livros.
Ator Primário:	Usuário ou sistema (ex: bibliotecário ou administrador) que solicita a consulta das pendências de um usuário.
Pré-condição:	<ul style="list-style-type: none">• O ID do usuário deve ser fornecido como parâmetro na URL.• O banco de dados deve estar conectado corretamente.• O usuário deve ter empréstimos pendentes registrados no sistema.
Pós-condição:	Uma lista com os empréstimos pendentes do usuário é retornada, incluindo informações sobre o livro emprestado e o usuário.
Diagrama:	<pre>graph TD; Inicio([Início]) --> Requisicao[/requisição com usuarioid/]; Requisicao --> Erro{Erro durante execução?}; Erro -- Sim --> Status500[Status de erro 500, mensagem e logar erro no console]; Erro -- Não --> Buscar[Buscar Empréstimos Pendentes: - where: { usuarioid, devolvido: false } - include: Usuario, Livro]; Status500 --> Fim([Fim]); Buscar --> Retornar200[Retornar status 200 com lista e empréstimos pendentes.]; Retornar200 --> Fim;</pre>
Fluxo Principal:	O ator envia uma requisição GET para o endpoint /emprestimos/pendentes/:usuarioid, onde :usuarioid é o ID do usuário cujos empréstimos pendentes serão listados. O sistema busca todos os empréstimos não devolvidos do usuário especificado, utilizando o ID do usuário na consulta (Emprestimo.findAll({ where: { usuarioid, devolvido: false } })). O sistema inclui as informações relacionadas ao usuário (include: [{ model: Usuario, as: 'usuario' }]) e ao livro (include: [{ model: Livro, as: 'livro' }]) associadas aos empréstimos pendentes.
Fluxo Alternativo:	. Se o usuário não tiver empréstimos pendentes, o sistema retorna: o Status HTTP 404. o A mensagem de erro: "Nenhum empréstimo pendente encontrado". Se ocorrer um erro ao buscar os empréstimos (por exemplo, falha no banco de dados), o sistema retorna: o Status HTTP 500 (Erro Interno). o A mensagem de erro: "Erro ao listar empréstimos pendentes"
Tratamento de exceções:	<ul style="list-style-type: none">• Retorna a mensagem de erro "Erro ao buscar empréstimos pendentes" com código de status 500.• Loga o erro no console, incluindo detalhes da exceção para facilitar a depuração.

ID:	UC014
Título:	Livros mais emprestados
Descrição:	Listar os livros mais emprestados, mostrando o título e a quantidade de vezes que cada livro foi emprestado, ordenados pela quantidade de empréstimos.
Ator Primário:	Usuário ou sistema (ex: bibliotecário ou administrador) que solicita o relatório de livros mais emprestados.
Pré-condição:	<ul style="list-style-type: none">• O banco de dados deve estar conectado corretamente.• O relacionamento entre os modelos Livro e Empréstimo deve estar configurado corretamente no Sequelize.
Pós-condição:	Um relatório com os livros mais emprestados é retornado, com o título do livro e a quantidade de empréstimos.
Diagrama:	 <pre>graph TD; Inicio([Início]) --> ReceberRequisicao[/Receber requisição/]; ReceberRequisicao --> ExecutarConsulta[Executar consulta]; ExecutarConsulta --> EmpréstimosEncontrados{Empréstimos encontrados?}; EmpréstimosEncontrados --> Status404[status 404 e "empréstimos não encontrados"]; EmpréstimosEncontrados --> ErroDuranteExecucao{Erro durante execução?}; ErroDuranteExecucao --> Status200[Status 200 e dados da consulta]; ErroDuranteExecucao --> Status500[Status 500 e logar erro no console]; Status404 --> Fim([Fim]); Status200 --> Fim; Status500 --> Fim;</pre>
Fluxo Principal:	O ator envia uma requisição GET para o endpoint /livros/mais-emprestados. Exemplo de requisição: O sistema consulta a tabela Livro no banco de dados, utilizando o Sequelize para contar quantos empréstimos foram feitos para cada livro, fazendo uso da função de agregação COUNT no campo empréstimos.id. O sistema agrupa os resultados por Livro.id para contar o número de empréstimos por livro e ordena os livros pela quantidade de empréstimos em ordem decrescente ('DESC'). O sistema retorna: Status HTTP 200 (OK). A lista de livros mais emprestados com seus respectivos títulos e a quantidade de empréstimos.
Fluxo Alternativo:	Se ocorrer um erro durante a consulta (por exemplo, falha no banco de dados), o sistema retorna: o Status HTTP 500 (Erro Interno). o A mensagem de erro: "Erro ao listar livros mais emprestados"

Tratamento de exceções:	Retorna a mensagem de erro “Ocorreu um erro ao registrar empréstimo” com código de status 500, e uma mensagem é mostrada no console, informando mais detalhes do erro.
-------------------------	--

ID:	UC015
Título:	Usuários com empréstimo pendentes
Descrição:	Listar todos os usuários que têm empréstimos pendentes (não devolvidos), incluindo seus dados (nome, email) e informações sobre os empréstimos.
Ator Primário:	Usuário ou sistema (ex: bibliotecário ou administrador) que solicita a lista de usuários com empréstimos pendentes.
Pré-condição:	<ul style="list-style-type: none">• O banco de dados deve estar conectado corretamente.• Os modelos Usuario e Empréstimo devem estar configurados com o relacionamento adequado no Sequelize.
Pós-condição:	A lista de usuários com empréstimos pendentes é retornada, contendo o nome, o e-mail e os empréstimos não devolvidos de cada usuário.
Diagrama:	<pre>graph TD; Inicio([Inicio]) --> ReceberRequisicao[/Receber requisição/]; ReceberRequisicao --> ExecutarConsulta[Executar consulta no banco]; ExecutarConsulta --> UsuariosEncontrados{Usuários encontrados?}; UsuariosEncontrados -- Não --> RetornarListaVazia[retornar lista vazia e status 200]; RetornarListaVazia --> Fim([Fim]); UsuariosEncontrados -- Sim --> ErroDuranteExecucao{Erro durante execução?}; ErroDuranteExecucao -- Sim --> Status500[Status 500 e loga erro no console]; Status500 --> Fim; ErroDuranteExecucao -- Não --> Status200[Status 200 e dados da consulta]; Status200 --> Fim;</pre>
Fluxo Principal:	O ator envia uma requisição GET para o endpoint /usuarios/com-emprestimos-pendentes. O sistema inclui as informações sobre os empréstimos (que são pendentes de devolução), utilizando o parâmetro required: true para garantir que apenas os usuários com empréstimos pendentes sejam retornados. O sistema retorna: Status HTTP 200 (OK). A lista de usuários com

	empréstimos pendentes, incluindo seus dados (id, nome, email) e a quantidade de empréstimos pendentes.
Fluxo Alternativo:	Se ocorrer um erro durante a consulta (por exemplo, falha no banco de dados), o sistema retorna: o Status HTTP 500 (Erro Interno). o A mensagem de erro: "Erro ao listar usuários com empréstimos pendentes".
Tratamento de exceções:	<ul style="list-style-type: none">• Retorna a mensagem de erro "Erro ao gerar relatório de usuários com empréstimos pendentes" com código de status 500.• Loga o erro no console, incluindo detalhes da exceção para facilitar a depuração.