

# Casos de Uso

ID:	UC001
Título:	Registrar Empréstimo;
Descrição:	Permitir que o usuário registre empréstimos feitos pelos clientes;
Ator Primário:	Usuário;
Pré-condição:	<ul style="list-style-type: none"><li>O cliente deve estar registrado no sistema;</li><li>O livro deve estar no estado “disponível”;</li></ul>
Pós-condição:	<ul style="list-style-type: none"><li>O empréstimo ficará registrado no sistema;</li></ul>
Diagrama:	<pre>graph TD     UC[Realizar empréstimo] --&gt; UC1[procurar empréstimos realizados]     UC --&gt; UC2[deletar empréstimo]     UC --&gt; UC3[Criar empréstimo]     UC1 --&gt; A1[findAllLoans()]     A1 --&gt; E1[empréstimos procurados]     E1 --&gt; O1["json id&lt;br/&gt;lista de empréstimos"]     UC2 --&gt; A2[removeLoan()]     A2 --&gt; E2[empréstimo deletado]     E2 --&gt; O2["json id&lt;br/&gt;mensagem:&lt;br/&gt;"Empréstimo deletado com sucesso""]     UC3 --&gt; A3[createLoan()]     A3 --&gt; E3[empréstimo realizado]     E3 --&gt; O3["json id&lt;br/&gt;mensagem:&lt;br/&gt;"Empréstimo realizado com sucesso""]</pre>
Fluxo Principal:	O usuário solicita o registro do empréstimo através do id do livro, chamando a createLoan() e finalizando a solicitação;
Fluxo Opcional:	
Fluxo Alternativo:	O id do usuário não existe, o id do livro não existe, erro de exceção e exceder o limite de empréstimos;
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:	UC002
Título:	Gerenciar Usuários;
Descrição:	Permitir que o administrador gerencie os clientes;
Ator Primário:	Administrador;
Pré-condição:	<ul style="list-style-type: none"><li>O cliente não deve haver registro prévio;</li></ul>
Pós-condição:	<ul style="list-style-type: none"><li>O usuário deverá ser registrado;</li><li>O usuário deverá haver a opção de ter seus dados consultados;</li></ul>
Diagrama:	<pre>graph TD     UC[Gerenciar usuários]     UC --&gt; UC1[Procurar]     UC --&gt; UC2[criar usuário]     UC --&gt; UC3[atualizar usuário]     UC --&gt; UC4[deletar usuário]     UC1 --&gt; UC1_1[por todos os usuários]     UC1 --&gt; UC1_2[por um usuário]     UC1_1 --&gt; UC1_1_1[findAllClientes()]     UC1_1_1 --&gt; UC1_1_2[clientes procurados]     UC1_1_2 --&gt; UC1_1_3[/json id informações dos usuários/]     UC1_2 --&gt; UC1_2_1[findClient()]     UC1_2_1 --&gt; UC1_2_2[cliente procurado]     UC1_2_2 --&gt; UC1_2_3[/json id informações do usuário/]     UC2 --&gt; UC2_1[createClient()]     UC2_1 --&gt; UC2_2[Usuário Criado]     UC2_2 --&gt; UC2_3[/json id mensagem: "Usuário criado com sucesso"/]     UC3 --&gt; UC3_1[updateClient()]     UC3_1 --&gt; UC3_2[usuário atualizado]     UC3_2 --&gt; UC3_3[/json id mensagem: "Usuário atualizado com sucesso"/]     UC4 --&gt; UC4_1[deleteClient()]     UC4_1 --&gt; UC4_2[usuário deletado]     UC4_2 --&gt; UC4_3[/json id mensagem: "Usuário deletado com sucesso"/]</pre> <p>The diagram is a UML Use Case Diagram for the 'Gerenciar usuários' (Manage users) use case. The central use case is 'Gerenciar usuários'. It has four main branches: 'Procurar' (Search), 'criar usuário' (Create user), 'atualizar usuário' (Update user), and 'deletar usuário' (Delete user). The 'Procurar' use case has two sub-use cases: 'por todos os usuários' (for all users) and 'por um usuário' (for one user). 'por todos os usuários' leads to 'findAllClientes()', which leads to 'clientes procurados', which leads to a success message 'json id informações dos usuários'. 'por um usuário' leads to 'findClient()', which leads to 'cliente procurado', which leads to a success message 'json id informações do usuário'. 'criar usuário' leads to 'createClient()', which leads to 'Usuário Criado', which leads to a success message 'json id mensagem: "Usuário criado com sucesso"'. 'atualizar usuário' leads to 'updateClient()', which leads to 'usuário atualizado', which leads to a success message 'json id mensagem: "Usuário atualizado com sucesso"'. 'deletar usuário' leads to 'deleteClient()', which leads to 'usuário deletado', which leads to a success message 'json id mensagem: "Usuário deletado com sucesso"'. The diagram also shows a 'usuário sistema' (system user) use case that is connected to the central 'Gerenciar usuários' use case.</p>
Fluxo Principal:	O usuário é adicionado no banco de dados, chamando a createClient() e finalizando a solicitação
Fluxo Opcional:	
Fluxo Alternativo:	O id do usuário não existe;
Tratamento de exceções:	Retorna com uma mensagem de erro “Usuário não identificado” com código: console.log(error.message)

ID:	UC003
Título:	Gerenciar Livros;
Descrição:	Permitir que o usuário acompanhe e modifique o fluxo de processos dos livros;
Ator Primário:	Usuário;
Pré-condição:	<ul style="list-style-type: none"><li>• O livro novo não pode estar registrado no sistema;</li><li>• O livro previamente existente não pode estar atualizado no sistema;</li></ul>
Pós-condição:	<ul style="list-style-type: none"><li>• O livro novo ficará registrado no sistema;</li><li>• O livro novo deverá haver a opção de ser modificado;</li><li>• O livro previamente existente deverá ficar registrado no sistema;</li><li>• O livro previamente existente deverá haver a opção de ser modificado;</li></ul>
Diagrama:	<pre>graph TD     GerenciarLivros[Gerenciar livros]     GerenciarLivros --&gt; ProcurarTodos[procurar por todos os livros]     GerenciarLivros --&gt; ProcurarLivro[procurar livro]     GerenciarLivros --&gt; AtualizarLivro[atualizar livro]     GerenciarLivros --&gt; RemoverLivro[remover livro]     GerenciarLivros --&gt; AdicionarLivro[Adicionar livro]      ProcurarTodos --&gt; FindAllBooks[findAllBooks()]     FindAllBooks --&gt; LivrosProcurados[livros procurados]     LivrosProcurados --&gt; JSONTodos[json id lista de todos os livros]      ProcurarLivro --&gt; FindBook[findBook()]     FindBook --&gt; LivroEncontrado[livro encontrado]     LivroEncontrado --&gt; JSONInfo[json id informações do livro]      AtualizarLivro --&gt; UpdateBook[updateBook()]     UpdateBook --&gt; LivroAtualizado[livro atualizado]     LivroAtualizado --&gt; JSONMsgSucesso1[json id mensagem: "Livro atualizado com sucesso"]      RemoverLivro --&gt; RemoveBook[removeBook()]     RemoveBook --&gt; LivroRemovido[livro removido]     LivroRemovido --&gt; JSONMsgSucesso2[json id mensagem: "Livro removido com sucesso"]      AdicionarLivro --&gt; AddBook[addBook()]     AddBook --&gt; LivroAdicionado[Livro adicionado]     LivroAdicionado --&gt; JSONMsgSucesso3[json id mensagem: "Livro adicionado com sucesso"]</pre>
Fluxo Principal:	O livro é adicionado ao banco de dados, podendo ser deletado, atualizado e consultado; caso o livro já esteja incluso no banco de dados e precise ser atualizado, adicionado ou deletado, seu id é utilizado para tal evento;
Fluxo Opcional:	
Fluxo Alternativo:	O livro não existe, o livro não conseguiu ser adicionado, o livro não conseguiu ser deletado, o livro não conseguiu ser consultado e o livro está indisponível;
Tratamento de exceções:	Retorna a mensagem de erro “Erro ao buscar livro” com o status 500, caso haja fluxo alternativo; <pre>catch(err) {     res.status(500).json({ error: 'Erro ao buscar livro.' });     console.log(error.message); }</pre>

ID:	UC004
Título:	Gerenciar Relatórios;
Descrição:	Permitir que o usuário registre relatórios feitos pelo banco de dados do sistema;
Ator Primário:	Usuário;
Pré-condição:	<ul style="list-style-type: none"><li>• O relatório deve apresentar as atividades recentes feitas no sistema;</li></ul>
Pós-condição:	<ul style="list-style-type: none"><li>• O relatório ficará registrado no sistema;</li><li>• O relatório deverá gerar uma lista dos empréstimos ocorridos;</li><li>• O relatório deverá gerar uma lista com os livros onde houveram maior número de empréstimos;</li></ul>
Diagrama:	<pre>graph TD     Actor[usuário sistema] --&gt; UC[Gerenciar relatório]     UC --&gt; UC1[procurar por livros mais emprestados]     UC --&gt; UC2[mostrar os clientes devedores]     UC1 --&gt; UC1_1[mostBorrowed()]     UC1_1 --&gt; UC1_2[relatório de livros gerado]     UC1_2 --&gt; UC1_3[json id lista de todos os livros]     UC2 --&gt; UC2_1[clientsWithActiveLoans()]     UC2_1 --&gt; UC2_2[relatório de clientes gerado]     UC2_2 --&gt; UC2_3[json id lista de clientes devedores]</pre> <p>The diagram illustrates the 'Gerenciar relatório' use case. It starts with an actor 'usuário sistema' interacting with the central use case 'Gerenciar relatório'. This use case branches into two parallel paths. The left path involves 'procurar por livros mais emprestados', which leads to the 'mostBorrowed()' operation, then to a 'relatório de livros gerado', and finally to a 'json id lista de todos os livros' output. The right path involves 'mostrar os clientes devedores', which leads to the 'clientsWithActiveLoans()' operation, then to a 'relatório de clientes gerado', and finally to a 'json id lista de clientes devedores' output.</p>
Fluxo Principal:	O usuário solicita o registro do empréstimo através do id do livro, chamando a registrarEmprestimo() e finalizando a solicitação;
Fluxo Opcional:	
Fluxo Alternativo:	<ul style="list-style-type: none"><li>• O id do usuário não existe;</li><li>• o id do livro não existe,</li><li>• erro de exceção</li><li>• exceder o limite de empréstimos;</li></ul>

<b>Tratamento de exceções:</b>	Manda o status 500 “Ocorreu um erro ao registrar relatório” com o código:
--------------------------------	---