
Plano de Testes

Gerenciamento de Biblioteca

Versão 1.0

Equipe: Luciano Bruno, Eduardo José,
Kácio Silva, Adonai Ermínio, Tyago Ferreira

Histórico de Alterações

Data	Versão	Descrição	Autor
13/08/2024	1.0	Criação do documento	Luciano Bruno, Eduardo José, Kácio Silva, Adonai Ermínio, Tyago Ferreira
10/08 a 14/08	1.0	Configuração do Ambiente de Testes	Eduardo José, Tyago Ferreira
15/08	1.0	Desenvolvimento de Scripts de Teste	Luciano Bruno
17/08	1.0	Desenvolvimento de Scripts de Teste	Kácio Silva
20/08	1.0	Desenvolvimento de Scripts de Teste	Adonai Ermínio
27/08	1.0	Desenvolvimento de Scripts de Teste	Tyago Ferreira
30/08	1.0	Execução de testes gerais, Finalização e Entrega da Documentação	Eduardo José

Conteúdo

1	INTRODUÇÃO	3
2	REQUISITOS A TESTAR	3
2.1	ITERAÇÃO 1	3
2.2	ITERAÇÃO 2	3
2.3	ITERAÇÃO 3	3
3	TIPOS DE TESTE	3
3.1	ITERAÇÃO 1	3
3.2	ITERAÇÃO 2	3
4	RECURSOS	3
4.1	AMBIENTE DE TESTE – SOFTWARE & HARDWARE	3
4.2	FERRAMENTAS DE TESTE	3
5	CRONOGRAMA	3
6	REFERÊNCIAS	3

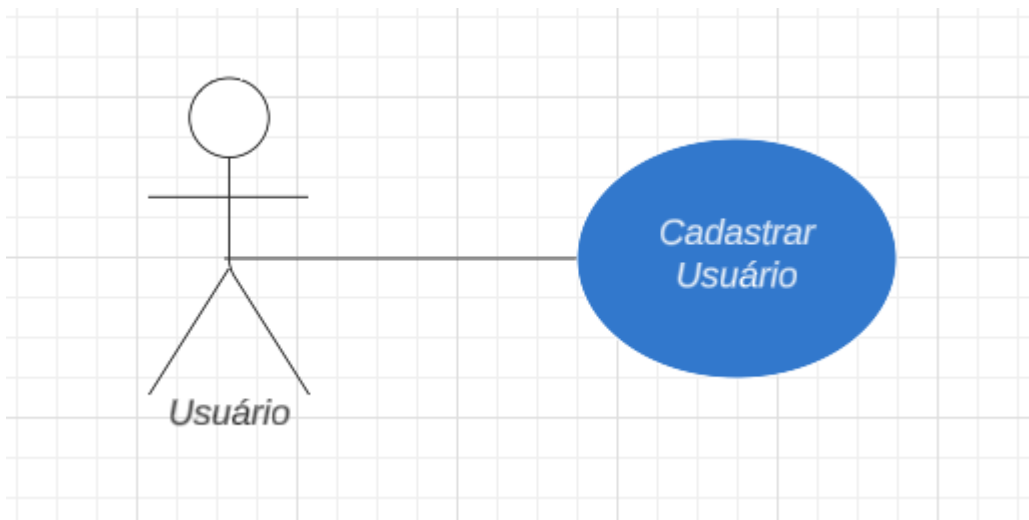
1 Introdução

O fluxo de testes, assim como os demais fluxos, está presente no processo de desenvolvimento de *software* ao longo de todas as suas fases, concentrando-se, no entanto, no planejamento dos testes na iteração inicial e no início de cada nova iteração e, durante as iterações, tendo seu foco no projeto e na execução dos testes, sobretudo nas iterações da fase de Construção.

Este documento descreve os requisitos a testar, os tipos de testes definidos para cada iteração, os recursos de hardware e software a serem empregados e o cronograma dos testes ao longo do projeto. As seções referentes aos requisitos, recursos e cronograma servem para permitir ao gerente do projeto acompanhar a evolução dos testes.

2 Requisitos a Testar

Caso de Uso 1: Cadastro de Usuário



Descrição: O sistema permite o cadastro de novos usuários. Os usuários fornecem suas informações pessoais e o sistema as armazena para futuras interações, como empréstimos e devoluções de livros.

Ator Principal: Usuário (ou bibliotecário)

Fluxo Principal:

1. O usuário solicita o cadastro no sistema.
2. O sistema solicita as informações necessárias: nome, CPF, e-mail, e telefone.
3. O usuário fornece os dados completos e válidos.
4. O sistema valida os dados fornecidos.
5. O sistema registra o usuário e exibe uma mensagem de sucesso.

Fluxos Alternativos:

Dados Incompletos:

1. O usuário não fornece todas as informações necessárias.
2. O sistema exibe uma mensagem de erro informando que todos os campos são obrigatórios.

Dados Inválidos (CPF):

1. O usuário fornece um CPF inválido.
2. O sistema exibe uma mensagem de erro informando que o CPF é inválido.

Dados Inválidos (E-mail):

1. O usuário fornece um e-mail inválido.
2. O sistema exibe uma mensagem de erro informando que o e-mail é inválido.

Erro de Banco de Dados (CPF ou E-mail já cadastrado):

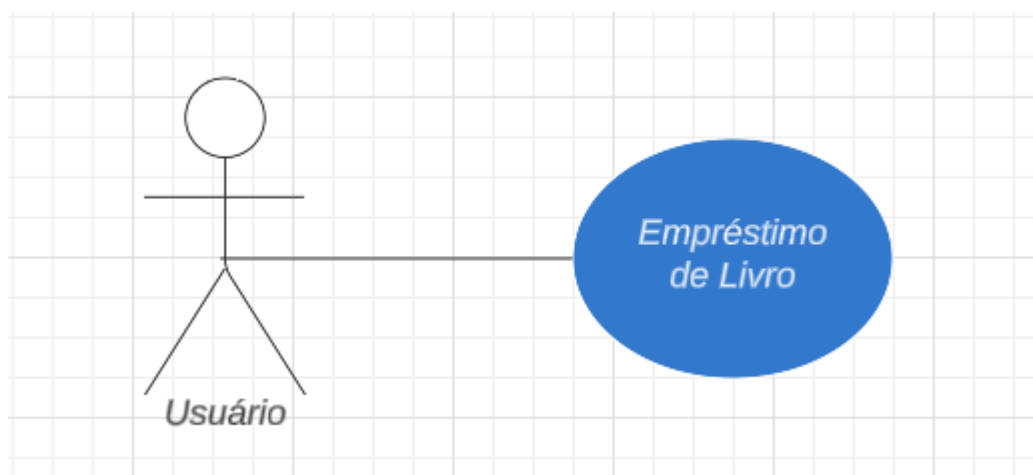
1. O sistema detecta que o CPF ou o e-mail fornecido já está cadastrado.
2. O sistema exibe uma mensagem de erro informando que o CPF ou e-mail já está cadastrado.

Plano de Testes

Cenários de Teste: Cadastro de Usuário

- **Cenário Principal: Cadastro de Usuário Válido**
 - **Descrição:** O usuário devolve um livro emprestado corretamente.
 - **Entrada:** Dados completos e válidos do usuário (nome, CPF, e-mail, telefone)
 - **Saída Esperada:** Mensagem de sucesso e usuário cadastrado no sistema
- **Cenário Alternativo: Cadastro de Usuário com Dados Incompletos**
 - **Descrição:** O usuário fornece dados incompletos para o cadastro.
 - **Entrada:** Dados incompletos (faltando e-mail)
 - **Saída Esperada:** Mensagem de erro informando que todos os campos são obrigatórios
- **Cenário Alternativo: Cadastro de Usuário com CPF Inválido**
 - **Entrada:** CPF inválido.
 - **Saída Esperada:** Mensagem de erro informando que o CPF é inválido.
- **Cenário Alternativo: Cadastro de Usuário com E-mail Inválido**
 - **Entrada:** E-mail inválido.
 - **Saída Esperada:** Mensagem de erro informando que o e-mail é inválido.
- **Cenário Alternativo: Erro de Banco de Dados ao Cadastrar Usuário**
 - **Entrada:** CPF ou e-mail já cadastrado.
 - **Saída Esperada:** Mensagem de erro informando que o CPF ou e-mail já está cadastrado.

Caso de Uso 2: Empréstimo de Livro



Descrição: O sistema permite que os usuários solicitem livros da biblioteca. O sistema verifica a disponibilidade do livro e registra o empréstimo.

Ator Principal: Usuário

Fluxo Principal:

1. O usuário solicita o empréstimo de um livro.
2. O sistema solicita a identificação do livro e do usuário.
3. O usuário fornece as identificações necessárias.
4. O sistema verifica:
 - Se o livro existe no sistema.
 - Se o livro está disponível (não emprestado).
 - Se o usuário está apto a emprestar (sem pendências).
5. O sistema registra o empréstimo e exibe uma mensagem de sucesso com a data prevista para devolução.

Fluxos Alternativos:

Livro Não Encontrado:

1. O sistema não encontra o livro com o ID fornecido.
2. O sistema exibe uma mensagem de erro informando que o livro não foi encontrado.

Livro Não Disponível:

1. O livro solicitado não está disponível (já emprestado).
2. O sistema exibe uma mensagem de erro informando que o livro não está disponível.

Usuário Inadimplente:

1. O usuário possui pendências (multas ou empréstimos não devolvidos).
2. O sistema exibe uma mensagem de erro informando que o usuário não pode emprestar livros até regularizar a situação.

Nenhum ID Fornecido:

1. O usuário não fornece o ID do livro ou do usuário.
2. O sistema exibe uma mensagem de erro informando que a identificação do usuário e do livro é obrigatória.

Erro de Banco de Dados:

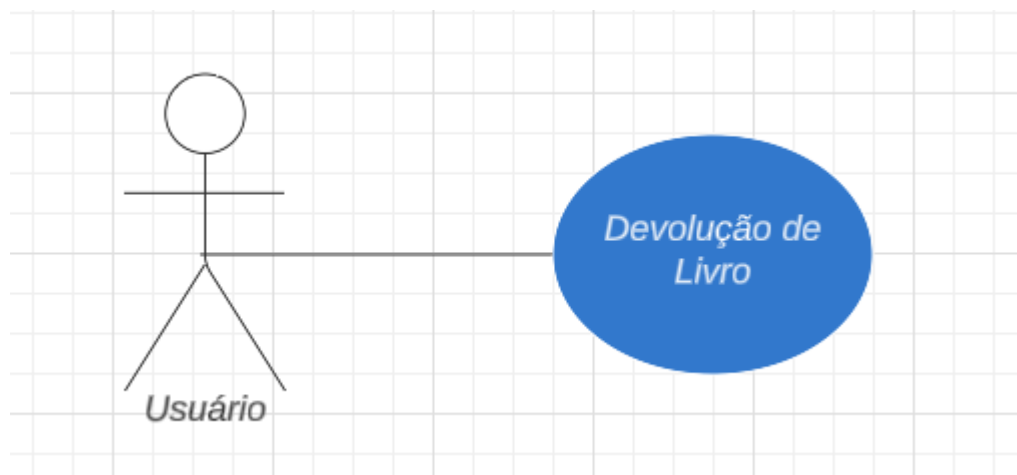
1. Ocorreu um erro ao acessar o banco de dados durante o processo de empréstimo.
2. O sistema exibe uma mensagem de erro informando sobre o problema com o banco de dados.

Plano de Testes

Cenários de Teste: Empréstimo de Livro

- **Cenário Principal: Empréstimo de Livro Disponível**
 - **Descrição:** O usuário solicita o empréstimo de um livro que está disponível.
 - **Entrada:** Identificação do usuário, identificação do livro disponível.
 - **Saída Esperada:** Mensagem de sucesso e registro de empréstimo no sistema com a data de devolução.
- **Cenário Alternativo: Livro Não Encontrado**
 - **Descrição:** O usuário solicita o empréstimo de um livro que não existe no sistema.
 - **Entrada:** Identificação do usuário, identificação do livro inexistente.
 - **Saída Esperada:** Mensagem de erro informando que o livro não foi encontrado.
- **Cenário Alternativo: Livro Não Disponível**
 - **Descrição:** O usuário solicita o empréstimo de um livro que não está disponível (já emprestado).
 - **Entrada:** Identificação do usuário, identificação do livro já emprestado.
 - **Saída Esperada:** Mensagem de erro informando que o livro não está disponível.
- **Cenário Alternativo: Nenhum ID Fornecido**
 - **Descrição:** O usuário não fornece a identificação do livro ou do usuário.
 - **Entrada:** Nenhuma identificação do livro ou do usuário.
 - **Saída Esperada:** Mensagem de erro informando que o ID do usuário e do livro são obrigatórios.
- **Cenário Alternativo: Erro de Banco de Dados**
 - **Descrição:** Ocorre um erro ao acessar o banco de dados durante o empréstimo.
 - **Entrada:** Identificação do usuário, identificação do livro.
 - **Saída Esperada:** Mensagem de erro informando sobre o problema com o banco de dados.

Caso de Uso 3: Devolução de Livro



Descrição: O sistema permite que os usuários devolvam livros emprestados. O sistema atualiza o status do livro e verifica se o usuário precisa pagar multa por devolução atrasada.

Ator Principal: Usuário

Fluxo Principal:

1. O usuário solicita a devolução de um livro.
2. O sistema solicita a identificação do livro e do usuário.
3. O usuário fornece as identificações necessárias.
4. O sistema verifica:
Se existe um registro de empréstimo para o livro e o usuário fornecidos.
Se o livro está atualmente emprestado.
5. O sistema registra a devolução e exibe uma mensagem de sucesso..

Fluxos Alternativos:

Livro Não Encontrado:

1. O sistema não encontra o livro com o ID fornecido.
2. O sistema exibe uma mensagem de erro informando que o livro não foi encontrado.

Nenhum ID Fornecido:

1. O usuário não fornece o ID do livro ou do usuário.
2. O sistema exibe uma mensagem de erro informando que a identificação do usuário e do livro é obrigatória.

Empréstimo Não Encontrado:

1. Não existe um registro de empréstimo para o livro e usuário fornecidos.
2. O sistema exibe uma mensagem de erro informando que não há registro de empréstimo para esse livro e usuário.

Erro de Banco de Dados:

1. Ocorreu um erro ao acessar o banco de dados durante o processo de devolução.
2. O sistema exibe uma mensagem de erro informando sobre o problema com o banco de dados.

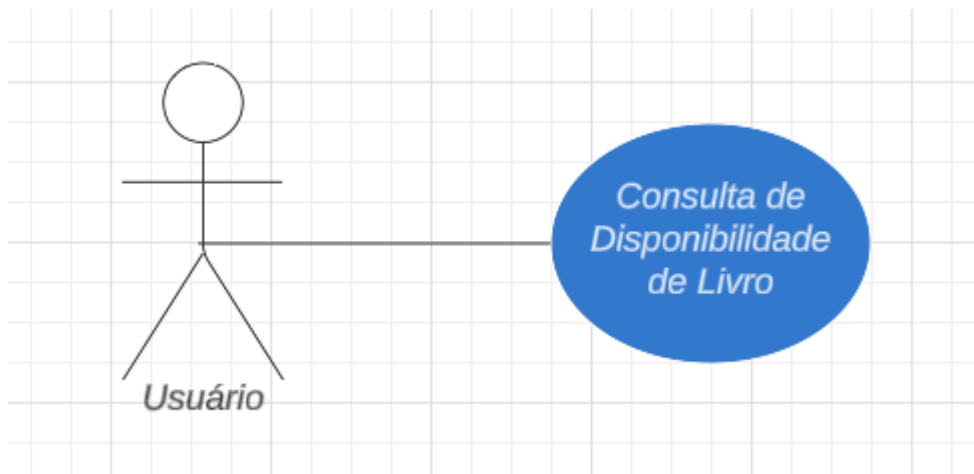
Plano de Testes

Cenários de Teste: Devolução de Livro

- **Cenário Principal: Devolução bem-sucedida**
 - **Descrição:** O usuário devolve um livro que está atualmente emprestado.
 - **Entrada:** Identificação do usuário, identificação do livro.
 - **Saída Esperada:** Mensagem de sucesso e registro da devolução no sistema.
- **Cenário Alternativo: Livro Não Encontrado**
 - **Descrição:** O usuário tenta devolver um livro que não existe no sistema.
 - **Entrada:** Identificação do usuário, identificação de um livro inexistente.
 - **Saída Esperada:** Mensagem de erro informando que o livro não foi encontrado.
- **Cenário Alternativo: Nenhum ID Fornecido**
 - **Descrição:** O usuário não fornece a identificação do livro ou do usuário.
 - **Entrada:** Nenhuma identificação do livro ou do usuário.

- **Saída Esperada:** Mensagem de erro informando que a identificação do usuário e do livro é obrigatória.
- **Cenário Alternativo: Empréstimo Não Encontrado**
 - **Descrição:** O usuário tenta devolver um livro para o qual não há registro de empréstimo.
 - **Entrada:** Identificação do usuário, identificação do livro.
 - **Saída Esperada:** Mensagem de erro informando que não há registro de empréstimo para esse livro e usuário.
- **Cenário Alternativo: Erro de Banco de Dados**
 - **Descrição:** Ocorre um erro ao acessar o banco de dados durante a devolução.
 - **Entrada:** Identificação do usuário, identificação do livro.
 - **Saída Esperada:** Mensagem de erro informando sobre o problema com o banco de dados.

Caso de Uso 4: Consulta de Disponibilidade de Livro



Descrição: O sistema permite que os usuários consultem a disponibilidade de livros na biblioteca. O sistema retorna se o livro está disponível para empréstimo ou não.

Ator Principal: Usuário

Fluxo Principal:

1. O usuário solicita a consulta de disponibilidade de um livro.
2. O sistema solicita a identificação do livro.
3. O usuário fornece a identificação do livro.
4. O sistema verifica a disponibilidade do livro no banco de dados.
5. O sistema exibe a disponibilidade do livro.

Fluxos Alternativos:

Livro Não Encontrado:

1. O sistema não encontra o livro com o ID fornecido.
2. O sistema exibe uma mensagem de erro informando que o livro não foi encontrado.

Nenhum ID Fornecido:

1. O usuário não fornece o ID do livro.
2. O sistema exibe uma mensagem de erro informando que o ID do livro é obrigatório.

Erro de Banco de Dados:

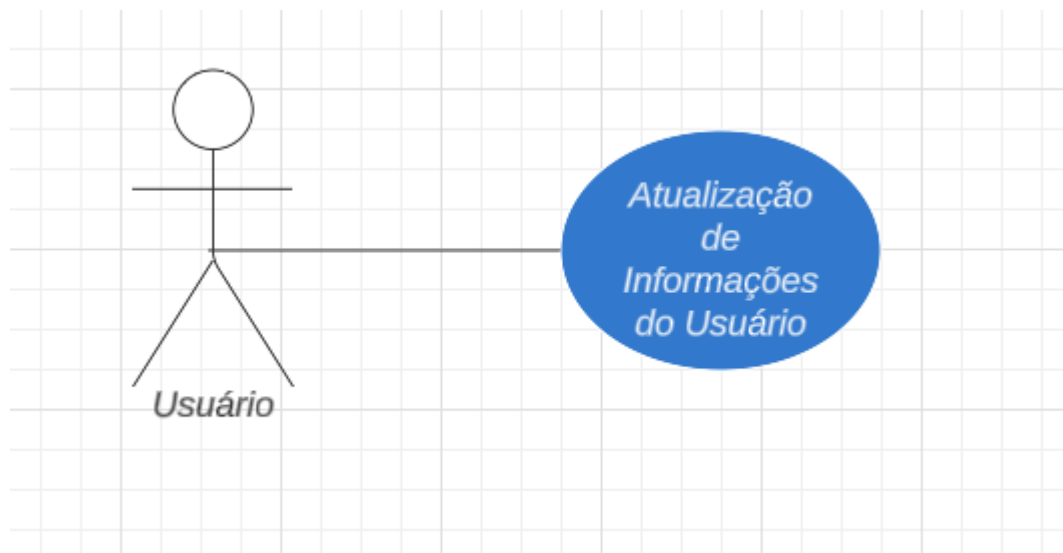
1. Ocorreu um erro ao acessar o banco de dados durante a consulta.
2. O sistema exibe uma mensagem de erro informando sobre o problema com o banco de dados.

Plano de Testes

Cenários de Teste: Consulta de Disponibilidade de Livro

- **Cenário Principal: Consulta de Livro Disponível**
 - **Descrição:** O usuário consulta a disponibilidade de um livro que está disponível para empréstimo.
 - **Entrada:** Identificação do livro.
 - **Saída Esperada:** Mensagem indicando que o livro está disponível.
- **Cenário Alternativo: Livro Não Encontrado**
 - **Descrição:** O usuário consulta a disponibilidade de um livro que não está registrado no sistema.
 - **Entrada:** Identificação do livro.
 - **Saída Esperada:** Mensagem indicando que o livro não foi encontrado.
- **Cenário Alternativo: Nenhum ID Fornecido**
 - **Descrição:** O usuário não fornece a identificação do livro ao solicitar a consulta.
 - **Entrada:** Nenhum ID fornecido.
 - **Saída Esperada:** Mensagem indicando que o ID do livro é obrigatório.
- **Cenário Alternativo: Erro de Banco de Dados**
 - **Descrição:** Ocorre um erro de banco de dados durante a consulta de disponibilidade.
 - **Entrada:** Identificação do livro.
 - **Saída Esperada:** Mensagem indicando que ocorreu um erro de banco de dados.

Caso de Uso 5: Atualização de Informações do Usuário



Descrição: O sistema permite que um usuário atualize seus dados pessoais. O usuário pode alterar informações como e-mail, telefone, ou endereço.

Ator Principal: Usuário (ou bibliotecário)

Fluxo Principal:

1. O usuário solicita a atualização de suas informações pessoais.
2. O sistema solicita os dados atuais e os novos dados desejados.
3. O usuário fornece as novas informações.
4. O sistema valida as novas informações:
Verifica se o e-mail fornecido está em um formato válido.
Verifica se as demais informações são válidas e completas.
5. Se as informações forem válidas, o sistema atualiza o registro do usuário no banco de dados.
6. O sistema exibe uma mensagem de sucesso indicando que as informações foram atualizadas.

Fluxos Alternativos:

Nenhuma informação fornecida para atualização:

1. O usuário não fornece nenhuma informação nova para atualizar.
2. O sistema exibe uma mensagem de erro indicando que nenhuma informação foi fornecida para atualizar.

Dados Inválidos:

1. O usuário fornece dados inválidos (por exemplo, um e-mail com formato incorreto).
2. O sistema exibe uma mensagem de erro indicando que os dados são inválidos.

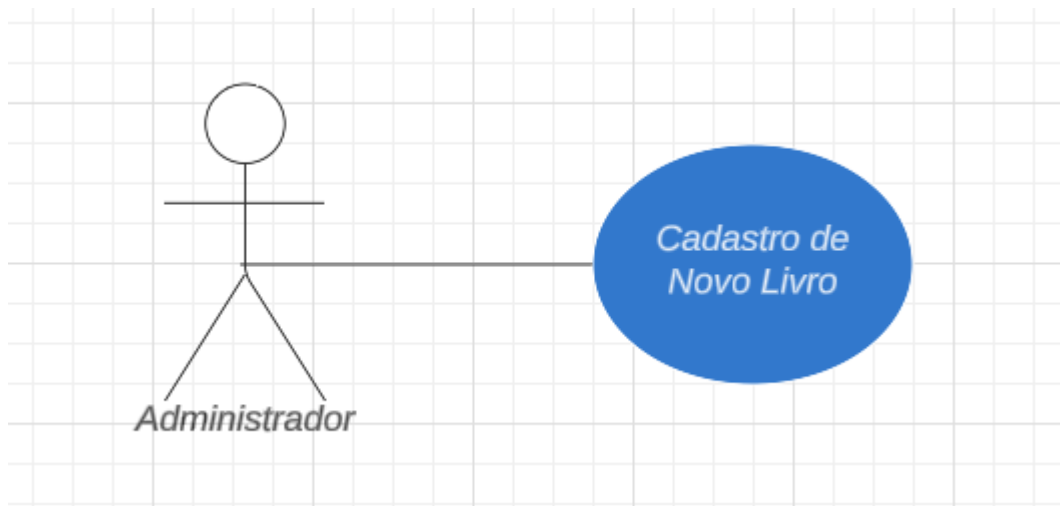
Erro de Banco de Dados:

1. Ocorreu um erro no banco de dados durante a tentativa de atualização.
2. O sistema exibe uma mensagem de erro indicando que houve um problema ao atualizar as informações.

Cenários de Teste: Atualização de Informações do Usuário

- **Atualização com Dados Válidos:**
 - **Descrição:** Atualizar as informações do usuário com dados válidos.
 - **Entrada:** Novo nome: "Carlos Silva", Novo e-mail: "carlos.silva@example.com", Novo telefone: "(11) 91234-5678"
 - **Saída Esperada:** Mensagem de sucesso "Usuário atualizado com sucesso."
- **Nenhuma Informação Fornecida:**
 - **Descrição:** O usuário tenta atualizar os dados sem fornecer nenhuma nova informação.
 - **Entrada:** Nenhuma nova informação fornecida.
 - **Saída Esperada:** Mensagem de erro "Nenhuma informação para atualizar."
- **Atualização com E-mail Inválido:**
 - **Descrição:** Tentar atualizar o e-mail do usuário com um formato inválido.
 - **Entrada:** Novo e-mail: "carlos.silvaexample.com"
 - **Saída Esperada:** Mensagem de erro "E-mail inválido."
- **Erro de Banco de Dados:**
 - **Descrição:** Simular um erro de banco de dados durante a tentativa de atualização.
 - **Entrada:** Novo nome: "Carlos Silva"
 - **Saída Esperada:** Mensagem de erro "Erro de banco de dados."

Caso de Uso 6: Cadastro de Novo Livro



Descrição: O sistema permite o cadastro de novos livros na biblioteca, incluindo detalhes como título, autor, ISBN e categoria.

Ator Principal: Administrador

Fluxo Principal:

1. O Administrador solicita o cadastro de um novo livro.
2. O sistema solicita informações sobre o livro (título, autor, ISBN, categoria).
3. O Administrador fornece os dados do livro.
4. O sistema valida os dados e registra o novo livro.
5. Se os dados forem válidos, o sistema registra o novo livro no banco de dados.
6. O sistema exibe uma mensagem de sucesso indicando que o livro foi cadastrado com sucesso.

Fluxos Alternativos:

Campos Obrigatórios Faltando:

1. O Administrador não preenche todos os campos obrigatórios.
2. O sistema exibe uma mensagem de erro indicando que todos os campos são obrigatórios.

ISBN Inválido:

1. O administrador fornece um ISBN em formato incorreto.
2. O sistema exibe uma mensagem de erro indicando que o ISBN é inválido.

ISBN Já Cadastrado:

1. O Administrador tenta cadastrar um livro com um ISBN que já existe no banco de dados.
2. O sistema exibe uma mensagem de erro indicando que o ISBN já foi cadastrado.

Erro de Banco de Dados:

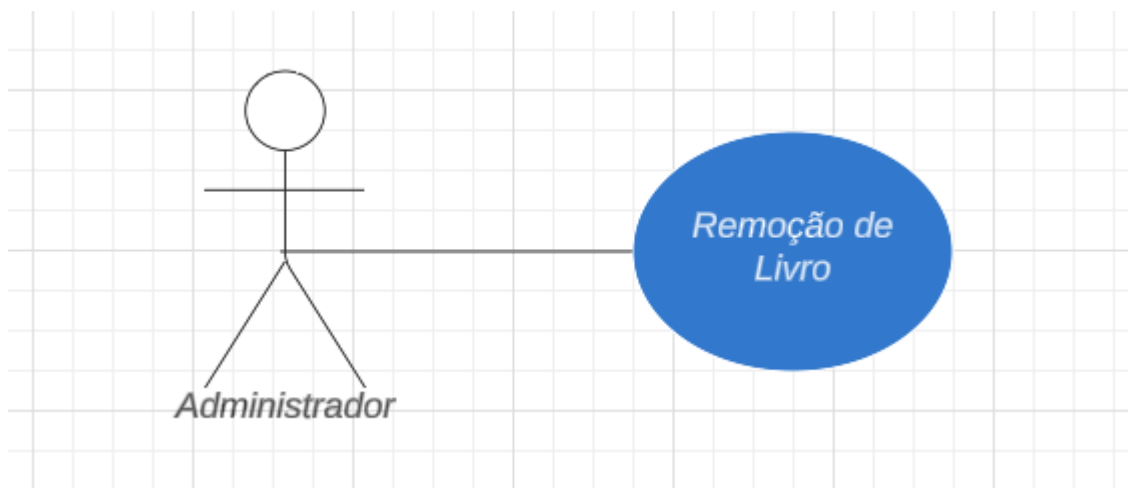
1. Ocorreu um erro no banco de dados durante a tentativa de cadastro do livro.
2. O sistema exibe uma mensagem de erro indicando que houve um problema ao cadastrar o livro.

Cenários de Teste: Cadastro de Novo Livro

- **Cadastro com Dados Válidos:**
 - **Descrição:** Cadastrar um novo livro com informações válidas.
 - **Entrada:** Título: "O Senhor dos Anéis", Autor: "J.R.R. Tolkien", ISBN: "978-3-16-148410-0", Categoria: "Fantasia"
 - **Saída Esperada:** Mensagem de sucesso "Livro cadastrado com sucesso."
- **Campos Obrigatórios Faltando:**
 - **Descrição:** Tentar cadastrar um livro sem preencher todos os campos obrigatórios.
 - **Entrada:** Título: "", Autor: "J.K. Rowling", ISBN: "978-3-16-148410-0", Categoria: "Fantasia"

- **Saída Esperada:** Mensagem de erro "Todos os campos são obrigatórios."
- **ISBN Inválido:**
 - **Descrição:** Tentar cadastrar um livro com ISBN em formato incorreto.
 - **Entrada:** Título: "Harry Potter e a Pedra Filosofal", Autor: "J.K. Rowling", ISBN: "978-3-16-148410-XYZ", Categoria: "Fantasia"
 - **Saída Esperada:** Mensagem de erro "ISBN inválido."
- **ISBN Já Cadastrado:**
 - **Descrição:** Tentar cadastrar um livro com um ISBN que já está cadastrado no sistema.
 - **Entrada:** Título: "O Hobbit", Autor: "J.R.R. Tolkien", ISBN: "978-0-345-33968-3", Categoria: "Fantasia"
 - **Saída Esperada:** Mensagem de erro "ISBN já cadastrado."

Caso de Uso 7: Remoção de Livro



Descrição: O sistema permite a remoção de livros que não estão mais disponíveis na biblioteca, seja por perda, dano ou outras razões.

Ator Principal: Administrador

Fluxo Principal:

1. O Administrador solicita a remoção de um livro.
2. O sistema solicita a identificação do livro a ser removido.
3. O Administrador fornece a identificação do livro.
4. O sistema valida a existência do livro no banco de dados.
5. Se o livro existir, o sistema o remove do cadastro.
6. O sistema exibe uma mensagem de sucesso indicando que o livro foi removido.

Fluxos Alternativos: Remoção de Livro

Livro Não Encontrado:

1. O administrador fornece a identificação de um livro que não está no sistema.
2. O sistema exibe uma mensagem de erro indicando que o livro não foi encontrado.

Nenhum ID Fornecido:

1. O administrador não fornece um ID válido para o livro a ser removido.
2. O sistema exibe uma mensagem de erro indicando que o ID do livro é obrigatório.

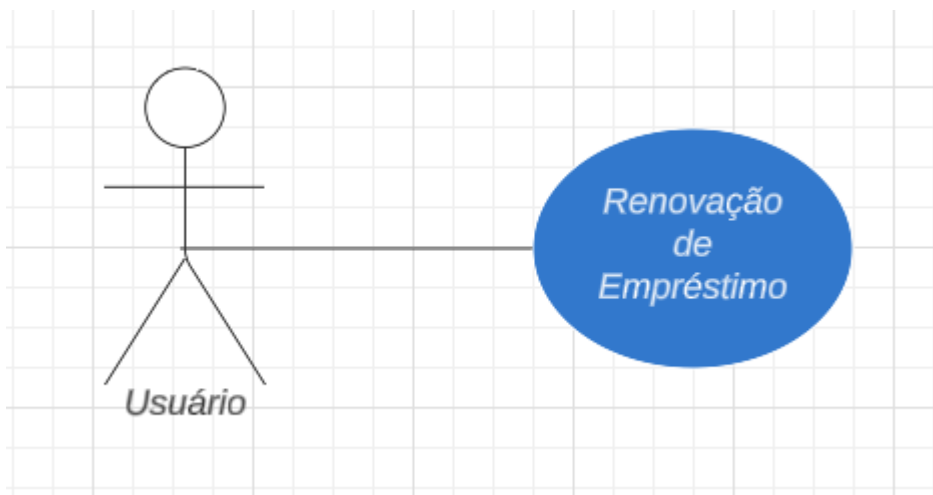
Erro de Banco de Dados:

1. Ocorreu um erro no banco de dados durante a tentativa de remover o livro.
2. O sistema exibe uma mensagem de erro indicando que houve um problema ao tentar remover o livro.

Cenários de Teste:

- **Remoção de Livro Existente:**
 - **Descrição:** Remover um livro existente da biblioteca.
 - **Entrada:** ID do livro: 1
 - **Saída Esperada:** Mensagem de sucesso "Livro removido com sucesso."
- **Remoção de Livro Não Encontrado:**
 - **Descrição:** Tentar remover um livro que não está no sistema.
 - **Entrada:** ID do livro: 999 (ID inexistente)
 - **Saída Esperada:** Mensagem de erro "Livro não encontrado."
- **Nenhum ID Fornecido:**
 - **Descrição:** Tentar remover um livro sem fornecer um ID.
 - **Entrada:** ID do livro: None
 - **Saída Esperada:** Mensagem de erro "ID do livro é obrigatório."
- **Erro de Banco de Dados ao Remover:**
 - **Descrição:** Ocorre um erro no banco de dados durante a tentativa de remoção do livro.
 - **Entrada:** ID do livro: 1
 - **Saída Esperada:** Mensagem de erro "Erro de banco de dados: <detalhes do erro>."

Caso de Uso 8: Renovação de Empréstimo



Descrição: O sistema permite que os usuários renovem o prazo de empréstimo de um livro, se permitido pelas políticas da biblioteca.

Ator Principal: Usuário

Fluxo Principal:

1. O usuário solicita a renovação do empréstimo de um livro.
2. O sistema solicita a identificação do livro e do usuário.
3. O usuário fornece as identificações necessárias.
4. O sistema verifica a possibilidade de renovação (por exemplo, se o livro está atualmente emprestado ao usuário e se as condições de renovação são atendidas).
5. Se permitido, o sistema atualiza a data de devolução do livro.
6. O sistema exibe uma mensagem de sucesso indicando que o empréstimo foi renovado.

Fluxos Alternativos:

Renovação Não Permitida:

1. O sistema verifica que o livro não pode ser renovado (por exemplo, já foi renovado o número máximo de vezes ou há uma reserva pendente).
2. O sistema exibe uma mensagem de erro indicando que a renovação não é permitida.

Livro Não Encontrado:

1. O sistema não encontra o livro com a identificação fornecida.
2. O sistema exibe uma mensagem de erro indicando que o livro não foi encontrado.

Livro Não Emprestado:

1. O sistema verifica que o livro não está emprestado ao usuário que solicitou a renovação.

2. O sistema exibe uma mensagem de erro indicando que o livro não está emprestado a este usuário.

Nenhum ID Fornecido:

1. O usuário não fornece um ID válido para o livro ou usuário.
2. O sistema exibe uma mensagem de erro indicando que o ID do livro e do usuário são obrigatórios.

Erro de Banco de Dados:

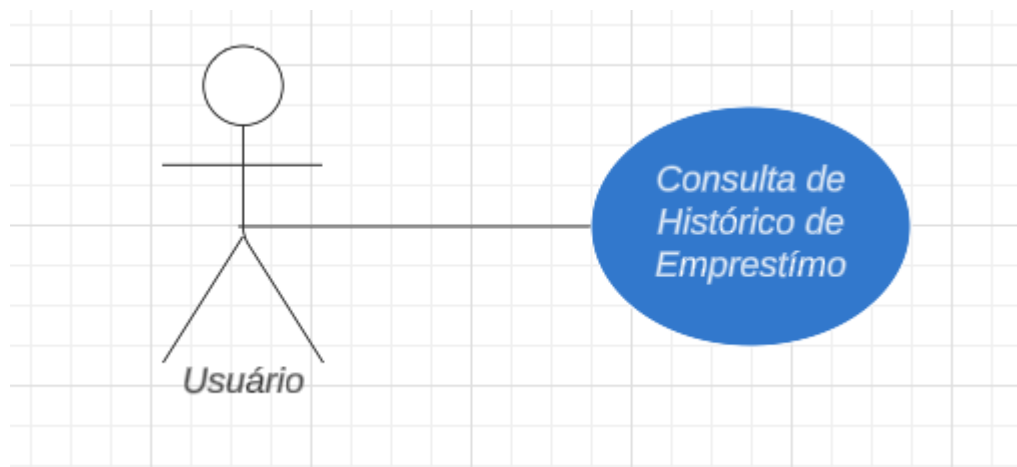
1. Ocorreu um erro no banco de dados durante a tentativa de renovar o empréstimo.
2. O sistema exibe uma mensagem de erro indicando que houve um problema ao tentar renovar o empréstimo.

Cenários de Teste: Renovação de Empréstimo

- **Renovação Bem-Sucedida:**
 - **Descrição:** Renovar o prazo de empréstimo de um livro com sucesso.
 - **Entrada:** Identificação do livro: LMN456, Identificação do usuário: 345678
 - **Saída Esperada:** Mensagem de sucesso "Empréstimo renovado com sucesso" e nova data de devolução registrada.
- **Renovação Não Permitida:**
 - **Descrição:** Tentar renovar um empréstimo que não pode ser renovado.
 - **Entrada:** Identificação do livro: XYZ789, Identificação do usuário: 789012
 - **Saída Esperada:** Mensagem de erro "Renovação não permitida."
- **Livro Não Encontrado:**
 - **Descrição:** Tentar renovar o empréstimo de um livro que não está registrado no sistema.
 - **Entrada:** Identificação do livro: 1
 - **Saída Esperada:** Mensagem de erro "Livro não encontrado."
- **Livro Não Empréstado:**
 - **Descrição:** Tentar renovar um livro que não está emprestado ao usuário que solicitou a renovação.
 - **Entrada:** Identificação do livro: 1, Identificação do usuário: 1
 - **Saída Esperada:** Mensagem de erro "O livro não está registrado como emprestado para este usuário."
- **Nenhum ID Fornecido:**
 - **Descrição:** Tentar renovar um empréstimo sem fornecer um ID de livro ou usuário.
 - **Entrada:** Identificação do livro: None, Identificação do usuário: None
 - **Saída Esperada:** Mensagem de erro "ID do usuário e do livro são obrigatórios."
- **Erro de Banco de Dados ao Renovar:**
 - **Descrição:** Ocorre um erro no banco de dados durante a tentativa de renovar o empréstimo.
 - **Entrada:** Identificação do livro: 1, Identificação do usuário: 1

- **Saída Esperada:** Mensagem de erro "Erro de banco de dados: <detalhes do erro>."

Caso de Uso 9: Consulta de Histórico de Empréstimos



Descrição: O sistema permite que os usuários consultem o histórico de empréstimos de livros, incluindo informações sobre livros emprestados e devolvidos.

Ator Principal: Usuário

Fluxo Principal:

1. O usuário solicita a consulta do histórico de empréstimos.
2. O sistema solicita a identificação do usuário.
3. O usuário fornece a identificação necessária.
4. O sistema recupera e exibe o histórico de empréstimos do usuário.

Fluxos Alternativos:

Histórico Não Encontrado:

1. O usuário não possui histórico de empréstimos.
2. O sistema exibe uma mensagem informando que não há registros.

Nenhum ID Fornecido:

1. O usuário não fornece um ID válido.
2. O sistema exibe uma mensagem de erro indicando que o ID do usuário é obrigatório.

Erro de Banco de Dados:

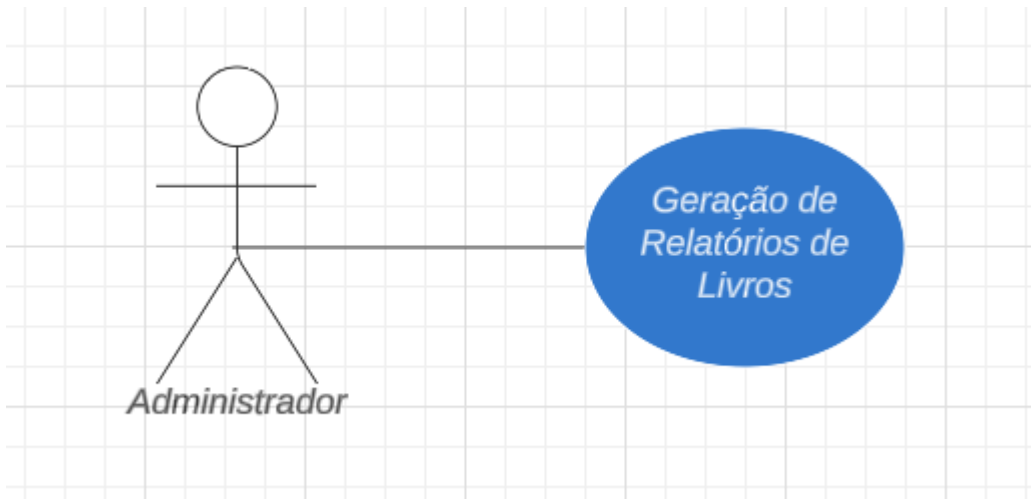
1. Ocorreu um erro no banco de dados durante a tentativa de consultar o histórico.

2. O sistema exibe uma mensagem de erro indicando que houve um problema ao tentar recuperar o histórico.

Cenários de Teste: Consulta de Histórico de Empréstimos

- **Consulta de Histórico com Empréstimos:**
 - **Descrição:** Consultar o histórico de empréstimos de um usuário com registros.
 - **Entrada:** Identificação do usuário: 123456
 - **Saída Esperada:** Lista de livros emprestados e devolvidos pelo usuário.
- **Consulta de Histórico Sem Empréstimos:**
 - **Descrição:** Consultar o histórico de empréstimos de um usuário sem registros.
 - **Entrada:** Identificação do usuário: 789012
 - **Saída Esperada:** Mensagem informando que não há histórico de empréstimos.
- **Nenhum ID Fornecido:**
 - **Descrição:** Tentar consultar o histórico de empréstimos sem fornecer um ID de usuário.
 - **Entrada:** Identificação do usuário: None
 - **Saída Esperada:** Mensagem de erro "ID do usuário é obrigatório."
- **Erro de Banco de Dados ao Consultar:**
 - **Descrição:** Ocorre um erro no banco de dados durante a tentativa de consultar o histórico de empréstimos.
 - **Entrada:** Identificação do usuário: 123456
 - **Saída Esperada:** Mensagem de erro "Erro de banco de dados: <detalhes do erro>."

Caso de Uso 10: Geração de Relatórios de Livros



Descrição: O sistema permite que os bibliotecários gerem relatórios sobre os livros da biblioteca, como relatórios de livros emprestados, disponíveis, ou com atraso.

Ator Principal: Administrador

Fluxo Principal:

1. O Administrador solicita a geração de um relatório.
2. O sistema solicita o tipo de relatório desejado (livros emprestados, disponíveis, com atraso).
3. O Administrador fornece o tipo de relatório.
4. O sistema gera e exibe o relatório solicitado.

Fluxos Alternativos:

Relatório Não Encontrado:

1. Não há dados para o tipo de relatório solicitado.
2. O sistema exibe uma mensagem informando que não há dados disponíveis.

Tipo de Relatório Inválido:

1. O administrador fornece um tipo de relatório que não é reconhecido pelo sistema.
2. O sistema exibe uma mensagem de erro indicando que o tipo de relatório é inválido.

Erro de Banco de Dados:

1. Ocorreu um erro no banco de dados durante a tentativa de gerar o relatório.
2. O sistema exibe uma mensagem de erro indicando que houve um problema ao tentar gerar o relatório.

Cenários de Teste: Geração de Relatórios de Livros

- **Geração de Relatório de Livros Emprestados:**
 - **Descrição:** Gerar um relatório contendo a lista de livros emprestados.
 - **Entrada:** Tipo de relatório: Livros Emprestados
 - **Saída Esperada:** Relatório contendo a lista de livros atualmente emprestados.
- **Geração de Relatório de Livros Disponíveis:**
 - **Descrição:** Gerar um relatório contendo a lista de livros disponíveis na biblioteca.
 - **Entrada:** Tipo de relatório: Livros Disponíveis
 - **Saída Esperada:** Relatório contendo a lista de livros disponíveis na biblioteca.
- **Geração de Relatório de Livros com Devoluções em Atraso:**
 - **Descrição:** Gerar um relatório contendo a lista de livros com devoluções em atraso.
 - **Entrada:** Tipo de relatório: Livros com Atraso
 - **Saída Esperada:** Relatório contendo a lista de livros com devoluções em atraso.
- **Tipo de Relatório Inválido:**
 - **Descrição:** Fornecer um tipo de relatório inválido.
 - **Entrada:** Tipo de relatório: invalido
 - **Saída Esperada:** Mensagem de erro "Tipo de relatório inválido."
- **Nenhum Dado Disponível para o Relatório Solicitado:**

- **Descrição:** Não há dados disponíveis para o tipo de relatório solicitado.
- **Entrada:** Tipo de relatório: Livros Emprestados
- **Saída Esperada:** Mensagem informando que não há dados disponíveis para o relatório solicitado.
- **Erro de Banco de Dados ao Gerar Relatório:**
 - **Descrição:** Ocorre um erro de banco de dados durante a tentativa de gerar o relatório.
 - **Entrada:** Tipo de relatório: Livros Emprestados
 - **Saída Esperada:** Mensagem de erro "Erro de banco de dados: <detalhes do erro>."

Esta seção contém os requisitos que são objetos dos testes a serem realizados. Esses requisitos são divididos, por iteração, em casos de uso e requisitos não funcionais conforme descrito abaixo.

2.1 Iteração 1

Casos de Uso

Identificador do Caso de Uso	Nome do Caso de Uso
CDU1	Cadastro de Usuário
CDU2	Empréstimo de Livro
CDU3	Devolução de Livro
CDU4	Consulta de Disponibilidade de Livro
CDU5	Atualização de Informações do Usuário

2.2 Iteração 2

Casos de Uso

Identificador do Caso de Uso	Nome do Caso de Uso
CDU6	Cadastro de Novo Livro
CDU7	Remoção de Livro
CDU8	Renovação de Empréstimo
CDU9	Consulta de Histórico de Empréstimo
CDU10	Geração de Relatórios de Livros

3 Tipos de Teste

3.1 Iteração 1

Objetivo:	Nesta iteração, serão testados os requisitos mais críticos do sistema, focando na robustez da
------------------	---

	arquitetura e na funcionalidade básica. O principal objetivo é garantir que os componentes individuais funcionem conforme o esperado.
Técnica:	<input checked="" type="checkbox"/> Manual <input type="checkbox"/> Automática
Estágio do teste: <input type="checkbox"/> Integração <input type="checkbox"/> Sistema <input checked="" type="checkbox"/> Unidade <input type="checkbox"/> Aceitação	Abordagem do teste <input checked="" type="checkbox"/> Caixa branca <input type="checkbox"/> Caixa preta
Responsável(is):	Programadores

3.2 Iteração 2

Objetivo:	Na segunda iteração, o foco será a integração entre os módulos desenvolvidos e a validação de funcionalidades mais complexas. O objetivo é garantir que os componentes funcionem corretamente em conjunto.
Técnica:	<input checked="" type="checkbox"/> Manual <input type="checkbox"/> Automática
Estágio do teste: <input type="checkbox"/> Integração <input type="checkbox"/> Sistema <input checked="" type="checkbox"/> Unidade <input type="checkbox"/> Aceitação	Abordagem do teste <input checked="" type="checkbox"/> Caixa branca <input type="checkbox"/> Caixa preta
Responsável(is):	Equipe de QA

4 Recursos

De extrema importância para o bom andamento dos testes, os recursos a serem utilizados durante os testes são descritos nessa seção. Os recursos estão divididos nas subseções que se seguem.

4.1 Ambiente de Teste – Software & Hardware

Hardware: Notebook

1. Processador: Intel Core i3 (ou equivalente)
2. Memória RAM: 8 GB (ou mais)
3. Armazenamento: HD de 465 GB

Software:

1. Sistema Operacional: Windows 10
2. IDE/Editor de Código: Visual Studio Code (VSCode) versão 1.83.0 (ou versão mais recente)
3. Banco de Dados: SQLite
4. Versão do Python: Python 3.12
4. Browsers: Google Chrome (versão mais recente)

5. Bibliotecas e Frameworks: pytest

6. Ferramentas de Teste: pytest

4.2 Ferramentas de Teste

IDE: Visual Studio Code (VSCode)

O Visual Studio Code (VS Code) é um editor de código-fonte desenvolvido pela Microsoft. É leve, gratuito e multiplataforma, funcionando em Windows, macOS e Linux. Destaca-se por ser altamente extensível, permitindo a instalação de uma vasta gama de extensões para adicionar funcionalidades, como suporte a diferentes linguagens de programação, depuração, controle de versões com Git, entre outras

Biblioteca de testes: Pytest

Ele pode ser usado para escrever vários tipos de testes de software, incluindo testes de unidade, testes de integração, testes de ponta a ponta e testes funcionais. Seus recursos incluem testes parametrizados, acessórios e reescrita assertiva.

5 Cronograma

Dia	Atividade	Descrição
01/08 a 09/08	Planejamento	Definição de casos de uso e testes, planejamento das atividades e alocação de recursos.
10/08 a 14/08	Configuração do Ambiente de Testes	Preparação e configuração do ambiente de testes, incluindo software, hardware e ferramentas.
15/08 a 30/08	Desenvolvimento de Scripts de Teste	Criação e revisão de scripts de teste para cobertura completa dos requisitos.
30/08 a 03/09	Execução de testes gerais, Finalização e Entrega da Documentação	Execução dos testes, ajuste de scripts e ambiente, e entrega da documentação final.

6 Referências

Documentação Oficial PyTest - <https://docs.pytest.org/en/stable/>

7 Testes

Cadastro de Usuário:


```

1 import pytest
2 import sqlite3
3 from unittest.mock import patch, MagicMock
4 from app import cadastrar_usuario
5
6
7 def test_cadastrar_usuario_sucesso():
8     nome = "João da Silva"
9     cpf = "12345678901"
10    email = "joao.silva@example.com"
11    telefone = "(11) 91234-5678"
12
13    with patch("app.connect_db") as mock_connect_db, \
14         patch("app.is_valid_cpf", return_value=True), \
15         patch("app.is_valid_email", return_value=True):
16        mock_conn = MagicMock()
17        mock_cursor = MagicMock()
18        mock_connect_db.return_value = mock_conn
19        mock_conn.cursor.return_value = mock_cursor
20        mock_cursor.execute.return_value = None
21
22        result = cadastrar_usuario(nome, cpf, email, telefone)
23
24        assert result == "Usuário cadastrado com sucesso"
25
26
27 def test_cadastrar_usuario_campos_faltando():
28     result = cadastrar_usuario("", "12345678901", "joao.silva@example.com", "(11) 91234-5678")
29
30     assert result == "Todos os campos são obrigatórios"
31
32
33 def test_cadastrar_usuario_cpf_invalido():
34     nome = "João da Silva"
35     cpf = "123"
36     email = "joao.silva@example.com"
37     telefone = "(11) 91234-5678"
38
39     with patch("app.is_valid_cpf", return_value=False):
40         result = cadastrar_usuario(nome, cpf, email, telefone)
41
42     assert result == "CPF inválido"
43
44
45 def test_cadastrar_usuario_email_invalido():
46     nome = "João da Silva"
47     cpf = "12345678901"
48     email = "joao.silvaexample.com"
49     telefone = "(11) 91234-5678"
50
51     with patch("app.is_valid_email", return_value=False):
52         result = cadastrar_usuario(nome, cpf, email, telefone)
53
54     assert result == "E-mail inválido"
55
56
57 def test_cadastrar_usuario_erro_bd():
58     nome = "João da Silva"
59     cpf = "12345678901"
60     email = "joao.silva@example.com"
61     telefone = "(11) 91234-5678"
62
63     with patch("app.connect_db") as mock_connect_db, \
64         patch("app.is_valid_cpf", return_value=True), \
65         patch("app.is_valid_email", return_value=True):
66        mock_conn = MagicMock()
67        mock_cursor = MagicMock()
68        mock_connect_db.return_value = mock_conn
69        mock_conn.cursor.return_value = mock_cursor
70        mock_cursor.execute.side_effect = sqlite3.IntegrityError("CPF ou e-mail já cadastrado")
71

```

Empréstimo de Livro:

```

1 import pytest
2 import sqlite3
3 from unittest.mock import patch, MagicMock
4 from datetime import datetime, timedelta
5 from app import emprestar_livro
6
7
8 def test_emprestar_livro_sucesso():
9     usuario_id = 1
10    livro_id = 1
11    data_emprestimo = datetime.now().date()
12    data_devolucao = data_emprestimo + timedelta(days=14)
13
14    with patch("app.connect_db") as mock_connect_db:
15        mock_conn = MagicMock()
16        mock_cursor = MagicMock()
17        mock_connect_db.return_value = mock_conn
18        mock_conn.cursor.return_value = mock_cursor
19
20        mock_cursor.execute.side_effect = [
21            MagicMock(),
22            MagicMock(),
23            MagicMock()
24        ]
25        mock_cursor.fetchone.return_value = ['Disponível']
26
27        result = emprestar_livro(usuario_id, livro_id)
28
29        assert result["success"] is True
30        assert result["message"] == f"Empréstimo realizado com sucesso. Data de devolução: {data_devolucao}"
31
32
33 def test_emprestar_livro_nao_encontrado():
34     usuario_id = 1
35     livro_id = 999
36
37     with patch("app.connect_db") as mock_connect_db:
38         mock_conn = MagicMock()
39         mock_cursor = MagicMock()
40         mock_connect_db.return_value = mock_conn
41         mock_conn.cursor.return_value = mock_cursor
42         mock_cursor.execute.return_value = None
43         mock_cursor.fetchone.return_value = None
44
45         result = emprestar_livro(usuario_id, livro_id)
46
47         assert result["success"] is False
48         assert result["message"] == "Livro não encontrado"
49
50
51 def test_emprestar_livro_nao_disponivel():
52     usuario_id = 1
53     livro_id = 1
54
55     with patch("app.connect_db") as mock_connect_db:
56         mock_conn = MagicMock()
57         mock_cursor = MagicMock()
58         mock_connect_db.return_value = mock_conn
59         mock_conn.cursor.return_value = mock_cursor
60         mock_cursor.execute.return_value = None
61         mock_cursor.fetchone.return_value = ['Emprestado']
62
63         result = emprestar_livro(usuario_id, livro_id)
64
65         assert result["success"] is False
66         assert result["message"] == "O livro não está disponível"
67
68
69 def test_emprestar_livro_sem_id():
70     result = emprestar_livro(None, None)
71
72     assert result["success"] is False
73     assert result["message"] == "ID do usuário e do livro são obrigatórios"
74
75
76 def test_emprestar_livro_erro_bd():
77     usuario_id = 1
78     livro_id = 1
79
80     with patch("app.connect_db") as mock_connect_db:

```

Devolução de Livro:

```

1 import pytest
2 import sqlite3
3 from unittest.mock import patch, MagicMock
4 from app import devolver_livro
5
6
7 def test_devolver_livro_sucesso():
8     usuario_id = 1
9     livro_id = 1
10
11     with patch("app.connect_db") as mock_connect_db:
12         mock_conn = MagicMock()
13         mock_cursor = MagicMock()
14         mock_connect_db.return_value = mock_conn
15         mock_conn.cursor.return_value = mock_cursor
16
17         mock_cursor.execute.side_effect = [
18             MagicMock(),
19             MagicMock(),
20             MagicMock(),
21             MagicMock()
22         ]
23         mock_cursor.fetchone.side_effect = [['Emprestado'], [1]]
24
25         result = devolver_livro(usuario_id, livro_id)
26
27         assert result["success"] is True
28         assert result["message"] == "Devolução registrada com sucesso"
29
30
31 def test_devolver_livro_nao_encontrado():
32     usuario_id = 1
33     livro_id = 1
34
35     with patch("app.connect_db") as mock_connect_db:
36         mock_conn = MagicMock()
37         mock_cursor = MagicMock()
38         mock_connect_db.return_value = mock_conn
39         mock_conn.cursor.return_value = mock_cursor
40
41         mock_cursor.execute.side_effect = [MagicMock()]
42         mock_cursor.fetchone.return_value = None
43
44         result = devolver_livro(usuario_id, livro_id)
45
46         assert result["success"] is False
47         assert result["message"] == "Livro não encontrado"
48
49
50 def test_devolver_livro_nao_emprestado():
51     usuario_id = 1
52     livro_id = 1
53
54     with patch("app.connect_db") as mock_connect_db:
55         mock_conn = MagicMock()
56         mock_cursor = MagicMock()
57         mock_connect_db.return_value = mock_conn
58         mock_conn.cursor.return_value = mock_cursor
59
60         mock_cursor.execute.side_effect = [MagicMock()]
61         mock_cursor.fetchone.side_effect = None
62
63         result = devolver_livro(usuario_id, livro_id)
64
65         assert result["success"] is False
66         assert result["message"] == "O livro não está marcado como emprestado"
67
68
69
70 def test_devolver_livro_nao_emprestado_para_este_usuario():
71     usuario_id = 1
72     livro_id = 1
73
74     with patch("app.connect_db") as mock_connect_db:
75         mock_conn = MagicMock()

```

Consulta de Disponibilidade de Livro:

```
1 import pytest
2 import sqlite3
3 from unittest.mock import patch, MagicMock
4 from app import consultar_disponibilidade
5
6
7 def test_consultar_disponibilidade_sucesso():
8     livro_id = 1
9
10    with patch("app.connect_db") as mock_connect_db:
11        mock_conn = MagicMock()
12        mock_cursor = MagicMock()
13        mock_connect_db.return_value = mock_conn
14        mock_conn.cursor.return_value = mock_cursor
15
16        mock_cursor.execute.side_effect = [MagicMock()]
17        mock_cursor.fetchone.return_value = ['Disponível']
18
19        result = consultar_disponibilidade(livro_id)
20
21        assert result["success"] is True
22        assert result["status"] == 'Disponível'
23
24
25 def test_consultar_disponibilidade_livro_nao_encontrado():
26     livro_id = 1
27
28     with patch("app.connect_db") as mock_connect_db:
29         mock_conn = MagicMock()
30         mock_cursor = MagicMock()
31         mock_connect_db.return_value = mock_conn
32         mock_conn.cursor.return_value = mock_cursor
33
34         mock_cursor.execute.side_effect = [MagicMock()]
35         mock_cursor.fetchone.return_value = None
36
37         result = consultar_disponibilidade(livro_id)
38
39         assert result["success"] is False
40         assert result["message"] == "Livro não encontrado"
41
42
43 def test_consultar_disponibilidade_sem_id():
44     result = consultar_disponibilidade(None)
45
46     assert result["success"] is False
47     assert result["message"] == "ID do livro é obrigatório"
48
49
50 def test_consultar_disponibilidade_erro_bd():
51     livro_id = 1
52
53     with patch("app.connect_db") as mock_connect_db:
54         mock_conn = MagicMock()
55         mock_cursor = MagicMock()
56         mock_connect_db.return_value = mock_conn
57         mock_conn.cursor.return_value = mock_cursor
58         mock_cursor.execute.side_effect = sqlite3.Error("Erro de banco de dados")
59
60         result = consultar_disponibilidade(livro_id)
61
62         assert result["success"] is False
```

Atualização de Informações do Usuário:


```

1  import pytest
2  import sqlite3
3  from unittest.mock import patch, MagicMock
4  from app import atualizar_usuario
5
6
7  def test_atualizar_usuario_sucesso():
8      user_id = 1
9      nome = "Carlos Silva"
10     email = "carlos.silva@example.com"
11     telefone = "(11) 91234-5678"
12
13     with patch("app.connect_db") as mock_connect_db:
14         mock_conn = MagicMock()
15         mock_cursor = MagicMock()
16         mock_connect_db.return_value = mock_conn
17         mock_conn.cursor.return_value = mock_cursor
18         mock_cursor.execute.return_value = None
19
20         result = atualizar_usuario(user_id, nome, email, telefone)
21
22         assert result["success"] is True
23         assert result["message"] == "Usuário atualizado com sucesso"
24
25
26  def test_atualizar_usuario_sem_informacao():
27      user_id = 1
28
29      result = atualizar_usuario(user_id)
30
31      assert result["success"] is False
32      assert result["message"] == "Nenhuma informação para atualizar"
33
34
35  def test_atualizar_usuario_email_invalido():
36      user_id = 1
37      email = "carlos.silvaexample.com"
38
39      result = atualizar_usuario(user_id, email=email)
40
41      assert result["success"] is False
42      assert result["message"] == "E-mail inválido"
43
44
45  def test_atualizar_usuario_erro_bd():
46      user_id = 1
47      nome = "Carlos Silva"
48
49      with patch("app.connect_db") as mock_connect_db:
50         mock_conn = MagicMock()
51         mock_cursor = MagicMock()
52         mock_connect_db.return_value = mock_conn
53         mock_conn.cursor.return_value = mock_cursor
54         mock_cursor.execute.side_effect = sqlite3.Error("Erro de banco de dados")
55
56         result = atualizar_usuario(user_id, nome=nome)
57
58         assert result["success"] is False
59         assert result["message"] == "Erro de banco de dados"
60
61
62

```

Cadastro de Novo Livro:

```

1  import pytest
2  import sqlite3
3  from unittest.mock import patch, MagicMock
4  from app import cadastrar_livro, is_valid_isbn
5
6
7  def test_cadastrar_livro_sucesso():
8      titulo = "O Senhor dos Anéis"
9      autor = "J.R.R. Tolkien"
10     isbn = "978-3-16-148410-0"
11     categoria = "Fantasia"
12
13     with patch("app.connect_db") as mock_connect_db, \
14         patch("app.is_valid_isbn", return_value=True):
15         mock_conn = MagicMock()
16         mock_cursor = MagicMock()
17         mock_connect_db.return_value = mock_conn
18         mock_conn.cursor.return_value = mock_cursor
19         mock_cursor.execute.return_value = None
20
21         result = cadastrar_livro(titulo, autor, isbn, categoria)
22
23         assert result["success"] is True
24         assert result["message"] == "Livro cadastrado com sucesso"
25
26
27  def test_cadastrar_livro_campos_faltando():
28     result = cadastrar_livro("", "J.K. Rowling", "978-3-16-148410-0", "Fantasia")
29
30     assert result["success"] is False
31     assert result["message"] == "Todos os campos são obrigatórios"
32
33
34  def test_cadastrar_livro_isbn_invalido():
35     titulo = "Harry Potter e a Pedra Filosofal"
36     autor = "J.K. Rowling"
37     isbn = "978-3-16-148410-XYZ"
38     categoria = "Fantasia"
39
40     with patch("app.is_valid_isbn", return_value=False):
41         result = cadastrar_livro(titulo, autor, isbn, categoria)
42
43         assert result["success"] is False
44         assert result["message"] == "ISBN inválido"
45
46
47  def test_cadastrar_livro_isbn_ja_cadastrado():
48     titulo = "O Hobbit"
49     autor = "J.R.R. Tolkien"
50     isbn = "978-0-345-33968-3"
51     categoria = "Fantasia"
52
53     with patch("app.connect_db") as mock_connect_db, \
54         patch("app.is_valid_isbn", return_value=True):
55         mock_conn = MagicMock()
56         mock_cursor = MagicMock()
57         mock_connect_db.return_value = mock_conn
58         mock_conn.cursor.return_value = mock_cursor
59         mock_cursor.execute.side_effect = sqlite3.IntegrityError("ISBN já cadastrado")
60
61         result = cadastrar_livro(titulo, autor, isbn, categoria)
62
63         assert result["success"] is False
64         assert result["message"] == "ISBN já cadastrado"
65

```

Remoção de Livro:

```

1  import pytest
2  import sqlite3
3  from unittest.mock import patch, MagicMock
4  from app import remover_livro
5
6
7  def test_remover_livro_sucesso():
8      livro_id = 1
9
10     with patch("app.connect_db") as mock_connect_db:
11         mock_conn = MagicMock()
12         mock_cursor = MagicMock()
13         mock_connect_db.return_value = mock_conn
14         mock_conn.cursor.return_value = mock_cursor
15         mock_cursor.execute.return_value = None
16         mock_cursor.rowcount = 1
17
18         result = remover_livro(livro_id)
19
20         assert result["success"] is True
21         assert result["message"] == "Livro removido com sucesso"
22
23
24  def test_remover_livro_nao_encontrado():
25      livro_id = 999
26
27      with patch("app.connect_db") as mock_connect_db:
28          mock_conn = MagicMock()
29          mock_cursor = MagicMock()
30          mock_connect_db.return_value = mock_conn
31          mock_conn.cursor.return_value = mock_cursor
32          mock_cursor.execute.return_value = None
33          mock_cursor.rowcount = 0
34
35          result = remover_livro(livro_id)
36
37          assert result["success"] is False
38          assert result["message"] == "Livro não encontrado"
39
40
41  def test_remover_livro_sem_id():
42      result = remover_livro(None)
43
44      assert result["success"] is False
45      assert result["message"] == "ID do livro é obrigatório"
46
47
48  def test_remover_livro_erro_bd():
49      livro_id = 1
50
51      with patch("app.connect_db") as mock_connect_db:
52          mock_conn = MagicMock()
53          mock_cursor = MagicMock()
54          mock_connect_db.return_value = mock_conn
55          mock_conn.cursor.return_value = mock_cursor
56          mock_cursor.execute.side_effect = sqlite3.Error("Erro de banco de dados")
57
58          result = remover_livro(livro_id)
59
60          assert result["success"] is False
61          assert result["message"].startswith("Erro de banco de dados")
62

```

Renovação de Empréstimo:

```

1 import pytest
2 import sqlite3
3 from unittest.mock import patch, MagicMock
4 from app import renovar_emprestimo
5
6 def test_renovar_emprestimo_sucesso():
7     usuario_id = 1
8     livro_id = 1
9
10    with patch("app.connect_db") as mock_connect_db:
11        mock_conn = MagicMock()
12        mock_cursor = MagicMock()
13        mock_connect_db.return_value = mock_conn
14        mock_conn.cursor.return_value = mock_cursor
15
16        mock_cursor.execute.side_effect = [
17            MagicMock(),
18            MagicMock(),
19            MagicMock()
20        ]
21        mock_cursor.fetchone.side_effect = [['Emprestado'], [1, '2024-08-30']]
22
23        result = renovar_emprestimo(usuario_id, livro_id)
24
25        assert result["success"] is True
26        assert result["message"] == "Empréstimo renovado com sucesso"
27
28 def test_renovar_emprestimo_nao_encontrado():
29     usuario_id = 1
30     livro_id = 1
31
32     with patch("app.connect_db") as mock_connect_db:
33         mock_conn = MagicMock()
34         mock_cursor = MagicMock()
35         mock_connect_db.return_value = mock_conn
36         mock_conn.cursor.return_value = mock_cursor
37
38         mock_cursor.execute.side_effect = [MagicMock()]
39         mock_cursor.fetchone.return_value = None
40
41         result = renovar_emprestimo(usuario_id, livro_id)
42
43         assert result["success"] is False
44         assert result["message"] == 'Livro não encontrado'
45
46 def test_renovar_emprestimo_livro_disponivel():
47     usuario_id = 1
48     livro_id = 1
49
50     with patch("app.connect_db") as mock_connect_db:
51         mock_conn = MagicMock()
52         mock_cursor = MagicMock()
53         mock_connect_db.return_value = mock_conn
54         mock_conn.cursor.return_value = mock_cursor
55
56         mock_cursor.execute.side_effect = [MagicMock()]
57         mock_cursor.fetchone.side_effect = ['Disponível']
58
59         result = renovar_emprestimo(usuario_id, livro_id)
60
61         assert result["success"] is False
62         assert result["message"] == "O livro não está marcado como emprestado"
63
64
65 def test_renovar_emprestimo_nao_emprestado():
66     usuario_id = 1
67     livro_id = 1
68
69     with patch("app.connect_db") as mock_connect_db:
70         mock_conn = MagicMock()
71         mock_cursor = MagicMock()
72         mock_connect_db.return_value = mock_conn
73         mock_conn.cursor.return_value = mock_cursor
74
75         mock_cursor.execute.side_effect = [

```

Consulta de Histórico de Empréstimo:


```

1  import pytest
2  import sqlite3
3  from unittest.mock import patch, MagicMock
4  from app import consultar_historico
5
6
7  def test_consultar_historico_sucesso():
8      usuario_id = 1
9
10     with patch("app.connect_db") as mock_connect_db:
11         mock_conn = MagicMock()
12         mock_cursor = MagicMock()
13         mock_connect_db.return_value = mock_conn
14         mock_conn.cursor.return_value = mock_cursor
15
16         mock_cursor.execute.side_effect = [MagicMock()]
17         mock_cursor.fetchall.return_value = [
18             ('Livro A', '2024-08-01', '2024-08-15'),
19             ('Livro B', '2024-08-10', '2024-08-24')
20         ]
21
22         result = consultar_historico(usuario_id)
23
24         assert result["success"] is True
25         assert len(result["historico"]) == 2
26         assert result["historico"][0] == ('Livro A', '2024-08-01', '2024-08-15')
27
28
29  def test_consultar_historico_vazio():
30      usuario_id = 1
31
32      with patch("app.connect_db") as mock_connect_db:
33          mock_conn = MagicMock()
34          mock_cursor = MagicMock()
35          mock_connect_db.return_value = mock_conn
36          mock_conn.cursor.return_value = mock_cursor
37
38          mock_cursor.execute.side_effect = [MagicMock()]
39          mock_cursor.fetchall.return_value = []
40
41          result = consultar_historico(usuario_id)
42
43          assert result["success"] is False
44          assert result["message"] == "Nenhum histórico encontrado"
45
46
47  def test_consultar_historico_sem_id():
48      result = consultar_historico(None)
49
50      assert result["success"] is False
51      assert result["message"] == "ID do usuário é obrigatório"
52
53
54  def test_consultar_historico_erro_bd():
55      usuario_id = 1
56
57      with patch("app.connect_db") as mock_connect_db:
58          mock_conn = MagicMock()
59          mock_cursor = MagicMock()
60          mock_connect_db.return_value = mock_conn
61          mock_conn.cursor.return_value = mock_cursor
62          mock_cursor.execute.side_effect = sqlite3.Error("Erro de banco de dados")

```

Geração de Relatórios de Livros:

```

1  import pytest
2  import sqlite3
3  from unittest.mock import patch, MagicMock
4  from app import gerar_relatorio
5
6
7  def test_gerar_relatorio_emprestados():
8      with patch("app.connect_db") as mock_connect_db:
9          mock_conn = MagicMock()
10         mock_cursor = MagicMock()
11         mock_connect_db.return_value = mock_conn
12         mock_conn.cursor.return_value = mock_cursor
13
14         mock_cursor.execute.return_value = None
15         mock_cursor.fetchall.return_value = [
16             ('Livro 1', '2024-09-14'),
17             ('Livro 2', '2024-09-15')
18         ]
19
20         result = gerar_relatorio('emprestados')
21
22         assert result == {
23             "success": True,
24             "data": [
25                 ('Livro 1', '2024-09-14'),
26                 ('Livro 2', '2024-09-15')
27             ]
28         }
29
30
31  def test_gerar_relatorio_disponiveis():
32      with patch("app.connect_db") as mock_connect_db:
33          mock_conn = MagicMock()
34          mock_cursor = MagicMock()
35          mock_connect_db.return_value = mock_conn
36          mock_conn.cursor.return_value = mock_cursor
37
38          mock_cursor.execute.return_value = None
39          mock_cursor.fetchall.return_value = [
40              ('Livro 1',),
41              ('Livro 3',)
42          ]
43
44          result = gerar_relatorio('disponiveis')
45
46          assert result == {
47              "success": True,
48              "data": [
49                  ('Livro 1',),
50                  ('Livro 3',)
51              ]
52          }
53
54  def test_gerar_relatorio_atraso():
55      with patch("app.connect_db") as mock_connect_db:
56          mock_conn = MagicMock()
57          mock_cursor = MagicMock()
58          mock_connect_db.return_value = mock_conn
59          mock_conn.cursor.return_value = mock_cursor
60
61          mock_cursor.execute.return_value = None
62          mock_cursor.fetchall.return_value = [
63              ('Livro 1', '2024-08-28'),
64              ('Livro 2', '2024-08-27')
65          ]
66
67          result = gerar_relatorio('atraso')

```

Resultado dos testes:

```
===== test session starts =====
platform win32 -- Python 3.12.5, pytest-8.3.2, pluggy-1.5.0
rootdir: C:\Users\eduar\Documents\python\Biblioteca

tests\test_atualizar_usuario.py .... [ 7%]
tests\test_bd.py .. [ 11%]
tests\test_cadastrar_livro.py ..... [ 23%]
tests\test_cadastrar_usuario.py ..... [ 32%]
tests\test_consultar_disponibilidade.py .... [ 40%]
tests\test_consultar_historico.py .... [ 48%]
tests\test_devolver_livro.py ..... [ 59%]
tests\test_emprestar_livro.py ..... [ 69%]
tests\test gerar_relatorio.py ..... [ 80%]
tests\test_remover_livro.py .... [ 88%]
tests\test_renovar_emprestimo.py ..... [100%]

===== 52 passed in 0.82s =====
```

Métrica de qualidade utilizada foi o Coverage e esses foram nossos resultados:

Coverage report: 100%

Files Functions Classes

coverage.py v7.6.1, created at 2024-08-31 22:01 -0300

File ▲	statements	missing	excluded	coverage
__init__.py	0	0	0	100%
app.py	203	0	0	100%
tests__init__.py	0	0	0	100%
tests\conftest.py	3	0	0	100%
tests\test_atualizar_usuario.py	41	0	0	100%
tests\test_bd.py	18	0	0	100%
tests\test_cadastrar_livro.py	51	0	0	100%
tests\test_cadastrar_usuario.py	49	0	0	100%
tests\test_consultar_disponibilidade.py	43	0	0	100%
tests\test_consultar_historico.py	44	0	0	100%
tests\test_devolver_livro.py	72	0	0	100%
tests\test_emprestar_livro.py	62	0	0	100%
tests\test gerar_relatorio.py	58	0	0	100%
tests\test_remover_livro.py	43	0	0	100%
tests\test_renovar_emprestimo.py	72	0	0	100%
Total	759	0	0	100%

coverage.py v7.6.1, created at 2024-08-31 22:01 -0300