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

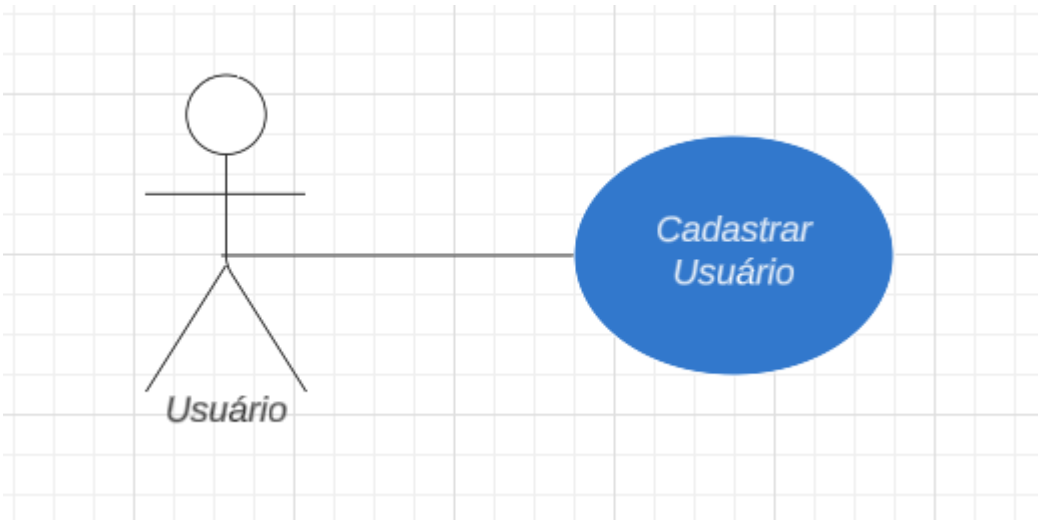
1 Introdução

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):

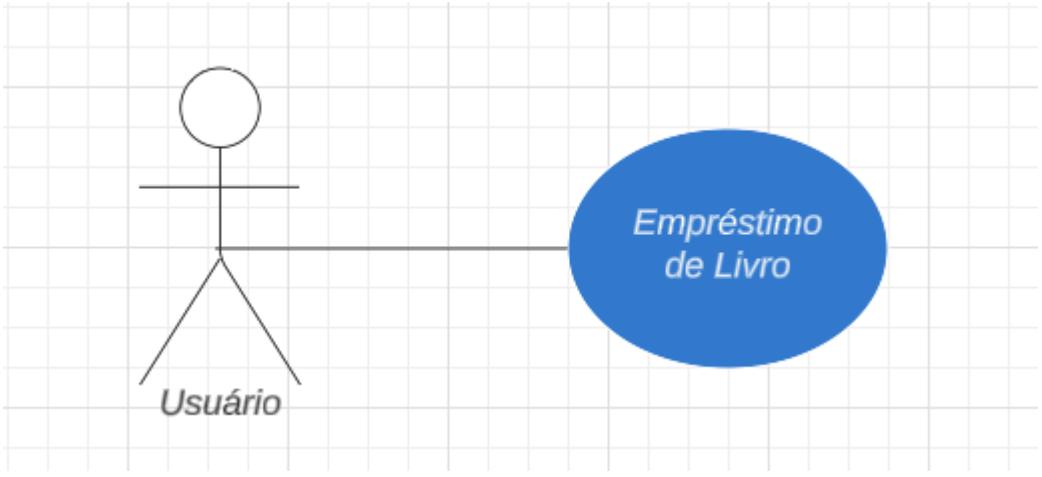
1. O sistema detecta que o CPF ou o e-mail fornecido já está cadastrado.
2. O sistema exibe uma mensagem de erro informando que o CPF ou e-mail já está cadastrado.

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:

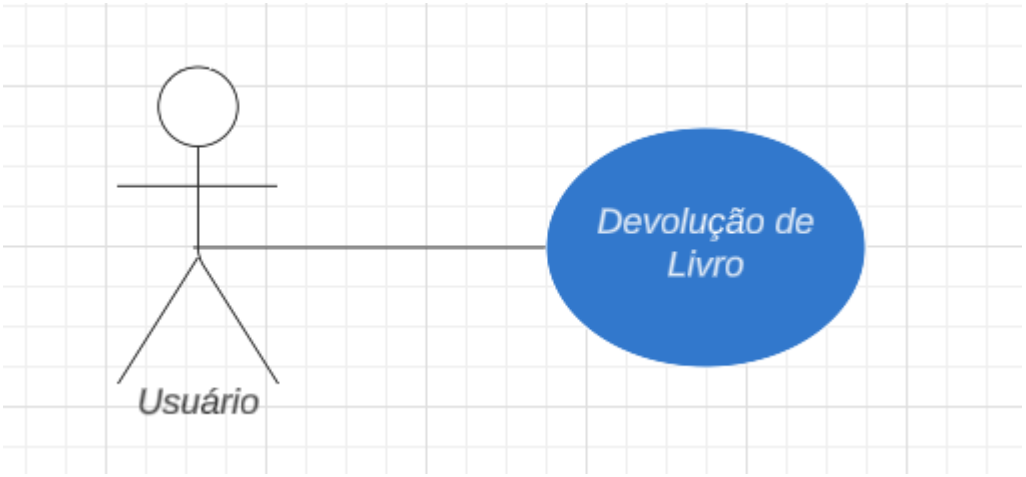
1. Ocorreu um erro ao acessar o banco de dados durante o processo de empréstimo.
2. O sistema exibe uma mensagem de erro informando sobre o problema com o banco de dados.

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.

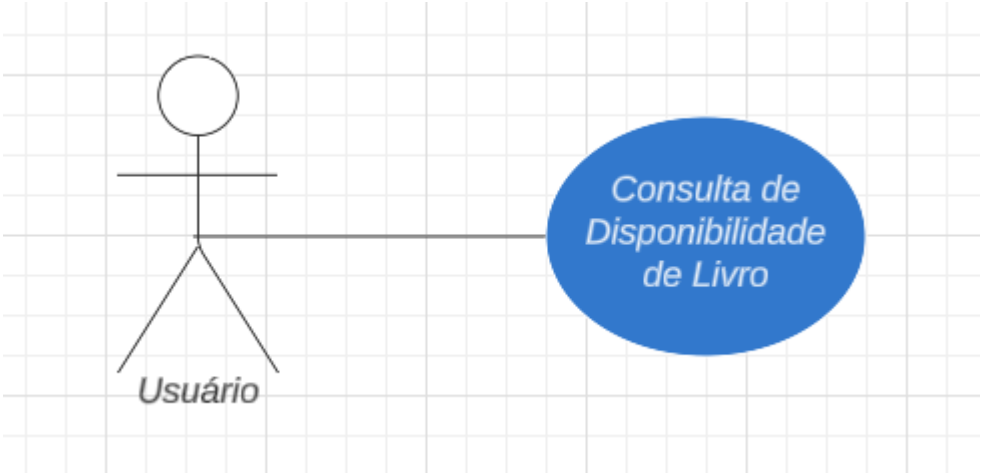
Plano de Testes

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.

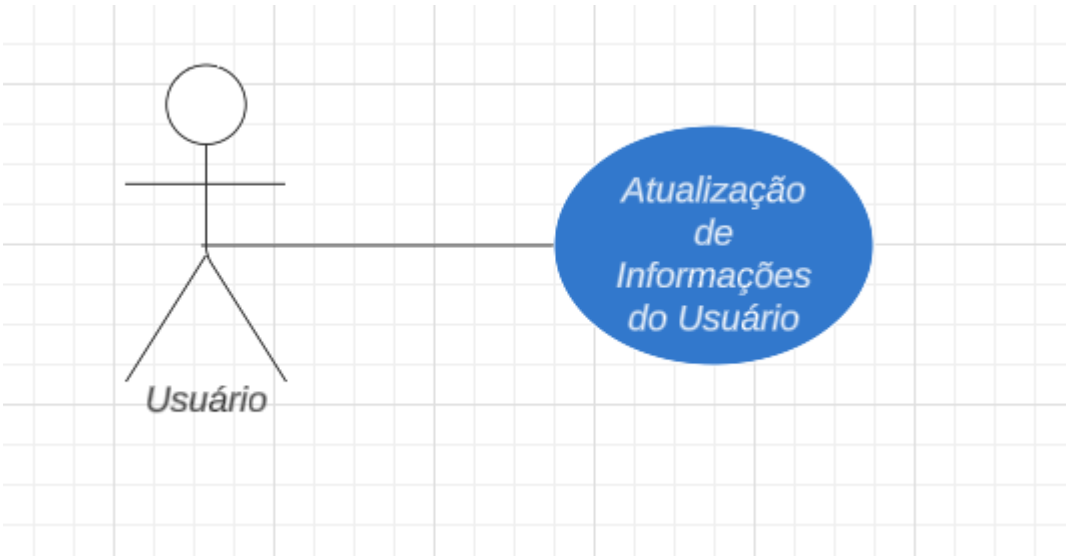
Erro de Banco de Dados:

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.

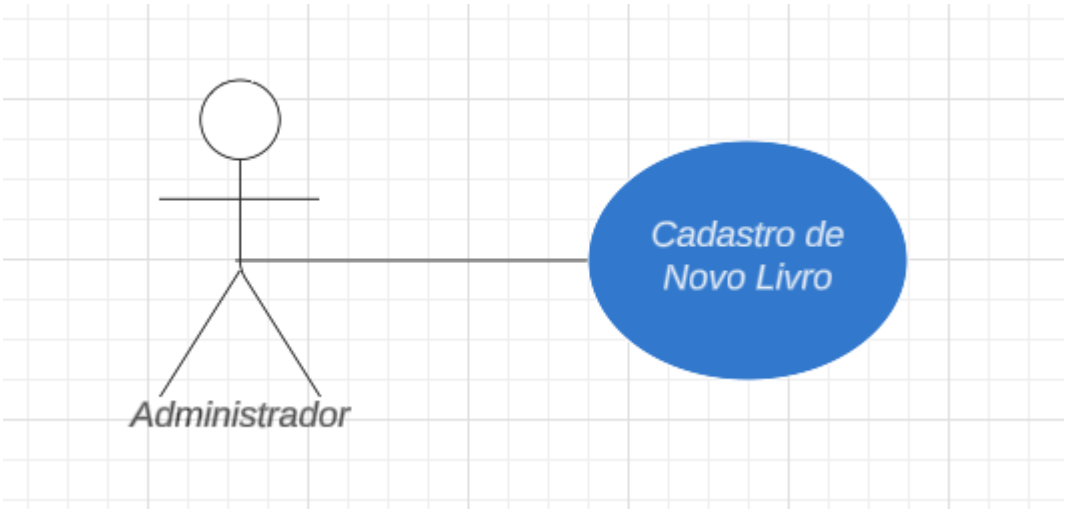
Erro de Banco de Dados:

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:

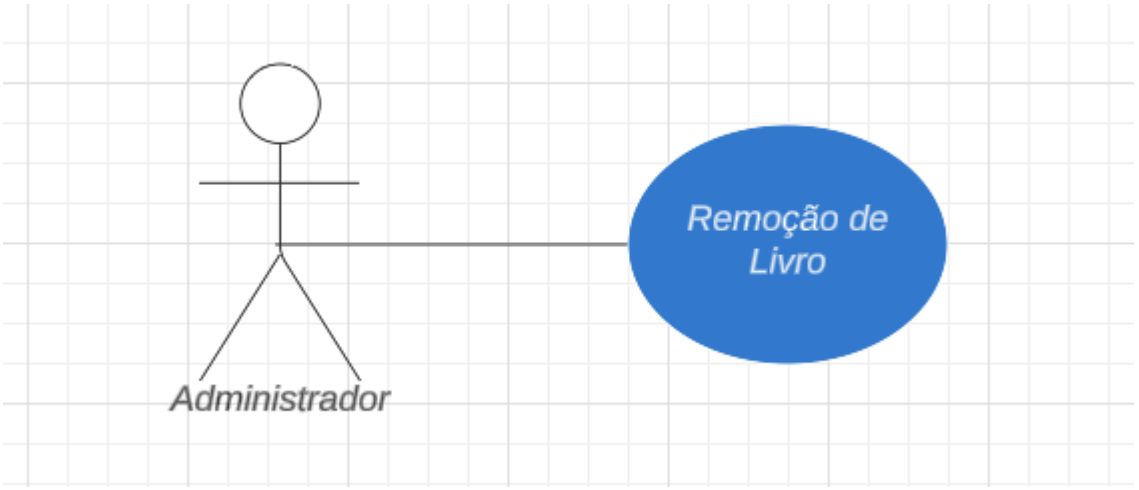
1. Ocorreu um erro no banco de dados durante a tentativa de cadastro do livro.
2. O sistema exibe uma mensagem de erro indicando que houve um problema ao cadastrar o livro.

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.

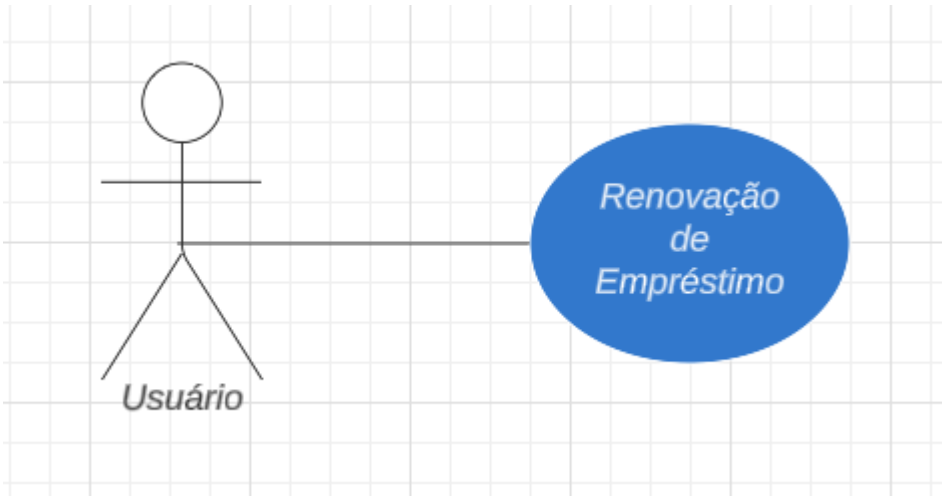
Erro de Banco de Dados:

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:

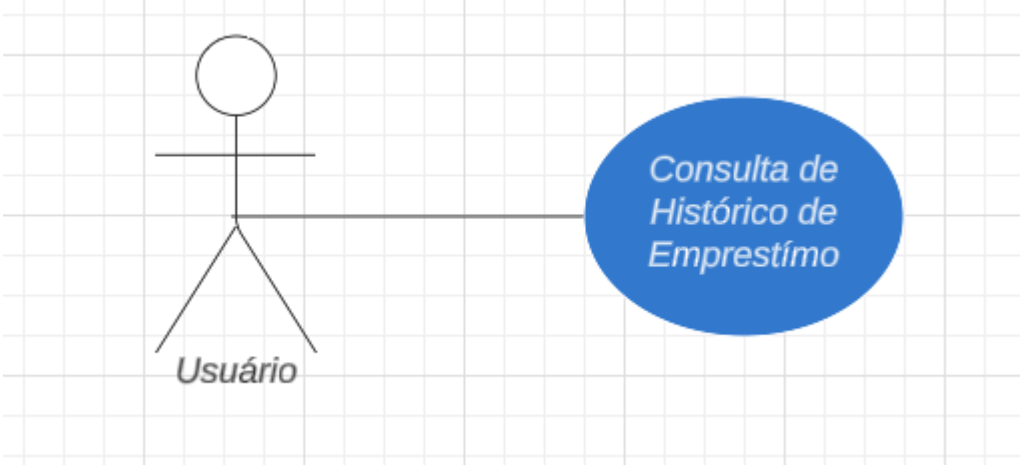
1. Ocorreu um erro no banco de dados durante a tentativa de renovar o empréstimo.
2. O sistema exibe uma mensagem de erro indicando que houve um problema ao tentar renovar o empréstimo.

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 Emprestado:**
  - **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:

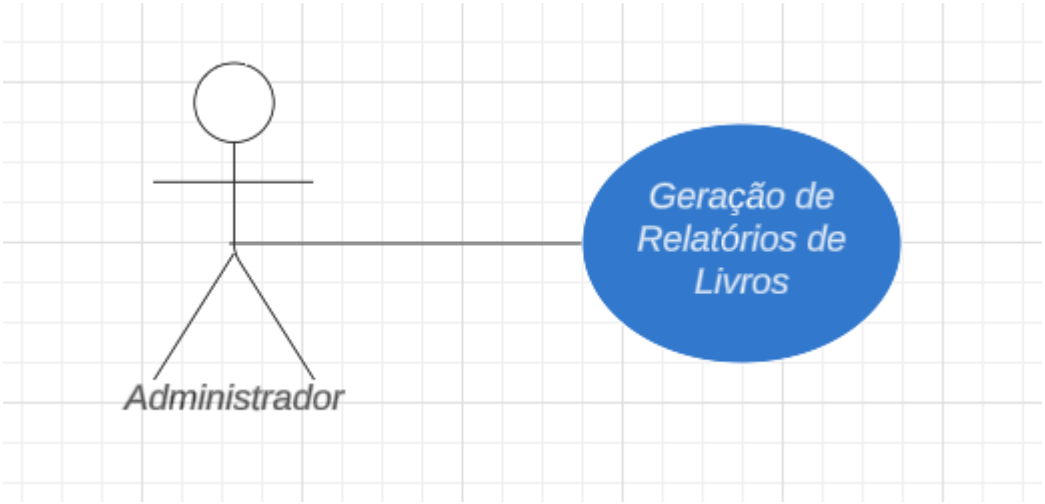
1. Ocorreu um erro no banco de dados durante a tentativa de consultar o histórico.

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:	☒ Manual ☐ Automática
Estágio do teste: ☐ Integração ☐ Sistema ☒ Unidade ☐ Aceitação	Abordagem do teste ☒ Caixa branca ☐ 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:	☒ Manual ☐ Automática
Estágio do teste: ☐ Integração ☐ Sistema ☒ Unidade ☐ Aceitação	Abordagem do teste ☒ Caixa branca ☐ 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:

```python
import pytest
import sqlite3
from unittest.mock import patch, MagicMock
from app import cadastrar_usuario


def test_cadastrar_usuario_sucesso():
    nome = "João da Silva"
    cpf = "12345678901"
    email = "joao.silva@example.com"
    telefone = "(11) 91234-5678"

    with patch("app.connect_db") as mock_connect_db, \
         patch("app.is_valid_cpf", return_value=True), \
         patch("app.is_valid_email", return_value=True):
        mock_conn = MagicMock()
        mock_cursor = MagicMock()
        mock_connect_db.return_value = mock_conn
        mock_conn.cursor.return_value = mock_cursor
        mock_cursor.execute.return_value = None

        result = cadastrar_usuario(nome, cpf, email, telefone)

        assert result == "Usuário cadastrado com sucesso"


def test_cadastrar_usuario_campos_faltando():
    result = cadastrar_usuario("", "12345678901", "joao.silva@example.com", "(11) 91234-5678")

    assert result == "Todos os campos são obrigatórios"


def test_cadastrar_usuario_cpf_invalido():
    nome = "João da Silva"
    cpf = "123"
    email = "joao.silva@example.com"
    telefone = "(11) 91234-5678"

    with patch("app.is_valid_cpf", return_value=False):
        result = cadastrar_usuario(nome, cpf, email, telefone)

    assert result == "CPF inválido"


def test_cadastrar_usuario_email_invalido():
    nome = "João da Silva"
    cpf = "12345678901"
    email = "joao.silvaexample.com"
    telefone = "(11) 91234-5678"

    with patch("app.is_valid_email", return_value=False):
        result = cadastrar_usuario(nome, cpf, email, telefone)

    assert result == "E-mail inválido"


def test_cadastrar_usuario_erro_bd():
    nome = "João da Silva"
    cpf = "12345678901"
    email = "joao.silva@example.com"
    telefone = "(11) 91234-5678"

    with patch("app.connect_db") as mock_connect_db, \
         patch("app.is_valid_cpf", return_value=True), \
         patch("app.is_valid_email", return_value=True):
        mock_conn = MagicMock()
        mock_cursor = MagicMock()
        mock_connect_db.return_value = mock_conn
        mock_conn.cursor.return_value = mock_cursor
        mock_cursor.execute.side_effect = sqlite3.IntegrityError("CPF ou e-mail já cadastrado")

```

Empréstimo de Livro:

```python
import pytest
import sqlite3
from unittest.mock import patch, MagicMock
from datetime import datetime, timedelta
from app import emprestar_livro


def test_emprestar_livro_sucesso():
    usuario_id = 1
    livro_id = 1
    data_emprestimo = datetime.now().date()
    data_devolucao = data_emprestimo + timedelta(days=14)

    with patch("app.connect_db") as mock_connect_db:
        mock_conn = MagicMock()
        mock_cursor = MagicMock()
        mock_connect_db.return_value = mock_conn
        mock_conn.cursor.return_value = mock_cursor

        mock_cursor.execute.side_effect = [
            MagicMock(),
            MagicMock(),
            MagicMock()
        ]
        mock_cursor.fetchone.return_value = ['Disponível']

        result = emprestar_livro(usuario_id, livro_id)

        assert result["success"] is True
        assert result["message"] == f"Empréstimo realizado com sucesso. Data de devolução: {data_devolucao}"


def test_emprestar_livro_nao_encontrado():
    usuario_id = 1
    livro_id = 999

    with patch("app.connect_db") as mock_connect_db:
        mock_conn = MagicMock()
        mock_cursor = MagicMock()
        mock_connect_db.return_value = mock_conn
        mock_conn.cursor.return_value = mock_cursor
        mock_cursor.execute.return_value = None
        mock_cursor.fetchone.return_value = None

        result = emprestar_livro(usuario_id, livro_id)

        assert result["success"] is False
        assert result["message"] == "Livro não encontrado"


def test_emprestar_livro_nao_disponivel():
    usuario_id = 1
    livro_id = 1

    with patch("app.connect_db") as mock_connect_db:
        mock_conn = MagicMock()
        mock_cursor = MagicMock()
        mock_connect_db.return_value = mock_conn
        mock_conn.cursor.return_value = mock_cursor
        mock_cursor.execute.return_value = None
        mock_cursor.fetchone.return_value = ['Emprestado']

        result = emprestar_livro(usuario_id, livro_id)

        assert result["success"] is False
        assert result["message"] == "O livro não está disponível"


def test_emprestar_livro_sem_id():
    result = emprestar_livro(None, None)

    assert result["success"] is False
    assert result["message"] == "ID do usuário e do livro são obrigatórios"


def test_emprestar_livro_erro_bd():
    usuario_id = 1
    livro_id = 1

    with patch("app.connect_db") as mock_connect_db:
```

Devolução de Livro:

```python
import pytest
import sqlite3
from unittest.mock import patch, MagicMock
from app import devolver_livro


def test_devolver_livro_sucesso():
    usuario_id = 1
    livro_id = 1

    with patch("app.connect_db") as mock_connect_db:
        mock_conn = MagicMock()
        mock_cursor = MagicMock()
        mock_connect_db.return_value = mock_conn
        mock_conn.cursor.return_value = mock_cursor

        mock_cursor.execute.side_effect = [
            MagicMock(),
            MagicMock(),
            MagicMock(),
            MagicMock()
        ]
        mock_cursor.fetchone.side_effect = [['Emprestado'], [1]]

        result = devolver_livro(usuario_id, livro_id)

        assert result["success"] is True
        assert result["message"] == "Devolução registrada com sucesso"


def test_devolver_livro_nao_encontrado():
    usuario_id = 1
    livro_id = 1

    with patch("app.connect_db") as mock_connect_db:
        mock_conn = MagicMock()
        mock_cursor = MagicMock()
        mock_connect_db.return_value = mock_conn
        mock_conn.cursor.return_value = mock_cursor

        mock_cursor.execute.side_effect = [MagicMock()]
        mock_cursor.fetchone.return_value = None

        result = devolver_livro(usuario_id, livro_id)

        assert result["success"] is False
        assert result["message"] == "Livro não encontrado"


def test_devolver_livro_nao_emprestado():
    usuario_id = 1
    livro_id = 1

    with patch("app.connect_db") as mock_connect_db:
        mock_conn = MagicMock()
        mock_cursor = MagicMock()
        mock_connect_db.return_value = mock_conn
        mock_conn.cursor.return_value = mock_cursor

        mock_cursor.execute.side_effect = [MagicMock()]
        mock_cursor.fetchone.side_effect = None

        result = devolver_livro(usuario_id, livro_id)

        assert result["success"] is False
        assert result["message"] == "O livro não está marcado como emprestado"



def test_devolver_livro_nao_emprestado_para_este_usuario():
    usuario_id = 1
    livro_id = 1

    with patch("app.connect_db") as mock_connect_db:
```

Consulta de Disponibilidade de Livro:

```python
import pytest
import sqlite3
from unittest.mock import patch, MagicMock
from app import consultar_disponibilidade


def test_consultar_disponibilidade_sucesso():
    livro_id = 1

    with patch("app.connect_db") as mock_connect_db:
        mock_conn = MagicMock()
        mock_cursor = MagicMock()
        mock_connect_db.return_value = mock_conn
        mock_conn.cursor.return_value = mock_cursor

        mock_cursor.execute.side_effect = [MagicMock()]
        mock_cursor.fetchone.return_value = ['Disponível']

        result = consultar_disponibilidade(livro_id)

        assert result["success"] is True
        assert result["status"] == 'Disponível'


def test_consultar_disponibilidade_livro_nao_encontrado():
    livro_id = 1

    with patch("app.connect_db") as mock_connect_db:
        mock_conn = MagicMock()
        mock_cursor = MagicMock()
        mock_connect_db.return_value = mock_conn
        mock_conn.cursor.return_value = mock_cursor

        mock_cursor.execute.side_effect = [MagicMock()]
        mock_cursor.fetchone.return_value = None

        result = consultar_disponibilidade(livro_id)

        assert result["success"] is False
        assert result["message"] == "Livro não encontrado"


def test_consultar_disponibilidade_sem_id():
    result = consultar_disponibilidade(None)

    assert result["success"] is False
    assert result["message"] == "ID do livro é obrigatório"


def test_consultar_disponibilidade_erro_bd():
    livro_id = 1

    with patch("app.connect_db") as mock_connect_db:
        mock_conn = MagicMock()
        mock_cursor = MagicMock()
        mock_connect_db.return_value = mock_conn
        mock_conn.cursor.return_value = mock_cursor
        mock_cursor.execute.side_effect = sqlite3.Error("Erro de banco de dados")

        result = consultar_disponibilidade(livro_id)

        assert result["success"] is False
```

Atualização de Informações do Usuário:

```python
import pytest
import sqlite3
from unittest.mock import patch, MagicMock
from app import atualizar_usuario


def test_atualizar_usuario_sucesso():
    user_id = 1
    nome = "Carlos Silva"
    email = "carlos.silva@example.com"
    telefone = "(11) 91234-5678"

    with patch("app.connect_db") as mock_connect_db:
        mock_conn = MagicMock()
        mock_cursor = MagicMock()
        mock_connect_db.return_value = mock_conn
        mock_conn.cursor.return_value = mock_cursor
        mock_cursor.execute.return_value = None

        result = atualizar_usuario(user_id, nome, email, telefone)

        assert result["success"] is True
        assert result["message"] == "Usuário atualizado com sucesso"


def test_atualizar_usuario_sem_informacao():
    user_id = 1

    result = atualizar_usuario(user_id)

    assert result["success"] is False
    assert result["message"] == "Nenhuma informação para atualizar"


def test_atualizar_usuario_email_invalido():
    user_id = 1
    email = "carlos.silvaexample.com"

    result = atualizar_usuario(user_id, email=email)

    assert result["success"] is False
    assert result["message"] == "E-mail inválido"


def test_atualizar_usuario_erro_bd():
    user_id = 1
    nome = "Carlos Silva"

    with patch("app.connect_db") as mock_connect_db:
        mock_conn = MagicMock()
        mock_cursor = MagicMock()
        mock_connect_db.return_value = mock_conn
        mock_conn.cursor.return_value = mock_cursor
        mock_cursor.execute.side_effect = sqlite3.Error("Erro de banco de dados")

        result = atualizar_usuario(user_id, nome=nome)

        assert result["success"] is False
        assert result["message"] == "Erro de banco de dados"
```

Cadastro de Novo Livro:

```python
import pytest
import sqlite3
from unittest.mock import patch, MagicMock
from app import cadastrar_livro, is_valid_isbn


def test_cadastrar_livro_sucesso():
    titulo = "O Senhor dos Anéis"
    autor = "J.R.R. Tolkien"
    isbn = "978-3-16-148410-0"
    categoria = "Fantasia"

    with patch("app.connect_db") as mock_connect_db, \
         patch("app.is_valid_isbn", return_value=True):
        mock_conn = MagicMock()
        mock_cursor = MagicMock()
        mock_connect_db.return_value = mock_conn
        mock_conn.cursor.return_value = mock_cursor
        mock_cursor.execute.return_value = None

        result = cadastrar_livro(titulo, autor, isbn, categoria)

        assert result["success"] is True
        assert result["message"] == "Livro cadastrado com sucesso"


def test_cadastrar_livro_campos_faltando():
    result = cadastrar_livro("", "J.K. Rowling", "978-3-16-148410-0", "Fantasia")

    assert result["success"] is False
    assert result["message"] == "Todos os campos são obrigatórios"


def test_cadastrar_livro_isbn_invalido():
    titulo = "Harry Potter e a Pedra Filosofal"
    autor = "J.K. Rowling"
    isbn = "978-3-16-148410-XYZ"
    categoria = "Fantasia"

    with patch("app.is_valid_isbn", return_value=False):
        result = cadastrar_livro(titulo, autor, isbn, categoria)

    assert result["success"] is False
    assert result["message"] == "ISBN inválido"


def test_cadastrar_livro_isbn_ja_cadastrado():
    titulo = "O Hobbit"
    autor = "J.R.R. Tolkien"
    isbn = "978-0-345-33968-3"
    categoria = "Fantasia"

    with patch("app.connect_db") as mock_connect_db, \
         patch("app.is_valid_isbn", return_value=True):
        mock_conn = MagicMock()
        mock_cursor = MagicMock()
        mock_connect_db.return_value = mock_conn
        mock_conn.cursor.return_value = mock_cursor
        mock_cursor.execute.side_effect = sqlite3.IntegrityError("ISBN já cadastrado")

        result = cadastrar_livro(titulo, autor, isbn, categoria)

        assert result["success"] is False
        assert result["message"] == "ISBN já cadastrado"

```

Remoção de Livro:

```python
import pytest
import sqlite3
from unittest.mock import patch, MagicMock
from app import remover_livro


def test_remover_livro_sucesso():
    livro_id = 1

    with patch("app.connect_db") as mock_connect_db:
        mock_conn = MagicMock()
        mock_cursor = MagicMock()
        mock_connect_db.return_value = mock_conn
        mock_conn.cursor.return_value = mock_cursor
        mock_cursor.execute.return_value = None
        mock_cursor.rowcount = 1

        result = remover_livro(livro_id)

        assert result["success"] is True
        assert result["message"] == "Livro removido com sucesso"


def test_remover_livro_nao_encontrado():
    livro_id = 999

    with patch("app.connect_db") as mock_connect_db:
        mock_conn = MagicMock()
        mock_cursor = MagicMock()
        mock_connect_db.return_value = mock_conn
        mock_conn.cursor.return_value = mock_cursor
        mock_cursor.execute.return_value = None
        mock_cursor.rowcount = 0

        result = remover_livro(livro_id)

        assert result["success"] is False
        assert result["message"] == "Livro não encontrado"


def test_remover_livro_sem_id():
    result = remover_livro(None)

    assert result["success"] is False
    assert result["message"] == "ID do livro é obrigatório"


def test_remover_livro_erro_bd():
    livro_id = 1

    with patch("app.connect_db") as mock_connect_db:
        mock_conn = MagicMock()
        mock_cursor = MagicMock()
        mock_connect_db.return_value = mock_conn
        mock_conn.cursor.return_value = mock_cursor
        mock_cursor.execute.side_effect = sqlite3.Error("Erro de banco de dados")

        result = remover_livro(livro_id)

        assert result["success"] is False
        assert result["message"].startswith("Erro de banco de dados")

```

Renovação de Empréstimo:

```python
import pytest
import sqlite3
from unittest.mock import patch, MagicMock
from app import renovar_emprestimo

def test_renovar_emprestimo_sucesso():
    usuario_id = 1
    livro_id = 1

    with patch("app.connect_db") as mock_connect_db:
        mock_conn = MagicMock()
        mock_cursor = MagicMock()
        mock_connect_db.return_value = mock_conn
        mock_conn.cursor.return_value = mock_cursor

        mock_cursor.execute.side_effect = [
            MagicMock(),
            MagicMock(),
            MagicMock()
        ]
        mock_cursor.fetchone.side_effect = [['Emprestado'], [1, '2024-08-30']]

        result = renovar_emprestimo(usuario_id, livro_id)

        assert result["success"] is True
        assert result["message"] == "Empréstimo renovado com sucesso"

def test_renovar_emprestimo_nao_encontrado():
    usuario_id = 1
    livro_id = 1

    with patch("app.connect_db") as mock_connect_db:
        mock_conn = MagicMock()
        mock_cursor = MagicMock()
        mock_connect_db.return_value = mock_conn
        mock_conn.cursor.return_value = mock_cursor

        mock_cursor.execute.side_effect = [MagicMock()]
        mock_cursor.fetchone.return_value = None

        result = renovar_emprestimo(usuario_id, livro_id)

        assert result["success"] is False
        assert result["message"] == 'Livro não encontrado'

def test_renovar_emprestimo_livro_disponivel():
    usuario_id = 1
    livro_id = 1

    with patch("app.connect_db") as mock_connect_db:
        mock_conn = MagicMock()
        mock_cursor = MagicMock()
        mock_connect_db.return_value = mock_conn
        mock_conn.cursor.return_value = mock_cursor

        mock_cursor.execute.side_effect = [MagicMock()]
        mock_cursor.fetchone.side_effect = ['Disponível']

        result = renovar_emprestimo(usuario_id, livro_id)

        assert result["success"] is False
        assert result["message"] == "O livro não está marcado como emprestado"


def test_renovar_emprestimo_nao_emprestado():
    usuario_id = 1
    livro_id = 1

    with patch("app.connect_db") as mock_connect_db:
        mock_conn = MagicMock()
        mock_cursor = MagicMock()
        mock_connect_db.return_value = mock_conn
        mock_conn.cursor.return_value = mock_cursor
```

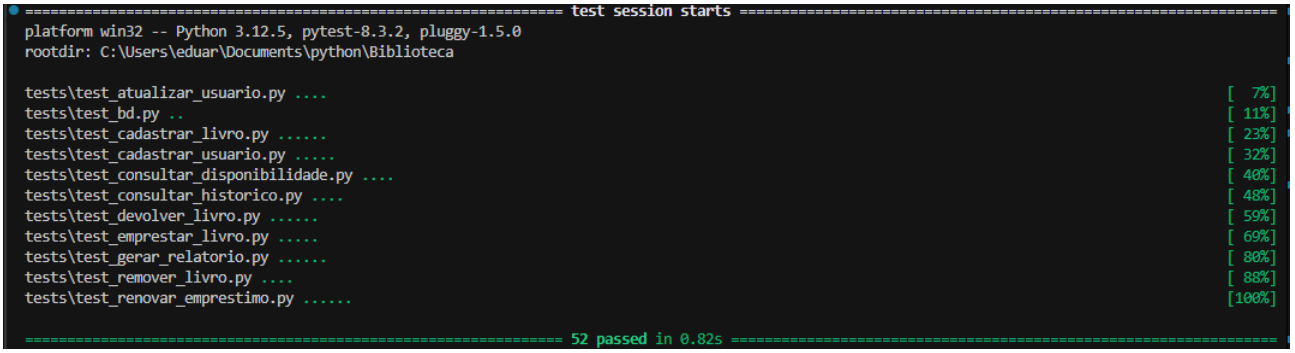
Consulta de Histórico de Empréstimo:

```python
import pytest
import sqlite3
from unittest.mock import patch, MagicMock
from app import consultar_historico


def test_consultar_historico_sucesso():
    usuario_id = 1

    with patch("app.connect_db") as mock_connect_db:
        mock_conn = MagicMock()
        mock_cursor = MagicMock()
        mock_connect_db.return_value = mock_conn
        mock_conn.cursor.return_value = mock_cursor

        mock_cursor.execute.side_effect = [MagicMock()]
        mock_cursor.fetchall.return_value = [
            ('Livro A', '2024-08-01', '2024-08-15'),
            ('Livro B', '2024-08-10', '2024-08-24')
        ]

        result = consultar_historico(usuario_id)

        assert result["success"] is True
        assert len(result["historico"]) == 2
        assert result["historico"][0] == ('Livro A', '2024-08-01', '2024-08-15')


def test_consultar_historico_vazio():
    usuario_id = 1

    with patch("app.connect_db") as mock_connect_db:
        mock_conn = MagicMock()
        mock_cursor = MagicMock()
        mock_connect_db.return_value = mock_conn
        mock_conn.cursor.return_value = mock_cursor

        mock_cursor.execute.side_effect = [MagicMock()]
        mock_cursor.fetchall.return_value = []

        result = consultar_historico(usuario_id)

        assert result["success"] is False
        assert result["message"] == "Nenhum histórico encontrado"


def test_consultar_historico_sem_id():
    result = consultar_historico(None)

    assert result["success"] is False
    assert result["message"] == "ID do usuário é obrigatório"


def test_consultar_historico_erro_bd():
    usuario_id = 1

    with patch("app.connect_db") as mock_connect_db:
        mock_conn = MagicMock()
        mock_cursor = MagicMock()
        mock_connect_db.return_value = mock_conn
        mock_conn.cursor.return_value = mock_cursor
        mock_cursor.execute.side_effect = sqlite3.Error("Erro de banco de dados")
```

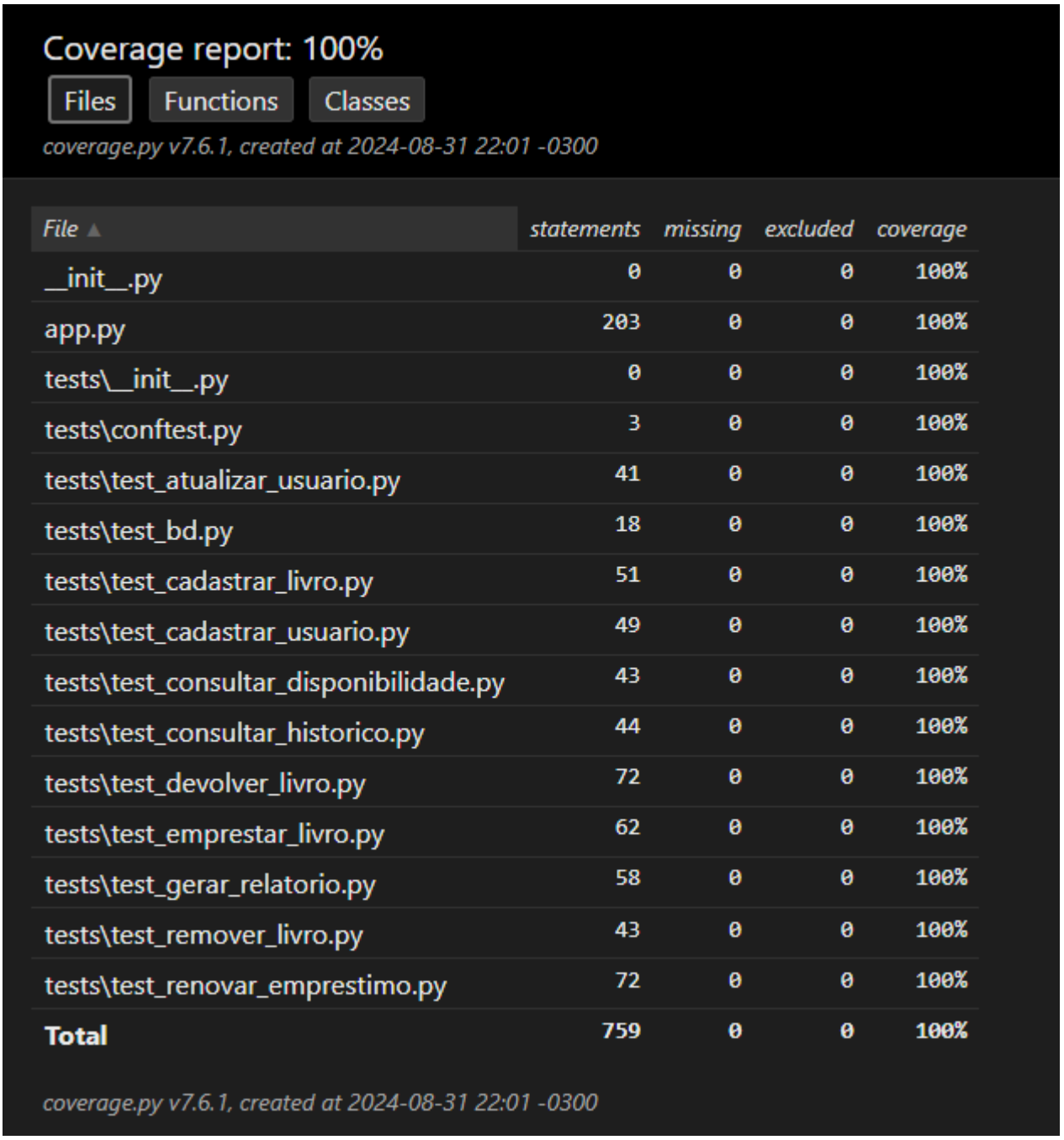
Geração de Relatórios de Livros:

```python
import pytest
import sqlite3
from unittest.mock import patch, MagicMock
from app import gerar_relatorio


def test_gerar_relatorio_emprestados():
    with patch("app.connect_db") as mock_connect_db:
        mock_conn = MagicMock()
        mock_cursor = MagicMock()
        mock_connect_db.return_value = mock_conn
        mock_conn.cursor.return_value = mock_cursor

        mock_cursor.execute.return_value = None
        mock_cursor.fetchall.return_value = [
            ('Livro 1', '2024-09-14'),
            ('Livro 2', '2024-09-15')
        ]

        result = gerar_relatorio('emprestados')

        assert result == {
            "success": True,
            "data": [
                ('Livro 1', '2024-09-14'),
                ('Livro 2', '2024-09-15')
            ]
        }


def test_gerar_relatorio_disponiveis():
    with patch("app.connect_db") as mock_connect_db:
        mock_conn = MagicMock()
        mock_cursor = MagicMock()
        mock_connect_db.return_value = mock_conn
        mock_conn.cursor.return_value = mock_cursor

        mock_cursor.execute.return_value = None
        mock_cursor.fetchall.return_value = [
            ('Livro 1',),
            ('Livro 3',)
        ]

        result = gerar_relatorio('disponiveis')

        assert result == {
            "success": True,
            "data": [
                ('Livro 1',),
                ('Livro 3',)
            ]
        }

def test_gerar_relatorio_atraso():
    with patch("app.connect_db") as mock_connect_db:
        mock_conn = MagicMock()
        mock_cursor = MagicMock()
        mock_connect_db.return_value = mock_conn
        mock_conn.cursor.return_value = mock_cursor

        mock_cursor.execute.return_value = None
        mock_cursor.fetchall.return_value = [
            ('Livro 1', '2024-08-28'),
            ('Livro 2', '2024-08-27')
        ]

        result = gerar_relatorio('atraso')
```

Resultado dos testes:



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