Plano de testes - API ServeRest

Versão 2.0

📜 Histórico de Revisões

Histórico de Revisões			
Versão	**Data**	**Descrição**	**Autor(a)**
1.0	6 de mai. de 2025	Elaboração Inicial	Carolina Hoewell
2.0	30 de mai. de 2025	Atualização de informações	Carolina Hoewell

---

📑 Sumário

---

1. Introdução 📘

Este plano de testes descreve a estratégia e a abordagem para testar a API ServeRest. Ela é uma API REST para automação de testes, que simula um sistema de e-commerce com funcionalidades como cadastro de usuários, login, gerenciamento de produtos, carrinhos e pedidos. O objetivo deste plano de testes é detalhar a estratégia para a análise aprofundada e a verificação das funcionalidades da API ServeRest. Garantindo, assim, que a API funcione conforme o esperado em termos de funcionalidade, confiabilidade, desempenho, segurança, atendendo aos requisitos de negócio e proporcionando uma experiência de integração eficaz e confiável.

O plano abrangerá a validação individual de cada endpoint e suas respectivas operações (como `GET`, `POST`, `PUT`, `DELETE`), a correta interpretação e processamento dos dados de entrada e saída, o tratamento adequado de diferentes cenários (incluindo

casos de sucesso e de erro), e a conformidade com as especificações da API. Serão empregadas técnicas de teste funcional para verificar o comportamento de cada funcionalidade em detalhes.

1.1 Objetivos do Teste

Este plano de testes serve como um guia. Ele pode ser ajustado conforme a evolução do projeto e as necessidades específicas, tendo como principais objetivos:

- Verificar se todas as funcionalidades da API ServeRest estão implementadas corretamente.
- Identificar e reportar defeitos.
- Garantir que a API seja estável e confiável.
- Validar que a API lida adequadamente com entradas inválidas e cenários de erro.
- Assegurar que os endpoints protegidos só possam ser acessados com autenticação válida.
- Verificar se a estrutura das respostas da API (JSON) está correta.

2. Escopo do Projeto 📌

2.1. Dentro do Escopo

2.1.1. Requisitos de Teste

2.1.1.1. Requisitos Funcionais

ID do Requisito Funcional	Nome do Requisito Funcional	Descrição do Requisito Funcional
RF01	Autenticação	Validar autenticação de login de usuários.
RF02	Cadastro de Usuários	Validar cadastro de usuários pelo administrador.
RF03	Lista de Usuários Cadastrados	Validar listagem de usuários cadastrados.
RF04	Busca de Usuário Cadastrados por ID	Validar ID de usuários cadastrados.
RF05	Edição de Usuário Cadastrado	Validar a edição de usuário cadastrado.
RF06	Exclusão de Usuário Cadastrado	Validar a exclusão de usuário pelo administrador.
RF07	Cadastro de Produtos	Validar cadastro de produtos pelo administrador.
RF08	Lista de Produtos Cadastrados	Validar listagem de produtos cadastrados.
RF09	Busca de Produto Cadastrado por ID	Validar ID de produto cadastrado.
RF10	Edição de Produto Cadastrado	Validar a edição de produto cadastrado.
RF11	Exclusão de Produto Cadastrado	Validar a exclusão de produto pelo administrador.

RF12	Cadastro de Carrinho	Validar cadastro de carrinho pelo usuário.
RF13	Lista de Carrinhos Cadastrados	Validar listagem de carrinhos cadastrados.
RF14	Buscar Carrinho Cadastrado por ID	Validar ID de carrinho cadastrado.
RF15	Conclusão de Compra	Validar a conclusão de uma compra.
RF16	Cancelamento de Compra	Validar o cancelamento de uma compra.
RF17	Excluir Carrinho Cadastrado	Validar a exclusão de carrinho pelo usuário.
RF18	Retornar Produtos para Estoque	Validar o retorno de produto para o estoque após exclusão de carrinho.
RF19	Relatórios	Validar geração e exibição de relatórios.

2.1.1.2 - Requisitos Não-Funcionais

ID do Requisito Não-Funcional	Nome do Requisito Não-Funcional	Descrição do Requisito Não-Funcional
RNF01	Códigos de Status HTTP	Validar se a API retorna os códigos de status corretos para diferentes cenários (ex: 200, 201, 400, 401, 403, 404, 500).
RNF02	Contratos da API	Garantir que as respostas da API estejam em conformidade com o contrato definido (JSON schema, tipos de dados, campos obrigatórios/opcionais).
RNF03	Segurança Básica	Testar a proteção de endpoints que requerem autenticação.
RNF04	Desempenho Básico	Avaliar o tempo de resposta da API sob carga normal (testes exploratórios) que deverão ser inferiores a 2 segundos.
RNF05	Segurança	Dados sensíveis devem ser criptografados.
RNF06	Usabilidade	Interface intuitiva e responsiva.
RNF07	Compatibilidade	Funcionamento em navegadores modernos.
RNF08	Acessibilidade	Usabilidade para usuários com deficiência conforme as diretrizes WCAG 2.1.

2.2 - Fora do Escopo

O que NÃO será testado (sugestões, podem variar conforme o projeto):

- Testes de carga e estresse exaustivos (a menos que especificado).
- Testes de penetração aprofundados.
- Testes de usabilidade da documentação da API.
- Testes de compatibilidade com diferentes frameworks de front-end (foco na API em si).
- Pagamentos (gateways) e Localização (traduções).
- Dispositivos como TVs, relógios, IoT, etc.
- Sistemas Operacionais como Windows <10, Android <11, iOS <15.
- Ferramentas diferenciadas como emuladores não oficiais, VPN's.

3. Estratégias de Testes 🔍

Cada estratégia foi escolhida para ampliar a cobertura da aplicação, maximizando a chance de encontrar falhas que impactem diretamente a experiência do usuário.

As sessões foram registradas com evidências visuais e descrições textuais para discussão e apresentação final.

Serão aplicados os seguintes tipos de testes e de acordo com a possibilidade eles serão automatizados:

- **Testes Funcionais:**
  - Testar cada endpoint individualmente (operações CRUD para usuários, produtos, carrinhos).
  - Testar fluxos de ponta a ponta (ex: cadastrar usuário, logar, cadastrar produto, criar carrinho, adicionar produto ao carrinho, finalizar compra).
  - Validação de schemas de requisição e resposta.
- **Testes de Validação de Dados:**
  - Testar com dados válidos.
  - Testar com dados inválidos (tipos incorretos, formatos inválidos, valores fora dos limites).
  - Testar com dados obrigatórios ausentes.
  - Testar com campos opcionais.
- **Testes de Segurança (Básicos):**
  - Tentar acessar endpoints protegidos sem token de autenticação.
  - Tentar acessar endpoints protegidos com token inválido ou expirado.
  - Verificar se dados sensíveis (como senhas) não são retornados nas respostas.
- **Testes de Erro e Tratamento de Exceções:**
  - Verificar se a API retorna mensagens de erro claras e códigos de status HTTP apropriados.
  - Testar cenários que podem causar erros no servidor (ex: tentar excluir um usuário que possui um carrinho ativo).
- **Testes de Contrato (Schema Validation):**
  - Validar se as respostas da API seguem o schema JSON esperado.
- **Testes Exploratórios:**
  - Testes manuais para descobrir cenários não cobertos pelos casos de teste formais.

4. Recursos 🧰

4.1 - Ambiente de Teste

- **URL Base da API:** `https://compassuol.serverest.dev`
- **Ferramentas de Teste:**
  - Postman para testes manuais e automatizados.

    - Frameworks de automação serão as Requests (Python).
- **Dados de Teste:** Serão gerados dados específicos para cada cenário de teste. Dados sensíveis como senhas serão fictícios.
- **Hardware**:
    - Desktop/Notebook: processador: 4vCPUs, 8GB RAM ou superior.
- **Software**:
    - SO Desktop/Notebook: Windows 10/11
    - SO Dispositivos Móveis: Android 11 (Red Velvet Cake) e IOS 15 ou superiores.
    - Navegadores Desktop/Notebook: Chrome, EDGE e Firefox.
    - Navegadores Dispositivos Móveis: Chrome (Android) e Safari (IOS).

4.2 - Ferramentas

- **Funcionais**:
    - Chrome, EDGE, Firefox, (para testes de interface).
    - Inspeção de elementos do navegador (F12).
- **Não-funcionais**:
    - Planilha para registro de métricas (Confluence).
    - BrowerStack para testes de compatibilidade.
    - Lighthouse para teste de desempenho.
- **Gerenciamento**:
    - Xmind para mapa mental.
    - Confluence.
    - Jira.
    - Microsoft Teams para comunicação.

5. Critérios de Entrada e Saída 🏁

5.1. Critérios de Entrada:

- A aplicação deverá estar online durante as sessões de teste.
- Os testes deverão ser executados utilizando a documentação da API (Swagger/OpenAPI) como guia, estando ela disponível e atualizada.
- Ambiente de teste configurado e estável, sendo as sessões de teste realizadas em navegadores atualizados.
- Build da API implantado no ambiente de teste.
- Ferramentas de teste instaladas e configuradas.

5.2. Critérios de Saída:

- Todos os casos de teste planejados foram executados.
- Deverá ser realizadas a maior cobertura possível de testes, tendo um percentual de casos de teste aprovados acima de 95%.
- Nenhum defeito crítico (blocker/critical) em aberto.
- Relatório de resumo dos testes gerado e revisado.

6. Casos de Teste (Categorias) 🧪

Serão criados casos de teste detalhados para cada um dos seguintes módulos/endpoints:

- **Autenticação (`/login`)**
    - Login com sucesso (credenciais válidas).
    - Tentativa de login com senha incorreta.

    - Tentativa de login com e-mail inexistente.
    - Tentativa de login com campos obrigatórios ausentes (e-mail, senha).
    - Tentativa de login com e-mail em formato inválido.
- **Usuários (`/usuarios`)**
    - `POST /usuarios`: Cadastrar novo usuário com sucesso.
    - `POST /usuarios`: Tentar cadastrar usuário com e-mail já existente.
    - `POST /usuarios`: Tentar cadastrar usuário com campos obrigatórios ausentes.
    - `POST /usuarios`: Tentar cadastrar usuário com dados inválidos (ex: e-mail sem "@").
    - `GET /usuarios`: Listar todos os usuários.
    - `GET /usuarios?_id={_id}`: Buscar usuário por ID existente.
    - `GET /usuarios?_id={_id}`: Buscar usuário por ID inexistente.
    - `PUT /usuarios/{_id}`: Editar usuário existente com sucesso (requer autenticação como admin).
    - `PUT /usuarios/{_id}`: Tentar editar usuário com e-mail já existente (de outro usuário).
    - `PUT /usuarios/{_id}`: Tentar editar usuário inexistente.
    - `PUT /usuarios/{_id}`: Tentar editar sem autenticação ou com token inválido.
    - `DELETE /usuarios/{_id}`: Excluir usuário existente com sucesso (requer autenticação como admin).
    - `DELETE /usuarios/{_id}`: Tentar excluir usuário inexistente.
    - `DELETE /usuarios/{_id}`: Tentar excluir sem autenticação ou com token inválido.
- **Produtos (`/produtos`)**
    - `POST /produtos`: Cadastrar novo produto com sucesso (requer autenticação).
    - `POST /produtos`: Tentar cadastrar produto com nome já existente.
    - `POST /produtos`: Tentar cadastrar produto com campos obrigatórios ausentes.
    - `POST /produtos`: Tentar cadastrar produto com dados inválidos (ex: preço negativo).
    - `POST /produtos`: Tentar cadastrar sem autenticação ou com token inválido.
    - `GET /produtos`: Listar todos os produtos.
    - `GET /produtos?_id={_id}`: Buscar produto por ID existente.
    - `GET /produtos?_id={_id}`: Buscar produto por ID inexistente.
    - `PUT /produtos/{_id}`: Editar produto existente com sucesso (requer autenticação).
    - `PUT /produtos/{_id}`: Tentar editar produto com nome já existente (de outro produto).
    - `PUT /produtos/{_id}`: Tentar editar produto inexistente.
    - `PUT /produtos/{_id}`: Tentar editar sem autenticação ou com token inválido.
    - `DELETE /produtos/{_id}`: Excluir produto existente com sucesso (requer autenticação).
    - `DELETE /produtos/{_id}`: Tentar excluir produto inexistente.
    - `DELETE /produtos/{_id}`: Tentar excluir sem autenticação ou com token inválido.
- **Carrinhos (`/carrinhos`)**
    - `POST /carrinhos`: Criar carrinho com sucesso (requer autenticação).
    - `POST /carrinhos`: Tentar criar carrinho com produto inexistente.
    - `POST /carrinhos`: Tentar criar carrinho com quantidade de produto inválida.
    - `POST /carrinhos`: Tentar criar carrinho sem autenticação ou com token inválido.
    - `POST /carrinhos`: Tentar cadastrar produto duplicado no mesmo carrinho (verificar regra de negócio).
    - `GET /carrinhos`: Listar todos os carrinhos.
    - `GET /carrinhos?_id={_id}`: Buscar carrinho por ID existente.
    - `GET /carrinhos?_id={_id}`: Buscar carrinho por ID inexistente.
    - `DELETE /carrinhos/concluir-compra`: Concluir compra de um carrinho existente (requer autenticação).
    - `DELETE /carrinhos/concluir-compra`: Tentar concluir compra de carrinho inexistente.

- `DELETE /carrinhos/concluir-compra`: Tentar concluir compra sem autenticação ou com token inválido.
- `DELETE /carrinhos/cancelar-compra`: Cancelar compra (devolve produtos ao estoque) de um carrinho existente (requer autenticação).
- `DELETE /carrinhos/cancelar-compra`: Tentar cancelar compra de carrinho inexistente.
- `DELETE /carrinhos/cancelar-compra`: Tentar cancelar compra sem autenticação ou com token inválido.

7. Priorização dos Cenários 🔢

ID Caso de Teste	Caso de Teste	Status do teste	Prioridade do Teste (Base de Impacto)	Probabilidade	Impacto	Justificativas
CT-001	Login com sucesso (credenciais válidas)		Crítico	P1		Funcionalidade essencial; falha impede qualquer acesso. É um fluxo comum, mas sensível a mudanças.
CT-002	Tentativa de login com senha incorreta		Médio	P3		Segurança e UX; sistema deve responder adequadamente. PF Média para garantir tratamento correto de erro comum.
CT-003	Tentativa de login com e-mail inexistente		Médio	P3		Segurança e UX; sistema deve responder adequadamente sem vazar informação se usuário existe ou não.
CT-004	Tentativa de login com campos obrigatórios ausentes (e-mail, senha)		Médio	P3		Validação de entrada básica; importante para UX.
CT-005	Tentativa de login com e-mail em formato inválido		Baixo	P4		Validação de formato; menor impacto se falhar comparado a outros cenários de login.
CT-006	Cadastrar novo usuário com sucesso		Alto	P2		Fluxo essencial para novos usuários. PF Média devido à complexidade de validações e persistência.
CT-007	Tentar cadastrar usuário com e-mail já existente		Médio	P3		Tratamento de duplicidade é importante para integridade dos dados e UX.
CT-008	Tentar cadastrar usuário com campos obrigatórios ausentes		Médio	P3		Validação de entrada básica.
CT-009	Tentar cadastrar usuário com e-mail com formato inválido		Baixo	P4		Validação de formato.
CT-010	Tentar cadastrar usuário com e-mail do domínio 'gmail.com' e 'hotmail.com '		Baixo	P4		Regra de negócio específica, provavelmente menos crítica que validações genéricas.
CT-011	Tentar cadastrar usuário com senha de 4 caracteres		Baixo	P4		Teste de regra de negócio específica para senha; impacto depende da política de segurança. Assumindo que há outras validações mais fortes.
CT-012	Tentar cadastrar usuário com senha de 17 caracteres		Baixo	P4		Teste de regra de negócio específica para senha.

CT-013	Listar todos os usuários		Médio	P3		Funcionalidade de admin ou para certos contextos; pode expor dados sensíveis se não protegida. PF Média pela sensibilidade e paginação/filtros. IF Médio se acessível indevidamente.
CT-014	Buscar usuário por ID existente		Baixo	P4		Funcionalidade básica de busca.
CT-015	Buscar usuário por ID inexistente		Baixo	P4		Tratamento de erro comum.
CT-016	Editar usuário existente com sucesso		Alto	P2		Admin pode precisar editar dados; falha impede gestão. Integridade de dados é crucial.
CT-017	Tentar editar usuário com e-mail já existente		Médio	P3		Validação de unicidade de e-mail na edição.
CT-018	Tentar editar usuário inexistente		Baixo	P4		Tratamento de erro.
CT-019	Editar usuário sem autenticação ou com token inválido		Crítico	P1		Violação de segurança, acesso não autorizado a modificação de dados. PF Alta pois é um vetor de ataque comum.
CT-020	Excluir usuário existente com sucesso		Alto	P2		Operação sensível, pode ter impacto em dados relacionados. Falha impede gestão.
CT-021	Tentar excluir usuário inexistente		Baixo	P4		Tratamento de erro.
CT-022	Excluir usuário sem autenticação ou com token inválido		Crítico	P1		Violação de segurança, exclusão não autorizada.
CT-023	Cadastrar novo produto com sucesso, com autenticação ou token válido		Alto	P2		Essencial para o negócio (e-commerce). PF Média devido a validações e integrações.
CT-024	Tentar cadastrar sem autenticação ou com token inválido		Crítico	P1		Falha de segurança, criação indevida de dados.
CT-025	Tentar cadastrar produto com nome já existente		Médio	P3		Regra de negócio para evitar duplicidade ou gerenciar versões.
CT-026	Tentar cadastrar produto com campos obrigatórios ausentes		Médio	P3		Validação básica.
CT-027	Tentar cadastrar produto com		Médio	P3		Validação de dados para garantir consistência e evitar problemas (ex: financeiros).

	dados inválidos (ex: preço negativo)					
CT-028	Listar todos os produtos		Alto	P2		Essencial para visualização em loja, admin. PF Média devido a paginação, filtros.
CT-029	Buscar produto por ID existente		Baixo	P4		Busca específica, funcionalidade básica.
CT-030	Buscar produto por ID inexistente		Baixo	P4		Tratamento de erro.
CT-031	Editar produto existente com sucesso		Alto	P2		Manutenção de informações de produto é crucial.
CT-032	Tentar editar produto com nome já existente		Médio	P3		Validação de unicidade de nome na edição.
CT-033	Tentar editar produto inexistente		Baixo	P4		Tratamento de erro.
CT-034	Editar produto sem autenticação ou com token inválido		Crítico	P1		Violação de segurança.
CT-035	Tentar cadastrar produto através de PUT ID inexistente		Baixo	P4		Interpreto como tentar usar PUT para criar se o ID não existe; comportamento depende da API (UPSERT ou erro). Se erro não tratado, pode ser M. Assumindo tratamento.
CT-036	Excluir produto existente com sucesso		Alto	P2		Operação sensível, pode impactar vendas se produto ainda é relevante.
CT-037	Tentar excluir produto inexistente		Baixo	P4		Tratamento de erro.
CT-038	Excluir produto sem autenticação ou com token inválido		Crítico	P1		Violação de segurança.
CT-039	Tentar excluir produto vinculado a um carrinho		Alto	P2		Regra de negócio importante; pode causar inconsistência no carrinho ou pedido se não tratada.
CT-040	Criar carrinho com sucesso		Alto	P2		Essencial para o fluxo de compra.
CT-041	Tentar criar carrinho com quantidade de produto inválida		Médio	P3		Validação de dados do carrinho.
CT-042	Tentar criar carrinho sem autenticação ou com token inválido		Crítico	P1		Violação de segurança ou falha em associar carrinho ao usuário.
CT-043	Tentar cadastrar produto já		Médio	P3		Regra de negócio para adição de itens (ex: incrementar quantidade vs. impedir).

	aplicado no mesmo carrinho (verificar regra de negócio)					
CT-044	Tentar criar segundo carrinho para o mesmo usuário		Médio	P3		Regra de negócio (sistema permite múltiplos carrinhos ou um por usuário?). Falha pode gerar confusão.
CT-045	Listar todos os carrinhos		Médio	P3		Geralmente para admin ou debug. Se exposto a usuários, pode ser falha de privacidade/segurança. Assumindo uso interno/admin.
CT-046	Buscar carrinho por ID existente		Baixo	P4		Funcionalidade básica.
CT-047	Buscar carrinho por ID inexistente		Baixo	P4		Tratamento de erro.
CT-048	Concluir compra de um carrinho existente		Crítico	P1		Ponto mais crítico do e-commerce. Envolve pagamento, estoque, pedidos. PF Alta pela complexidade.
CT-049	Tentar concluir compra de carrinho inexistente		Baixo	P4		Tratamento de erro, mas tentativa de finalizar compra inexistente pode indicar problema no fluxo.
CT-050	Tentar concluir compra sem autenticação ou com token inválido		Crítico	P1		Violação de segurança grave, processamento de compra não autorizada ou falha na atribuição.
CT-051	Cancelar compra (devolve produtos ao estoque) de um carrinho existente		Alto	P2		Importante para UX e gestão de estoque.
CT-052	Tentar cancelar compra de carrinho inexistente		Baixo	P4		Tratamento de erro.
CT-053	Tentar cancelar compra sem autenticação ou com token inválido		Crítico	P1		Violação de segurança ou cancelamento indevido.

Risco	Probabilidade	Impacto	Mitigação
Ambiente de teste instável ou indisponível.	Média	Alto	Comunicação com a equipe de infra/devops; agendar janelas de teste.

Documentação da API desatualizada.	Média	Médio	Revisão da documentação antes do início dos testes; comunicação com devs.
Mudanças frequentes na API durante o ciclo.	Alta	Alto	Testes automatizados e regressivos; comunicação constante com a equipe.
Falta de dados de teste adequados.	Baixa	Médio	Planejamento e criação antecipada de massa de dados.
Prazos apertados.	Média	Alto	Priorização de casos de teste críticos; automação de testes.

7. Completude dos Testes ✅

- Todos os casos de teste planejados serem devidamente executados.
- Testes de regressão serão executados após correções de bugs.
- O cronograma pode ser ajustado conforme necessidade do projeto.
- Cada membro da Squad executou sua técnica com duração de até 45 minutos.
- Todos os pontos principais da aplicação foram acessados durante os testes.
- Pelo menos 1 comportamento relevante identificado por sessão.

8. Priorização dos Testes em Caso de Mudança de Escopo 🔄

- Reunir PO, SM, Devs e QA
- Reavaliar:
  - Testes críticos (não podem ser cortados)
  - Novos testes necessários.
- Ajustar plano e prazos.

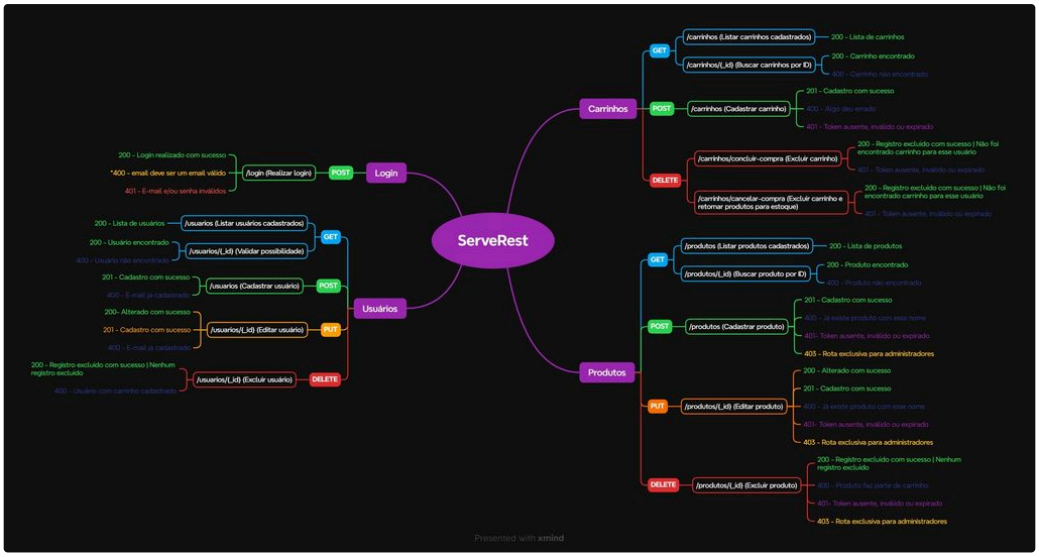
9. Artefatos 📂

9.1 Entrada

Acesso ao site: Front - ServeRest

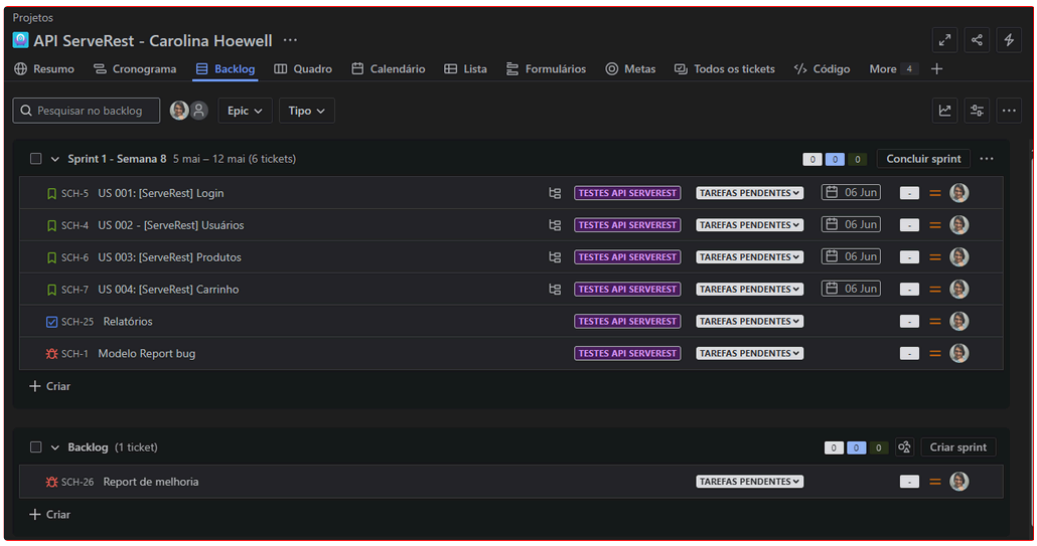
Página inicial da API ServRest

Mapa Mental do site:



Mapa mental da API ServeRest

Histórias de usuários:



Estrutura básica do Épico API ServeRest e suas User Stories

10. Gerenciamento de Comunicação 📢

- *Atualização diária no Confluence.*
- Comunicação via Teams.
- Daily Squad.

- Instabilidade na comunicação com o site durante a execução dos testes.
- Diferença de interpretação entre testadores sobre funcionalidades.
- Registro incompleto de falhas por falta de documentação de apoio.

12. Entregáveis 📑

Este Plano de Testes.

- Casos de Teste (em ferramenta de gerenciamento de testes ou planilha).
- Scripts de Testes Automatizados (se aplicável).
- Relatórios de Execução de Testes.
- Relatório de Defeitos (bugs).
- Relatório Final de Testes (com resumo dos resultados e aprovação/reprovação).

13. Cronograma e Marcos 📅

Etapa	Data
Elaboração do Plano de Teste	6 de mai. de 2025
Definição das Técnicas	6 de mai. de 2025
Execução dos Testes Exploratórios	8 de mai. de 2025
Elaboração dos Reports dos Artefatos de Saída	9 de mai. de 2025
Apresentação dos Reports dos Artefatos de Saída	9 de mai. de 2025

14. Aprovações ✅

Aprovações		
Participante	**Assinatura**	**Data**
Carlos Leonardo Alves Novaes		
Jacques de Jesus Figueiredo Schmitz Junior		
Amanda Cardoso de Almeida		