


## ServerRest - Planejamento de Testes



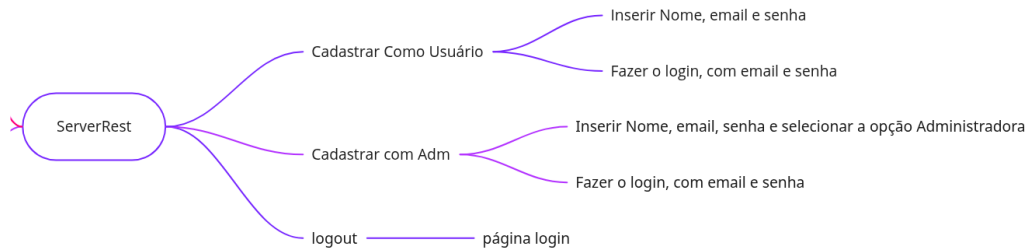
Proprietário: douglas paulo Cortes . . .

Última atualização em: ontem às 10:47 PM •  Veja quantas pessoas visualizaram a página

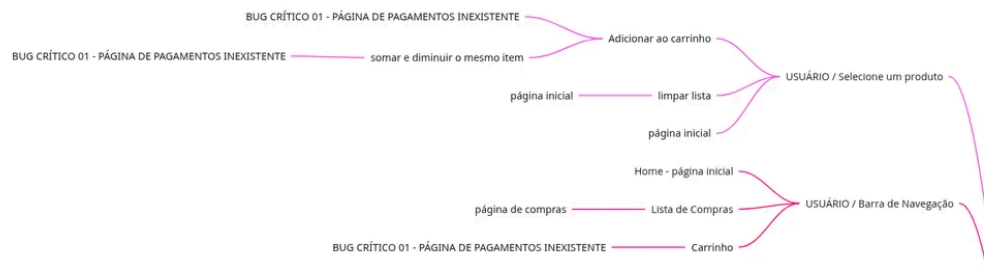
## Visão Geral do Projeto

- **Nome do Projeto:** ServerRest - Planejamento de Testes
- **Objetivo:** Documentar e executar testes exploratórios na aplicação ServerRest
- **Squad:** Nexus 101 ( @douglas paulo Cortes @Karen Késsia @Maria Eduarda Martins Rodrigues @Wesley C. @Anna Santoro )
- **Período:** 22/05/2025 - 25/05/2025
- Backlog:  TrilhaCompass | Backlog
- Ferramentas: Microsoft Excell, Test & Feedback, Ui.Vision IDE

## MAPA MENTAL



## Cadastro e Login



## Interações Usuário/Cliente





Funções de Administrador

→Mapa Completo

A partir do mapeamento realizado, propõe-se a elaboração de um roteiro estruturado de testes. Os fluxos previamente definidos deverão ser contemplados com automações, visando eficiência e repetibilidade nos testes.

Fluxo	Missão (Charter)	Observações/Notas Rápidas
Cadastro de usuário	Testar criação de usuário com dados válidos, inválidos (email errado, senha vazia)	
Login	Testar login com usuário correto, usuário inexistente, senha errada	Verificar retorno e erros
Listagem de usuários	Testar listagem pública, filtros se existirem, paginação (validar campos retornados)	
Atualização de usuário	Testar editar nome, email de usuário existente (e inexistente), sem token e com token inválido	Status correto: não autorizado
Exclusão de usuário	Testar exclusão de usuário existente, inexistente, sem token e com token inválido	Verificar mensagens de sucesso e erro
Token de autenticação	Testar rotas privadas sem token, com token inválido e com token expirado (se simular expiração)	Mensagens corretas?
Validação de mensagens de erro	Analisar se as mensagens de erro são claras, corretas e	

## 2. Estratégia de testes

### 2.1 Abordagem Geral

Inicialmente, serão listados os requisitos funcionais e restrições, com o objetivo de compreender de forma clara os critérios de aceite que a aplicação deve atender para cumprir seus propósitos de negócio e técnicos.

### Requisitos Funcionais (RF)

Código	Descrição
RF01	Permitir o cadastro de novos usuários com nome, email e senha válidos.
RF02	Permitir login de usuários existentes utilizando email e senha válidos.
RF03	Retornar um <b>token de autenticação</b> válido após login bem-sucedido.
RF04	Permitir a listagem de todos os usuários cadastrados.
RF05	Permitir a atualização (edição) de informações do usuário, como nome e email.
RF06	Permitir a exclusão de usuários do sistema.
RF07	Proteger rotas de atualização e exclusão de usuário utilizando autenticação via autenticação
RF08	Retornar mensagens de erro claras e específicas em casos de falhas (ex: cadastro inválido, login incorreto, autenticação inválida).

### Restrições

Código	Descrição
R01	Não permitir criação de usuários com emails duplicados.
R02	Senhas devem ter um tamanho mínimo (ex: 6 caracteres ou mais).

2.2 Estratégia de testes

Serão descritas as diferentes técnicas de teste a serem aplicadas, distribuídas entre os membros da equipe conforme suas especializações e responsabilidades. Cada técnica será acompanhada de uma justificativa clara, fundamentada nos objetivos do projeto, no tipo de aplicação e nos riscos identificados.

Também serão apresentadas as heurísticas adotadas para apoiar a elaboração dos cenários e casos de teste, garantindo cobertura adequada e priorização baseada em criticidade.

**Importante destacar:** será estimado o número total de baterias de testes previstas, com o intuito de avaliar a viabilidade e o custo-benefício da automação, considerando o esforço necessário para manutenção e execução contínua dos testes automatizados.

Estratégia	Descrição	Aplicação no Projeto ServeRest
Heurísticas de Consistência e Erro	Avaliar se as respostas da API são consistentes, se erros seguem padrões claros e se não há quebras inesperadas.	Testes de mensagens de erro (ex: email duplicado, senha inválida), status codes corretos (ex: 400, 401, 404).
Tours focados em Segurança	Explorar o sistema tentando identificar falhas de autenticação e autorização.	Testar acesso sem autenticação, com token inválido, acesso a rotas protegidas.
Charters de Cadastro e Login	Missões específicas para testar profundamente o cadastro e login, usando sessões exploratórias focadas.	Avaliar fluxo de criação de usuário, login, comportamento em erros de credenciais.
Mind Map baseado em Fluxos mapeados	Utilizar o mapa mental dos fluxos da aplicação para guiar a exploração dos testes.	Garantir que todos caminhos principais e alternativos da API estão cobertos nos testes exploratórios.
Heurística CRUD	Validar as operações de Create, Read, Update e Delete corretamente.	Verificar se cadastro, consulta, atualização e remoção de usuários estão funcionando conforme esperado.
Heurística CHIQUE	Lembrar dos princípios: <b>C</b> onsistência, <b>H</b> istória, <b>I</b> nformações, <b>Q</b> ueries, <b>E</b> struturas. Ajuda a explorar diferentes perspectivas da aplicação.	Aplicar análises: consistência de respostas, histórico de usuários criados/deletados, comportamento em queries, estrutura das respostas JSON.

3. Planejamento Detalhado

/ ServerRest - Planejamento de Testes1 link do JIRA

acompanhamento estruturado das execuções.

Além disso, será avaliada e selecionada uma ferramenta de testes que ofereça cobertura adequada aos fluxos já mapeados e explorados, de forma a garantir maior confiabilidade e eficiência durante as fases de validação.

3.2 Cronograma

Será realizada uma reunião inicial com a equipe para definição de responsabilidades e distribuição das tarefas de acordo com as competências individuais.

Em seguida, será estabelecido um fluxo de trabalho colaborativo, com papéis e entregas bem definidos.

Por fim, será proposto um cronograma com prazos realistas para cada etapa, prevendo uma reunião final para alinhamento e encerramento formal da entrega.

4. Baterias de Testes

Os fluxos listados a seguir representam os cenários ideais para automação de testes, visando maximizar a cobertura, reduzir o retrabalho manual e garantir consistência nas execuções.

Bateria	Fluxo	Item do fluxo	Subitem	Relevância para Automação
1	Cadastro	Válido	Senha exatamente o mínimo permitido	
1	Cadastro	Válido	Senha acima do mínimo permitido	
1	Cadastro	E-mail já existe	-	
1	Cadastro	Sem nada	-	
1	Cadastro	Inválido	Senha abaixo do mínimo permitido	
1	Cadastro	Inválido	E-mail inválido	
1	Retornar mensagens de erro	Visualizar em todos os testes quais são as mensagens de erro	-	
2	Executar o Fluxo anterior e Corrigir Bugs ou Efeitos colaterais encontrados	-	-	
2	Login	Válido	Como admin	
2	Login	Válido	Como usuário	

2	Login	Inválido	E-mail não existe	
---	-------	----------	-------------------	--

/ ServerRest - Planejamento de Testes

1 link do JIRA



2	Login	Inválido	Senha inválida	
2	Autenticação	Tentar entrar na URL pulando a etapa de login usando guia anônima	-	
2	Retornar mensagens de erro	Visualizar em todos os testes quais são as mensagens de erro	-	
3	Executar o Fluxo anterior e Corrigir Bugs ou Efeitos colaterais encontrados	-	-	
3	Listagem de usuários	Sem autenticação	-	
3	Listagem de usuários	Com autenticação	-	
3	Edição de usuários	Sem autenticação	-	
3	Edição de usuários	Com autenticação	-	
3	Exclusão de usuários	Sem autenticação	-	
3	Exclusão de usuários	Com autenticação	-	
3	Retornar mensagens de erro	Visualizar em todos os testes quais são as mensagens de erro	-	
4	Executar o Fluxo anterior e Corrigir Bugs ou Efeitos colaterais encontrados	-	-	
4	Listagem de produtos	-	-	

4	Edição de produtos	-	-	
---	--------------------	---	---	--

/ ServerRest - Planejamento de Testes

1 link do JIRA



	produtos			
4	Retornar mensagens de erro	Visualizar em todos os testes quais são as mensagens de erro	-	
5	Executar o Fluxo anterior e Corrigir Bugs ou Efeitos colaterais encontrados	-	-	
5	Relatórios	-	-	
5	Barra de navegação	Home	-	
5	Barra de navegação	Cadastrar Usuários	-	
5	Barra de navegação	Listar Usuários	-	
5	Barra de navegação	Relatórios	-	
5	Barra de navegação	Logout	-	
6	Executar o Fluxo anterior e Corrigir Bugs ou Efeitos colaterais encontrados	-	-	
6	Avaliar Métricas de Qualidade	-	-	
7	Correção de produtos	-	-	

Gráfico sem título

● Bateria ● Fluxo





**UI.Vision RPA** foi escolhida como a principal ferramenta para automação dos fluxos de interface do usuário. Trata-se de uma solução de automação visual que opera diretamente no navegador, sendo ideal para testes funcionais, validação de fluxos críticos e repetitivos na camada de apresentação ao usuário.

**Test & Feedback Extension** (Visual Studio / Azure DevOps) será utilizada como ferramenta de apoio para **documentação dos fluxos testados** e **registro de evidências** durante a execução dos testes manuais e automatizados.

## Resultados e Métricas

Os **bugs** identificados durante a execução dos testes serão registrados e classificados conforme sua gravidade e impacto no sistema. Cada defeito será vinculado aos casos de teste relacionados e priorizado de acordo com a sua urgência para a correção, utilizando a ferramenta **Jira**. Será mantido um histórico de erros encontrados, facilitando a rastreabilidade e análise de qualidade ao longo do tempo.

O tempo despendido em cada sessão de teste será monitorado e registrado, permitindo uma análise detalhada da eficiência do processo de automação e dos testes manuais. Esse dado é crucial para identificar pontos de melhoria no fluxo de trabalho e ajustar os testes automatizados para otimizar o tempo de execução, garantindo que os testes sejam realizados de forma eficiente e em tempo hábil.

Será acompanhada a **cobertura de testes**, ou seja, a porcentagem do sistema testado por meio dos testes automatizados em relação ao total de funcionalidades. A cobertura será calculada e monitorada de forma contínua, com o objetivo de garantir que os fluxos críticos, como **Cadastro**, **Login**, **Atualização** e **Exclusão**, estejam bem cobertos.

Diversas métricas de qualidade serão aplicadas para avaliar o desempenho do processo de testes, incluindo:

- **Taxa de Passagem/Fracasso:** A proporção de testes que passaram em relação ao total de testes executados.
- **Taxa de Defeitos:** O número de defeitos encontrados durante os testes em comparação ao número de testes realizados.



- **Tempo de Resolução de Bugs:** A quantidade média de tempo entre a descoberta de um bug e sua correção.

/ ServerRest - Planejamento de Testes

1 link do JIRA



- **Manutenibilidade dos Testes:** A facilidade com que os testes automatizados podem ser atualizados conforme novas funcionalidades ou alterações no sistema.

Essas métricas ajudarão a avaliar a eficácia do processo de testes, identificar áreas de melhoria e fornecer dados quantitativos para a tomada de decisões sobre otimização e ajustes no fluxo de trabalho.

Com base nos resultados obtidos ao longo do ciclo de testes, será realizada uma análise detalhada do desempenho do sistema, destacando tanto os pontos positivos quanto as áreas que apresentaram falhas. A análise levará em conta a cobertura de testes, a quantidade de **bugs encontrados**, o tempo de execução dos testes e a eficácia da automação implementada. Será avaliado, também, o impacto das falhas no sistema e a eficiência das soluções adotadas para resolvê-las.

## Objetivo dos Testes Exploratórios

O objetivo principal dos testes exploratórios na API **ServerRest** é avaliar sua **estabilidade, segurança e consistência**. O foco será em realizar sessões de testes baseadas em diferentes **heurísticas** de usabilidade e funcionalidade, a fim de identificar potenciais falhas ou inconsistências que possam impactar a performance e a experiência do usuário.

Durante os testes, serão mapeados os **fluxos principais** da aplicação, que incluem:

- **Cadastro**
- **Login**
- **Listagem**
- **Atualização**
- **Exclusão**

Esses fluxos serão testados minuciosamente para garantir que todas as funcionalidades estão operando conforme esperado e para detectar possíveis falhas de integração ou usabilidade.

A heurística **CHIQUE** será adotada para guiar as sessões de teste. A heurística abrange os seguintes aspectos:

- **Campos Obrigatórios:** Garantir que os campos obrigatórios sejam devidamente validados e que o sistema não permita o envio de formulários com campos obrigatórios em branco.
- **Habilitar/Desabilitar Formulários:** Verificar se os formulários são corretamente habilitados ou desabilitados conforme o fluxo do usuário e os requisitos do sistema.
- **Interrupção da Ação:** Avaliar o comportamento da aplicação quando uma ação é interrompida ou não concluída, assegurando que o sistema retorne ao estado esperado.
- **Quebra de Fluxos:** Testar cenários em que o fluxo de trabalho é interrompido de forma inesperada, como erros ou falhas, para garantir que o sistema lide corretamente com exceções.
- **Usabilidade dos Menus:** Avaliar a clareza e a acessibilidade dos menus e elementos de navegação, garantindo uma experiência fluida para o usuário.
- **Estouro de Campos:** Verificar se o sistema lida adequadamente com entradas de dados que excedem o tamanho limite dos campos, evitando comportamentos inesperados.



Esses critérios serão aplicados para explorar e avaliar a robustez da **API ServerRest**, assegurando que ela atenda aos requisitos de qualidade e usabilidade definidos.

/ ServerRest - Planejamento de Testes

1 link do JIRA



### Casos de Teste a serem executados

ID	Módulo/Função	Cenário	Tipo de Dado	Pré-condições	Ação	Resultado Esperado	Resultado Encontrado	Observações
TC01	Cadastro	Senha exatamente no mínimo permitido	Dados válidos	-	Realizar cadastro com senha mínima	Cadastro realizado com sucesso		
TC02	Cadastro	Senha acima do mínimo permitido	Dados válidos	-	Realizar cadastro com senha segura	Cadastro realizado com sucesso		
TC03	Cadastro	E-mail já existente	E-mail duplicado	E-mail cadastrado	Tentar cadastrar mesmo e-mail	Exibir mensagem de erro adequada		Validar código de status
TC04	Cadastro	Sem preencher campos	Dados em branco	-	Submeter formulário vazio	Mensagem de erro por campos obrigatórios		
TC05	Cadastro	Senha abaixo do mínimo permitido	Senha curta	-	Preencher senha inválida	Mensagem de erro sobre política de senha		
TC06	Cadastro	E-mail com formato inválido	E-mail inválido	-	Submeter com e-mail mal formatado	Exibir mensagem de erro de e-mail inválido		
TC07	Cadastro/Login/Listagem	Mensagens de erro	Todos os fluxos com falhas	-	Forçar erros intencionais	Todas as mensagens de erro devem estar claras e informativas		Registrar prints e logs
TC08	Cadastro	Correção de bugs encontrados	-	Com base em TC01-07	Aplicar correções após falhas	Repetir testes e validar sucesso		

TC09	Login	Login válido	Credenciais	Usuário	Efetuar login	Acesso ao		
------	-------	--------------	-------------	---------	---------------	-----------	--	--

/ ServerRest - Planejamento de Testes

1 link do JIRA



TC10	Login	Login válido como usuário	Credenciais válidas	Usuário comum	Efetuar login	Acesso ao sistema autorizado		
TC11	Login	E-mail inexistente	Dados inválidos	-	Tentar login com e-mail inexistente	Mensagem de erro: usuário não encontrado		
TC12	Login	Campos em branco	Dados vazios	-	Submeter login em branco	Mensagem de erro por campos obrigatórios		
TC13	Login	Senha incorreta	Credenciais inválidas	-	Submeter senha errada	Mensagem de erro por senha inválida		
TC14	Autenticação	Acesso direto à URL sem login (guia anônima)	Não autenticado	Sem login	Acessar URL protegida diretamente	Redirecionamento para tela de login		
TC15	Login	Mensagens de erro	Todos os fluxos com falhas	-	Forçar erros de autenticação	Mensagens claras e consistentes		
TC16	Login	Correção de bugs encontrados	-	Com base em TC09-15	Repetir testes e validar sucesso	Fluxo de login corrigido		
TC17	Usuários	Listar usuários sem autenticação	Não autenticado	-	Acessar listagem sem login	Bloqueio de acesso e mensagem de permissão negada		
TC18	Usuários	Listar usuários com autenticação	Autenticado	Login realizado	Listar usuários	Lista de usuários carregada com sucesso		
TC19	Usuários	Editar sem autenticação	Não autenticado	-	Acessar edição de	Bloqueio e mensagem		

		o			usuário	de erro de autenticaçã		
--	--	---	--	--	---------	------------------------	--	--

/ ServerRest - Planejamento de Testes

1 link do JIRA

TC20	Usuários	Editar com autenticaçã o	Autenticado	Login realizado	Editar dados do usuário	Edição concluída e refletida na base		
TC21	Usuários	Excluir sem autenticaçã o	Não autenticado	-	Tentar excluir usuário	Bloqueio e mensagem de erro		
TC22	Usuários	Excluir com autenticaçã o	Autenticado	Login realizado	Excluir usuário	Usuário removido com sucesso		Confirmar via GET após exclusão
TC23	Usuários	Mensagens de erro	Fluxos de edição/listag em/exclusão	-	Forçar erros diversos	Exibir mensagens de erro adequadas		
TC24	Usuários	Correção de bugs	-	Após testes	Validar correções	Fluxos funcionando corretament e		
TC25	Produtos	Listar produtos	-	-	Listar produtos via GET	Listagem bem-sucedida		
TC26	Produtos	Editar produto	-	Produto existente	Editar produto via PUT	Edição refletida no retorno da API		
TC27	Produtos	Excluir produto	-	Produto existente	Excluir produto via DELETE	Produto removido com sucesso		Confirmar via GET após exclusão
TC28	Produtos	Mensagens de erro	-	Testes negativos	Validar mensagens em operações inválidas	Mensagens adequadas em todos os fluxos		
TC29	Produtos	Correção de bugs	-	Após testes	Validar correções	Operações funcionando corretament e		
TC30	Relatórios	Acesso e visualização	-	Login realizado	Acessar página de	Relatórios carregados		

relatórios

corretament  
e

/ ServerRest - Planejamento de Testes

1 link do JIRA



				realizado	página Home	mento correto		
TC32	Navegação	Link: Cadastrar Usuário	-	Login realizado	Clicar no link de cadastro	Acesso à tela de cadastro		
TC33	Navegação	Link: Listar Usuários	-	Login realizado	Clicar no link de listagem	Acesso à tela de usuários		
TC34	Navegação	Link: Relatórios	-	Login realizado	Clicar no link de relatórios	Acesso aos dados gerenciais		
TC35	Navegação	Link: Logout	-	Login realizado	Clicar em sair	Logout com sucesso e redirecionamento ao login		
TC36	Sistema Geral	Correção de bugs e efeitos colaterais	-	Após execuções	Validar estabilidade geral após correções	Sistema estável e íntegro		
TC37	Bugs	Correção final de	-	Última rodada	Corrigir e revisar	Sistema livre de falhas		

### Teste Manual Exploratório – Documentação com Test & Feedback

Com o objetivo de padronizar a abordagem de testes manuais e garantir o rastreamento eficiente dos cenários executados, optou-se pelo uso da extensão **Test & Feedback** (Microsoft DevLabs). Essa ferramenta permite capturar evidências de execução, como capturas de tela, anotações e registros de eventos, promovendo maior rastreabilidade e reprodutibilidade dos testes.

#### Cenário de Teste CT01 – Fluxo de Cadastro

**Objetivo:** Validar o processo de cadastro de novos usuários, conforme os requisitos estabelecidos pela aplicação ServerRest.

#### Procedimento:

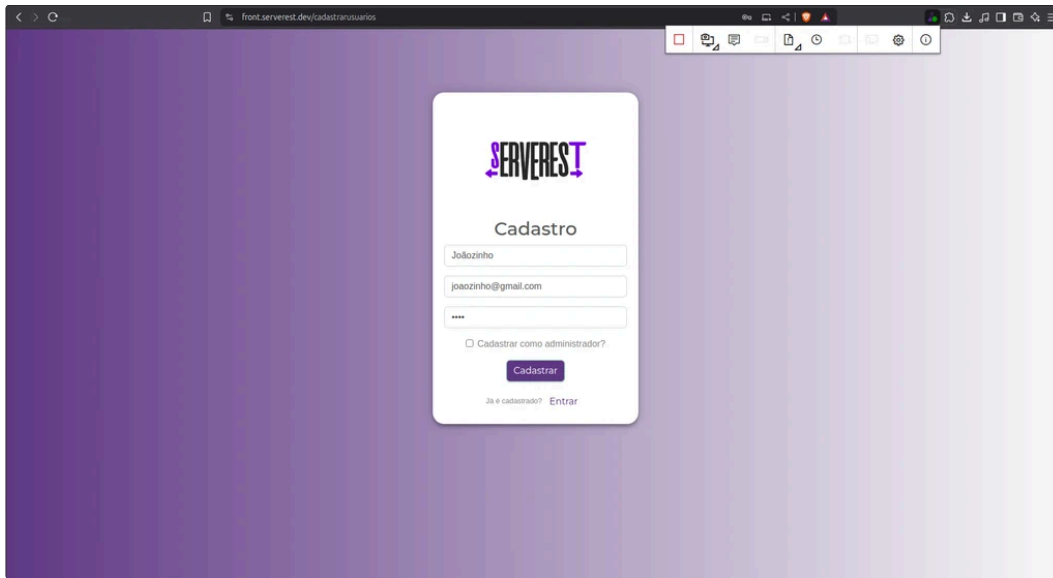
1. Acesse o navegador de sua preferência.
2. Digite a URL da aplicação no campo de endereço ( <https://serverest.dev> ).
3. Certifique-se de que a extensão **Test & Feedback** está instalada e habilitada.
4. Inicie a gravação da sessão clicando no ícone de **"Play"** na barra da extensão.
5. Navegue até a funcionalidade de **Cadastro** e preencha todos os campos obrigatórios conforme solicitado.
6. Submeta o formulário e observe o comportamento da aplicação.

7. Caso ocorra algum erro ou comportamento inesperado, utilize a funcionalidade de **captura de evidências** para registrar o evento.

/ ServerRest - Planejamento de Testes 1 link do JIRA



**Resultado Esperado:** O sistema deve permitir o cadastro com sucesso, apresentando uma mensagem de confirmação adequada, sem falhas ou comportamentos anômalos.



## Teste Automatizado – Ui.Vision (Visual)

### CT01 – Cadastro de Usuário

**Objetivo:** Automatizar o fluxo de cadastro de usuários utilizando a interface gráfica do Ui.Vision, sem codificação, apenas com ações visuais.

### Pré-requisitos

- Navegador Google Chrome ou Firefox instalado.
- Extensão **Ui.Vision RPA** instalada e fixada na barra de ferramentas.
- URL da aplicação em ambiente de testes: <https://serverest.dev> (exemplo).

### Tutorial Passo a Passo (Visual)

#### 1. Abrir o navegador e acessar a URL da aplicação.

→ Navegue até <https://serverest.dev>.

#### 2. Abrir a extensão Ui.Vision e criar um novo macro.

- Clique no ícone da extensão.
- Vá até a aba **Macro** e clique em **New Macro**.
- Nomeie como **CT01\_Cadastro**.

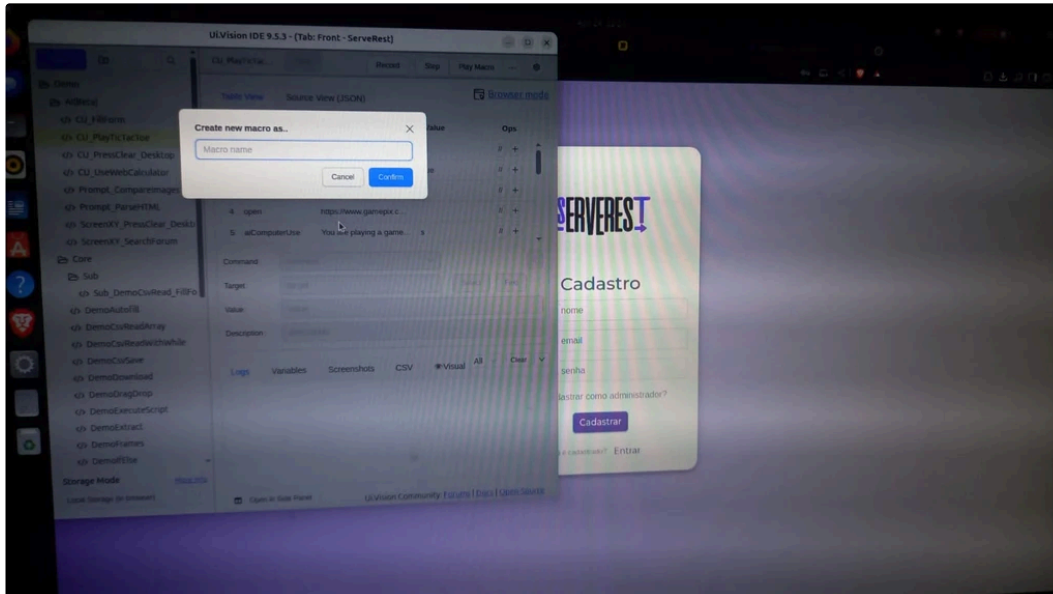


### 3. Iniciar a gravação.

/ ServerRest - Planejamento de Testes 1 link do JIRA



→ A partir daqui, **tudo que for clicado ou digitado será registrado automaticamente.**



### 4. Executar o fluxo normalmente como usuário:

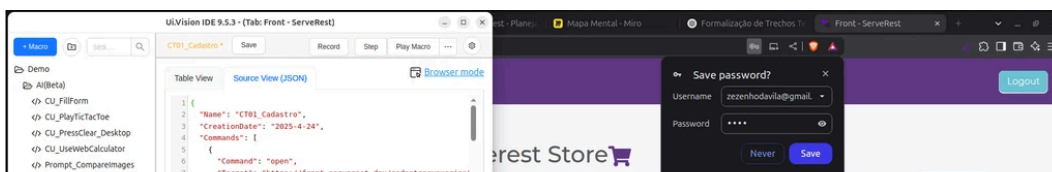
- ✓ Clique em **“Cadastrar”**
- ✓ Preencha o campo **nome**
- ✓ Preencha o campo **e-mail válido**
- ✓ Preencha a **senha com pelo menos 6 caracteres**
- ✓ Clique no botão **“Cadastrar”** ou equivalente.

### 5. Parar a gravação.

- Volte ao painel do Ui.Vision.
- Clique no botão **“Stop”** para finalizar o macro.

### 6. Executar o macro para testar a automação.

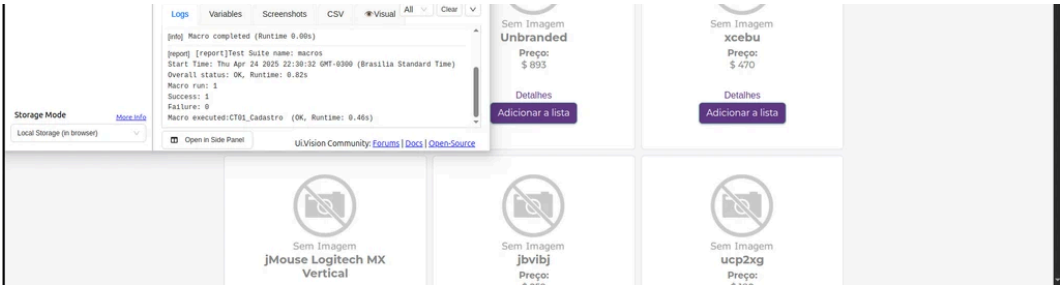
- Com o macro **CT01\_Cadastro** selecionado, clique em **“Play Macro”** (ícone de triângulo).
- Observe a execução automática replicando o cadastro.





/ ServerRest - Planejamento de Testes

1 link do JIRA



## Análise de Riscos no Plano de Testes

Durante a elaboração do plano de testes para o projeto ServerRest, foi realizada uma análise detalhada dos riscos que poderiam comprometer a eficácia, a rastreabilidade ou a continuidade dos testes. Esta análise considerou não apenas os aspectos técnicos do sistema, mas também as ferramentas e abordagens adotadas pela squad para realizar os testes manuais e automatizados.

A estratégia de testes adotada inclui:

- Testes manuais exploratórios utilizando a extensão **Test & Feedback**, com foco na heurística **CHIQUE**;
- Testes automatizados no frontend com a ferramenta **Ui.Vision**, utilizando interface visual e gravação de macros;
- Registro e documentação no **Confluence**, além da criação de **Épicos e Casos de Teste no Jira**;
- Planejamento de execução contínua e reuso dos fluxos via integração CI/CD (rodando testes diariamente);
- Análise de performance e stress com ferramenta dedicada (ex: Apache JMeter).

Com base em todo esse cenário, foram mapeados riscos relacionados ao **fator humano**, à **fragilidade das ferramentas visuais**, à **fragmentação da documentação**, e à **ausência de métricas claras**. A tabela a seguir consolida cada risco identificado, sua probabilidade de ocorrência, o impacto que pode gerar, e a estratégia de mitigação proposta para neutralizar ou reduzir seus efeitos no projeto.

Risco	Descrição	Probabilidade	Impacto	Estratégia de Mitigação
1. Erros humanos em testes manuais	Mesmo com uso do Test & Feedback, os testers podem esquecer de registrar bugs ou seguir o passo correto.	Alta	Média	Padronizar tutoriais e boas práticas; criar checklist visual e vídeo para cada fluxo




2. <b>Aprendizado</b>	Como é uma ferramenta visual para	Média	Alta	Treinar a equipe com
/ ServerRest - Planejamento de Testes 1 link do JIRA				
	Executar sem entender lógica de automação.			Simular erros comuns e como corrigi-los
3. <b>Falta de versionamento nos testes automatizados (macros Ui.Vision)</b>	Pode haver sobrescrita de arquivos ou perda de fluxos prontos.	Alta	Alta	Criar repositório centralizado no Git ou no Jira (via anexo) com histórico por versão
4. <b>Inconsistência de dados nos testes automatizados</b>	Dados reais no ambiente podem gerar falhas.	Alta	Alta	Automatizar script para popular a base com dados padrão antes da execução
5. <b>UI instável quebrando automações</b>	Mudanças visuais no sistema invalidam seletores utilizados no Ui.Vision.	Alta	Alta	Usar atributos únicos e fixos como IDs; revisar layout a cada sprint com time de Dev
6. <b>Subnotificação de erros nos testes</b>	Bugs não reportados formalmente ou não linkados ao Jira.	Média	Alta	Estabelecer rotina de revisão semanal dos registros da Test & Feedback; criar integração com Jira
7. <b>Dificuldade em manter cobertura de testes completa</b>	Testes esquecem cenários de erro, borda ou stress.	Alta	Alta	Criar plano de cobertura com CHIQUE + tabela de rastreabilidade por fluxo
8. <b>Dificuldade para rodar testes periodicamente</b>	Sem CI/CD, automações ficam esquecidas.	Alta	Média	Usar agendamento diário via Runner local ou GitHub Actions (para JSONs do Ui.Vision)
9. <b>Ferramentas diferentes dificultando integração (Test &amp; Feedback, Ui.Vision, JMeter)</b>	Fragmentação da documentação e execução dos testes.	Alta	Média	Centralizar registros no Confluence, com tabelas de rastreabilidade entre ferramentas


10. Falta de	Equipe não tem	Média	Alta	Gerar painel no
--------------	----------------	-------	------	-----------------


/ ServerRest - Planejamento de Testes 1 link do JIRA


+ Adicionar categoria


Conteúdo relacionado ⓘ


**Início Rápido em Teste e QA**  
douglas paulo Cortes


 Mais parecido com essas


**Template- Anotações de reunião**  
Desenvolvimento de software


 Mais parecido com essas


**Introdução ao Confluence from Jira**  
douglas paulo Cortes


 Mais parecido com essas


**Visão geral**  
douglas paulo Cortes

 Mais parecido com essas

**Relatório de Melhorias - User Story "Meus Pedidos"**  
Anna Santoro

 Mais parecido com essas

**Loja Nexus Home**  
Loja Nexus

 Mais parecido com essas

    Seja o primeiro a adicionar uma reação