

Testes automatizados - Busca Rede

Um tutorial sobre como configurar e executar um teste

› Configuração de Ambiente

1. Baixar o Ruby+Devkit 3.2.3-1 (x64) em: <https://rubyinstaller.org/downloads/>
2. Após a instalação, execute o script abaixo no prompt de comando (cmd). Em caso de sucesso, a versão instalada será exibida



```
ruby -v
```

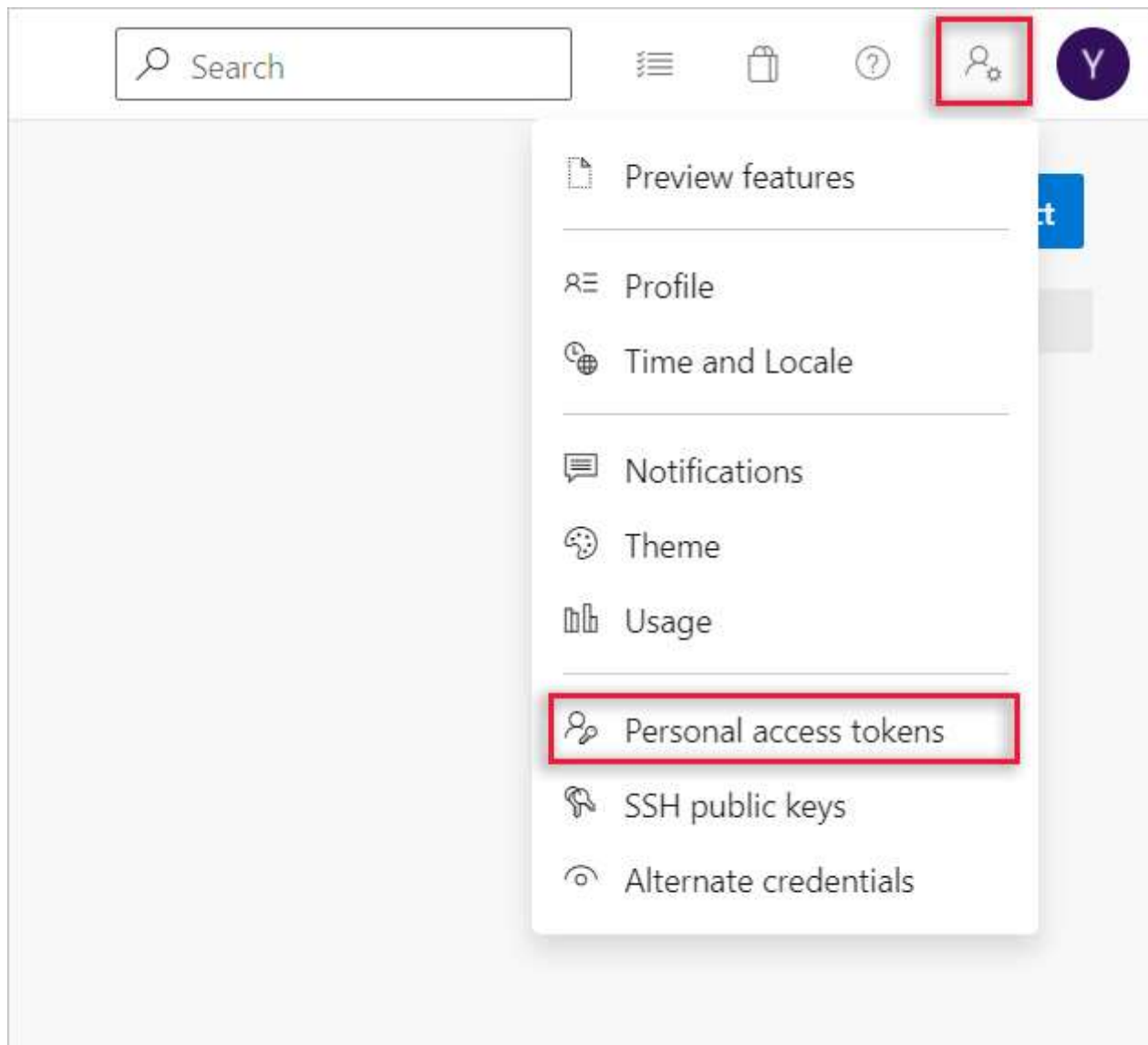


› Baixando do repositório para a máquina local

› Gerando chave de autenticação


1. Selecione o menu "Personal access token"





2. Selezione "New Token"



 Azure DevOps

User settings

General

About

Time and Locale

Notifications

Theme

Personal Access Tokens

These can be used instead of a password for

+

New Token

Token name

Git: https://mseng.visualstudio.com/
Code (Read & write); Packaging (Read)

3. Selecione a expiração pra 90 dias e "Read, write & manage" como permissão



Create a new personal access token



Name

token-exemplo

Organization

ub-Institucional



Expiration (UTC)

90 days



1/28/2024



Scopes

Authorize the scope of access associated with this token

Scopes ☐ Full access

☒ Custom defined

Work Items

Work items, queries, backlogs, plans, and metadata

☐ Read ☐ Read & write ☐ Read, write, & manage

Code

Source code, repositories, pull requests, and notifications

☒ Read ☒ Read & write ☒ Read, write, & manage ☐ Full ☐ Status

Build

Artifacts, definitions, requests, queue a build, and update build properties

Artifacts, definitions, requests, queue a build, and update build properties

☐ Read ☐ Read & execute

Release

Read, update, and delete releases, release pipelines, and stages

Show all scopes (29 more)

Create

Cancel

4. Copie o access token gerado e salve (será usado nos próximos passos), ele não será exibido de novo. Se perder, é possível criar 

Success!



You have successfully added a new personal access token. Copy the token now!

Jamall Hartnett's Token

Copy to clipboard

b6eme6inskhw4fze5vpyv



Warning - Make sure you copy the above token now. We don't store it and you will not be able to see it again.

Close

› Clonando projeto

1. No repositório, busque o o botão "Clone"



The screenshot shows the Azure DevOps web interface. On the left is a sidebar with navigation links: Overview, Boards, Repos, Files (selected), Commits, Pushes, and Branches. The main area is divided into two panes. The left pane shows the repository structure: a folder 'FiberTest1' and files '.gitattributes', '.gitignore', and 'FiberTests.sln'. The right pane, titled 'Files', shows a table of repository contents. The 'Clone' button is highlighted with a red rectangle. Below the table, there is a section for 'Contents' and 'History'.

Name ↑	Last change	Commits
FiberTest1	Friday	32e39466 Add project files....
.gitattributes	Friday	32e39466 Add project files....
.gitignore	Friday	32e39466 Add project files....
FiberTests.sln	Friday	32e39466 Add project files....

2. Copie o endereço e salve em algum bloco de notas



Clone Repository



Command line

HTTPS


SSH

`https://fiber-teams@dev.azure.com/fi`



Generate Git Credentials

IDE

 Clone in VS Code



Having problems authenticating in Git? Be sure to get the latest version:

- ① [Git for Windows](#) or our plugins for [IntelliJ](#), [Eclipse](#), [Android Studio](#) or: [Windows command line](#).

3. Agora, na URL copiada do git, troque o nome da organização pelo Access Token gerado



Exemplo:

```
Access token: 3ci6vwtj57tsb5zkfwsanxmbx3zgz3ahuaqe2jqhsgzs7ro52
```

```
URL git: https://ub-Institucional@dev.azure.com/ub-Institucional/SQUADS/_git/RNP
```

```
https://<alterar_organização_para_Access_token>@dev.azure.com/ub-Institucional/SQUADS/_git/RNP
```

```
https://3ci6vwtj57tsb5zkfwsanxmbx3zgz3ahuaqe2jqhsgzs7ro52@dev.azure.com/{Organization}/{Project}/_git/RNP
```

4. Adicione o comando "git clone" e faça o clone do projeto para sua máquina (troque pela sua chave)

```
git clone https://3ci6vwtj57tsb5zkfwsanxmbx3zgz3ahuaqe2jqhsgzs7ro52@dev.azure.com/ub-Institucional/SQUADS/_git/RNP
```

5. Acesse a pasta do seu projeto

```
cd RNP
```

› Criando uma branch local

6. Defina um nome para sua branch, pode ser por exemplo user story + seu nome



```
git checkout -b 123456_ciclano
```

› Subindo código para o Azure DevOps

Tudo certo com o seu código? Hora de subir para o azure.



7. Para adicionar TODAS as suas alterações, digite:

```
git add .
```

8. Agora, monte o seu commit e lembre-se de escrever uma mensagem coerente com a sua alterar_organização_para_Access_token

```
git commit -m "Inclusão dos testes automatizados de Login"
```

9. Subindo tudo para a sua branch no azure

```
git push origin 123456_ciclano
```

› Execução

10. Para executar tudo, basta escrever o comando "cucumber" na linha de comando, ou selecione algum teste em específico como no ex 

```
cucumber -t "@pesquisaSimples_validando_resultados"
```

› Documentação do Framework de Testes em Ruby + Cucumber

› Índice

- [Introdução](#)
- [O que é um Framework de Testes](#)
- [O que é Ruby](#)

- [O que é Cucumber](#)
- [O que é Gherkin](#)
- [Estrutura do Projeto](#)
 - [Arquivos e Pastas](#)
- [Criando um Novo Cenário](#)
- [Executando Cenários de Teste](#)
- [Dependências do Projeto](#)
- [Referências Bibliográficas](#)

› Introdução

Este documento fornece uma visão geral do framework de testes automatizados desenvolvido em Ruby e Cucumber, detalhando sua estrutura, componentes e linguagens utilizadas.

› O que é um Framework de Testes

Um framework de testes é um conjunto de diretrizes ou regras utilizadas para criar e projetar casos de testes. É uma estrutura conceitual que facilita a criação, manutenção e execução de testes automatizados, promovendo reusabilidade, extensibilidade e eficiência.

› O que é Ruby

Ruby é uma linguagem de programação dinâmica, open source, focada em simplicidade e produtividade. Possui uma sintaxe elegante, natural para ler e fácil de escrever. Ruby é popularmente utilizado em desenvolvimento web, scripts, e testes automatizados.

› O que é Cucumber

Cucumber é uma ferramenta de teste que suporta Behavior Driven Development (BDD). Permite a escrita de testes automatizados que são facilmente entendidos por não-programadores, pois usa a linguagem Gherkin, que descreve o comportamento do software em texto plano.

O que é Gherkin

Gherkin é uma linguagem que permite a descrição de funcionalidades de software em um formato legível e compreensível. Ela pode ser:

- **Imperativa:** Detalha passo a passo como a interação deve ocorrer.
- **Declarativa:** Foca no resultado desejado, sem detalhar os passos específicos.

Exemplos de Gherkin Imperativo e Declarativo

Gherkin Imperativo

O Gherkin imperativo detalha cada passo da interação do usuário com a interface:

Funcionalidade: Login do usuário

Cenário: Login com sucesso

```
Dado que eu estou na página de login
Quando eu preencho o campo de usuário com "usuario_exemplo"
E eu preencho o campo de senha com "senha_exemplo"
E eu clico no botão de login
Então eu devo ser redirecionado para a página inicial do usuário
```



Gherkin Declarativo

O Gherkin imperativo detalha cada passo da interação do usuário com a interface: Funcionalidade: Login do usuário

Cenário: Login com sucesso



Dado que eu estou na página de login
Quando eu realizo o login com credenciais válidas
Então eu devo ser redirecionado para a página inicial do usuário

› Estrutura do Projeto

› Arquivos e Pastas

- Arquivos:

- `.gitignore` : Define os arquivos e pastas ignorados pelo Git.
- `.rspec` : Configurações para o RSpec.
- `azure-pipelines.yml` : Configurações para CI/CD com Azure Pipelines.
- `cucumber.yml` : Configurações para o Cucumber.
- `Gemfile` : Lista as dependências do Ruby.

- Pastas:

- `config` : Contém arquivos de configuração.
- `documentação` : Armazena este e outros documentos relacionados ao projeto.
- `features` : Pasta principal para os testes.
 - `logs` : Armazena logs gerados durante os testes.
 - `specs` : Contém os arquivos `.feature` , escritos em Gherkin.
 - `step_definitions` : Contém os passos que implementam as definições de Gherkin.
 - `support` : Scripts de suporte.
 - `pages` : Define as páginas para os testes de interface.
 - `env.rb` : Configurações de ambiente.
 - `hooks.rb` : Define hooks para os testes.

› Criando um Novo Cenário

Para criar um novo cenário de teste:

1. **Arquivo .feature:** Comece criando um arquivo `.feature` dentro da pasta `specs`. Este arquivo deve descrever o comportamento do software em linguagem Gherkin.
2. **Step Definitions:** Após definir o cenário em Gherkin, crie os passos correspondentes na pasta `step_definitions`. Estes passos devem implementar a lógica descrita no arquivo `.feature`.
3. **Mapeamento de Elementos:** Se necessário, mapeie elementos da interface do usuário na pasta `pages`, para serem utilizados nos steps.

› Mapeamento de Elementos e Mapeadores em Ruby

O mapeamento de elementos é uma parte crucial dos testes de interface do usuário. Ele permite que os scripts de teste identifiquem e interajam com os elementos na página web. Aqui estão alguns dos mapeadores e métodos comuns usados em Ruby:

- **find:** Usado para localizar um elemento na página. Por exemplo, `find('#id_do_elemento')`.
- **click:** Utilizado para clicar em um elemento. Por exemplo, `find('#id_do_botao').click`.
- **set:** Empregado para inserir texto em campos de entrada. Por exemplo, `find('#id_do_campo').set('Texto para inserir')`.
- **text:** Usado para obter o texto de um elemento. Por exemplo, `find('#id_do_elemento').text`.

› Executando Cenários de Teste

- **Individualmente:** Para executar um cenário específico, use o comando `cucumber features/nome_do_arquivo.feature`.
- **Em Grupo:** Para executar um grupo de cenários, você pode organizar seus arquivos `.feature` com tags e executá-los usando `cucumber --tags @sua_tag`.

› Dependências do Projeto

```
gem 'rspec'
```

```
gem 'cucumber'
```

```
gem 'capybara'

gem 'selenium-webdriver'

gem 'pry'

gem 'site_prism'

gem "rake", "~> 13.0"

gem 'parallel_tests'

gem "ruby-lsp", "~> 0.3.8", :group => :development

gem 'rack-mini-profiler', '~> 3.0'

gem 'httparty'

gem 'report_builder'
```

Referências Bibliográficas

- [Ruby Documentation](#)
- [Cucumber Official Website](#)
- [Gherkin Reference](#)
- [BDD Guide by Cucumber](#)
- [Gherkin Imperativo vs Declarativo](#)