Linguagem/Framework escolhido: Ruby/Cucumber/Capybara

Escolhi a linguagem Ruby por ser uma linguagem de fácil entendimento e muito flexível devido não precisar de um compilador para ser construído e por ser orientado a objetos o que facilita a manipulação da estrutura dos objetos.
O cucumber permite descrever as regras de negócio com uma linguagem natural, o que facilita o entendimento de qualquer um da equipe.
E for fim o Capybara pois me permite executar ações na aplicação que o próprio usuário final faria.

Juntos, consigo automatizar uma aplicação com mais agilidade e com maior agregação de valor, uma vez que a documentação é viva.

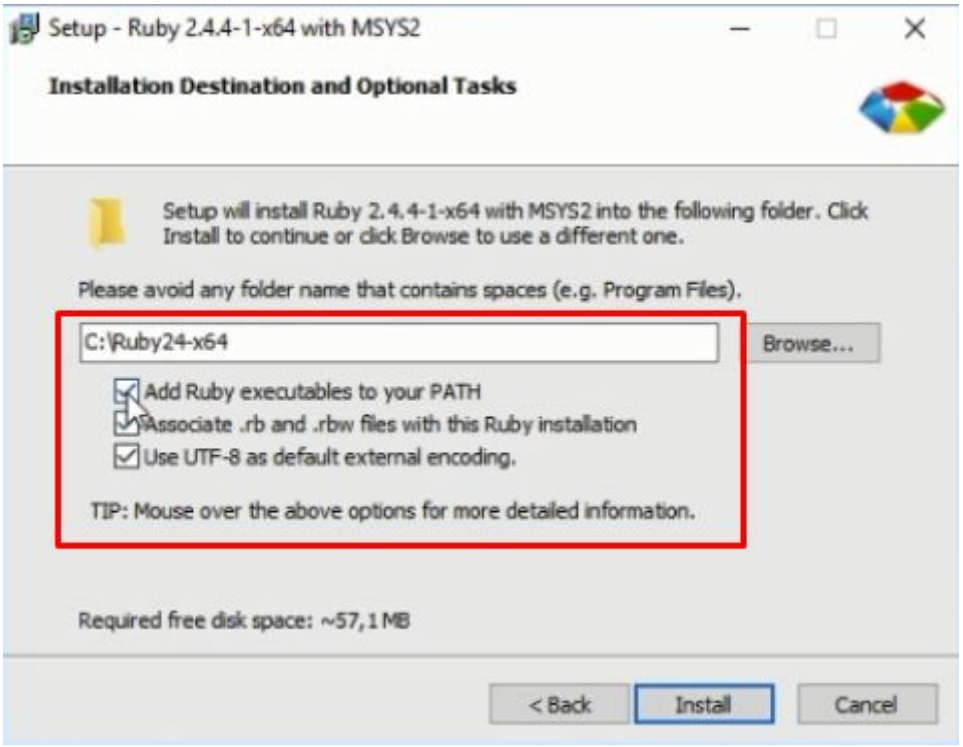
Estrutura do Projeto

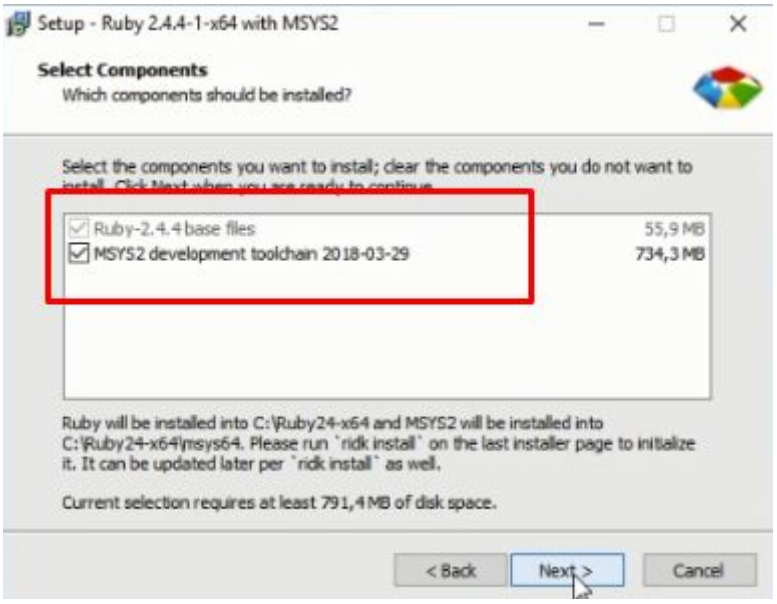
Eu escolhi criar apenas uma feature para descrever os cenários de testes, pois tinham o mesmo objetivo.
Separei as classes 'carrinho' e 'pagamento' mesmo sendo na mesma página, porque possuem objetivos distintos. A PageObject 'carrinho' tem como objetivo inserir um produto no carrinho, enquanto a PageObject 'pagamento' tem como objetivo inserir e validar as informações de pagamento.

Para Executar os testes:

1- Instalar o Ruby + DevKit através do link *https://rubyinstaller.org/downloads/*

Marcar todas as opções conforme imagens abaixo:





Para verificar se foi instalado, no Prompt de Comando digite **ruby -v**

Eu estou utilizando a versão 2.4.4p296

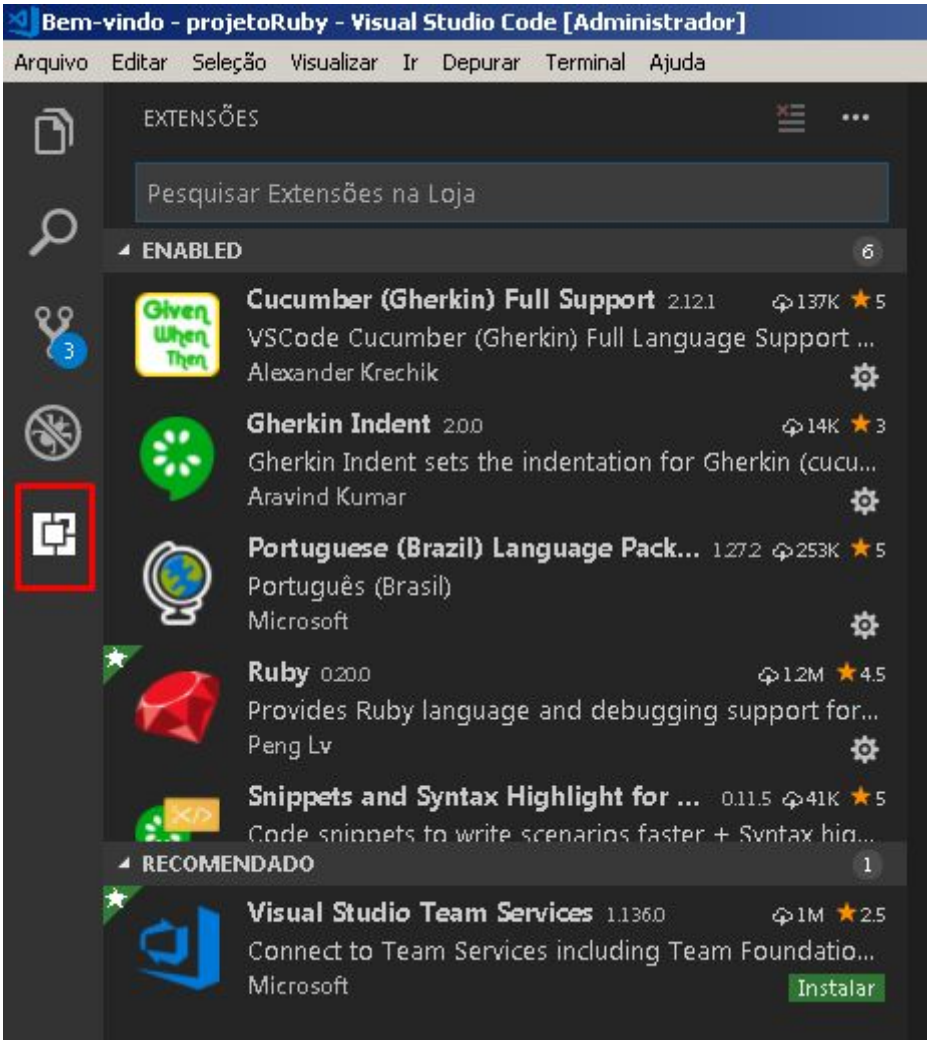
2- No Prompt de Comando execute o comando **gem install bundler**

3- Instalar o VisualStudio através do link *https://code.visualstudio.com/download*
Eu estou utilizando a versão 1.27.2

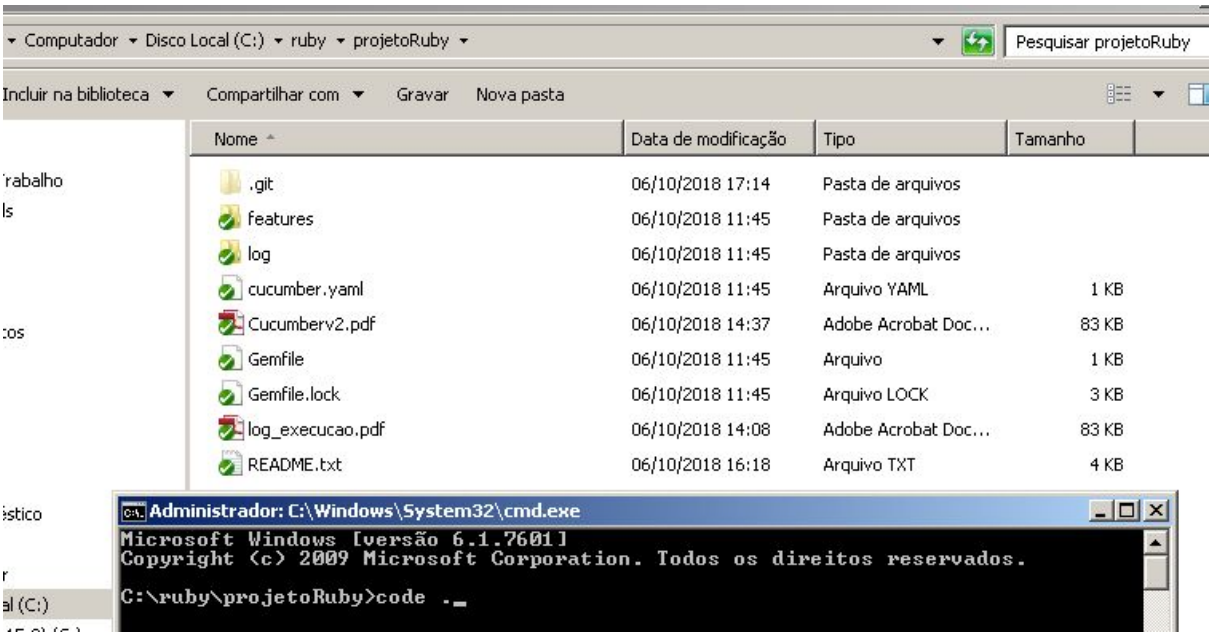
4- Instalar os plugins:

vscode-icons
Snippets and Syntax Highlight for Gherkin (Cucumber)
Gherkin Indent

Através do VisualStudio > Extensões



5- Para abrir o projeto vocês podem dentro da pasta do projeto abrir um prompt de comando e executar o comando **code .**



6- Para executar os testes vocês devem abrir um prompt de comando dentro da pasta raiz do projeto e executar o comando **cucumber**.
Eu deixei uma anotação em cada cenário, exemplo, @1, @2, @3, etc.
Caso queiram executar um teste em específico podem executar o comando **cucumber -t @1**, por exemplo.

7- Para visualizar o relatório de execução dos testes abra o arquivo **features.html** que fica dentro da pasta **log**.

Maiores Dificuldades:

1 - Eu investi muito tempo tentando utilizar os sites: Americanas, Submarino, CasasBahia, Amazon, entre outros.
Estes sites possuem validação de Captcha e eu não sabia que esta validação é justamente para impedir acessos automatizados na página,então investi muito tempo tentando encontrar uma forma de validar isso.

Lição Aprendida: sempre buscar entender uma funcionalidade. Uma pergunta que eu poderia ter feito logo no início era: para que serve o captcha :D

Acabei utilizando o Groupon que não possui validação de Captcha. Falha de Segurança? xD

2 - Tive problemas com conflito de dependências de Gem
Investi um tempinho procurando soluções, pois este problema nunca havia visto. Tentei algumas coisas que não deram certo até que executei os comandos abaixo e resolveu o problema:

'gem uninstall sinatra'
'gem uninstall rack'

Em outras pesquisas descobri que poderia ter executado o teste utilizando somente as versões de gem do meu projeto.

bundle exec cucumber

3 - Inicialmente utilizei muito XPath para encontrar alguns elementos do site, porque não estava conseguindo implementar a solução e tomei a decisão de fazer o teste funcionar e após refatorar.

Lições Aprendidas: eu não sabia que quando uma classe possui texto com espaço, exemplo, class="qtd-buttons qtd-add qtd-plus ", significa que são nomes de classes diferentes, e por isso não conseguia encontrar os elementos. Depois de muita pesquisa descobri que quando me deparo com essa situação eu devo escolher a classe única como parâmetro. Neste exemplo eu utilizei o qtd-plus.

4 - Não consegui implementar o click no botão pesquisar, então após pesquisas encontrei a solução abaixo que resolveu.

find(:css, 'input[name="q"]').native.send_keys(:return)

Não busquei outra solução, porque esta atendeu ao que eu precisava e aprendi um novo comando.

5 - Nesta validação:

```
Entao("eu deverei possuir dois produtos no meu carrinho") do
  qtd = find('input[name="qtd-cupom"]').value
  expect(qtd).to eql '2'
end
```

Investi um tempinho também para tentar guardar o conteúdo do campo numa variável até que consegui utilizando o value.
Antes havia tentando coisas como:
find('exemplo', :text = "exemplo")

```
realiza_compra = RealizaCompra.new
expect(realiza_compra.qtd_cupom.text).to eql '2'
```

Lição Aprendida: esta foi mais pesquisa mesmo. Não tem como fugir muito.

6 - Meus testes começaram a quebrar por causa das promoções que eu usei e que expiraram.

Lição Aprendida: neste cenário, onde a aplicação está em produção, sinceramente não sei como poderia resolver. Mas caso eu estivesse utilizando um ambiente controlado, faria um script de inserção da promoção para rodar antes dos testes e assim não ter esse problema.