

Formação em Testes Manuais e Automatizados

Uniesp-PB

indra



29/09/2021

Automação de Testes

1

Por que automatizar os testes?



- Reduzir a chance de encontrar erros nos testes
- Reduzir o esforço gasto com os testes manuais
- Execução de testes nos mais diversos cenários e dispositivos
- Redução no tempo de execução dos testes. Ex: Testes de Regressão

Quando usar automação no processo de testes?



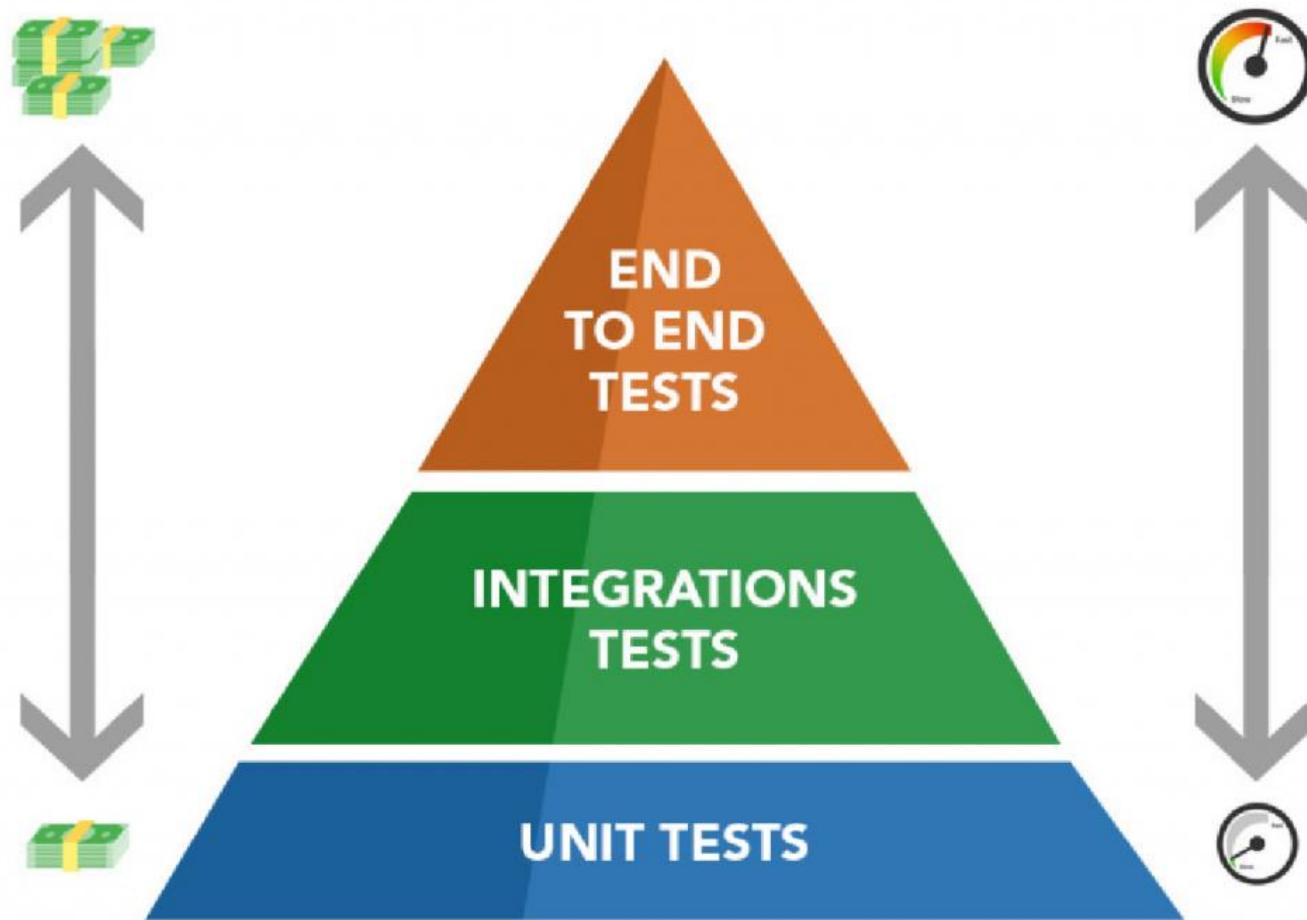
Podemos começar respondendo as seguintes perguntas:

- Teremos redução de custos e esforço?
- Estaremos suprindo as demandas do cliente?
- Manteremos a qualidade nos testes manuais?
- Conseguiremos integrar nossos testes automatizados no processo de desenvolvimento?

Níveis de Testes

	Testes Unitários	Integração	Sistema	Aceitação
Porque?	Garantir que o código foi desenvolvido corretamente	Garantir o funcionamento do fluxo entre componentes do sistema	Garantir o funcionamento do sistema após tudo estar integrado	Garantir que as expectativas e requisitos do cliente e do usuário final foram cumpridos
Quem?	Desenvolvedores/Arquitetos	Desenvolvedores/Arquitetos E QA	Tester Manual / Analista de Negócios / Product Owner	Desenvolvedor / Tester Manual / PO / Usuários finais do produto
O que?	Novos códigos + refatorações de códigos legados	Novo web service, componentes, controllers, etc.	Cenários de testes, fluxos de usuário e jornadas típicas, performance e testes de segurança	Verificar os critérios de aceitação das histórias e verificar os requisitos
Quando?	Assim que um novo código é escrito	Assim que um novo componente é adicionado	Quando o produto estiver completo / desenvolvido	Quando o software já esta pronto pra ir para produção
Onde?	Ambiente de desenvolvimento local + Integração Continua	Ambiente de desenvolvimento local + Integração Continua	Ambiente de testes e homologação	CI / Ambiente de Testes e homologação
Como? (Ferramentas e Métodos)	Automatizado, JUnit, TestNG, PHPUnit, Mocha, etc.	Automatizado, Soap, UI, Rest Assured, Postman, CYPRESS	Automatizados (Cypress, Selenium, Protractor, etc) e testes manuais / exploratórios	Automatizados (Cypress, Selenium, Protractor, etc) e Testes Manuais

Pirâmide de Testes automatizados



Let's go!

2

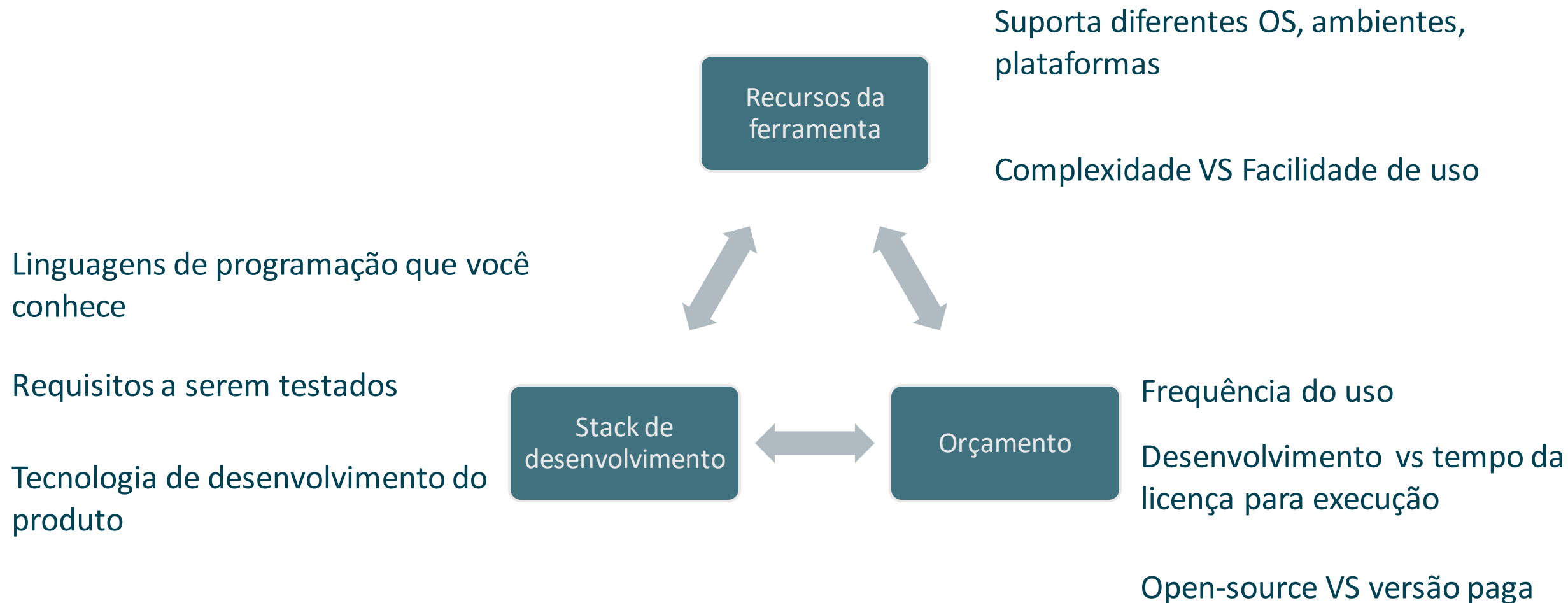
Como começar....



Casos de testes perfeitos para automatizar

- ✓ Execução manual demorada
- ✓ Possui um grande conjunto de dados envolvidos
- ✓ Irá prover cobertura de componentes de software estáveis
- ✓ Não requer criatividade na execução
- ✓ Requer que seja verificado em diversos e diferentes sistemas (browsers, OS, hardware, etc)
- ✓ Resultados limpos passou/falhou
- ✓ Altamente Repetitivo

Escolhendo as ferramentas de testes



Tecnologias/Ferramentas



LINGUAGENS CONSIDERADAS



Integração/Entrega Contínua



Jenkins



Geração de Relatórios



Lógica de Programação

Condicionais

Operadores

Variáveis

Funções

Sintaxe

Classes

Objetos



- **Principais características:**

- Utiliza a linguagem JavaScript;
- Baixa curva de aprendizado;
- Tem a estabilidade de execução como principal pilar, diminuindo falsos-positivos;
- Correções de problemas encontrados são rapidamente corrigidos pela equipe de desenvolvimento;
- Documentação altamente detalhada e de fácil compreensão;
- Fácil instalação;

- **Principais recursos:**

- Time-Travel: demonstra todos os passos efetuados de forma visual;
- Gravação de vídeos;
- Interface para acompanhamento de testes;

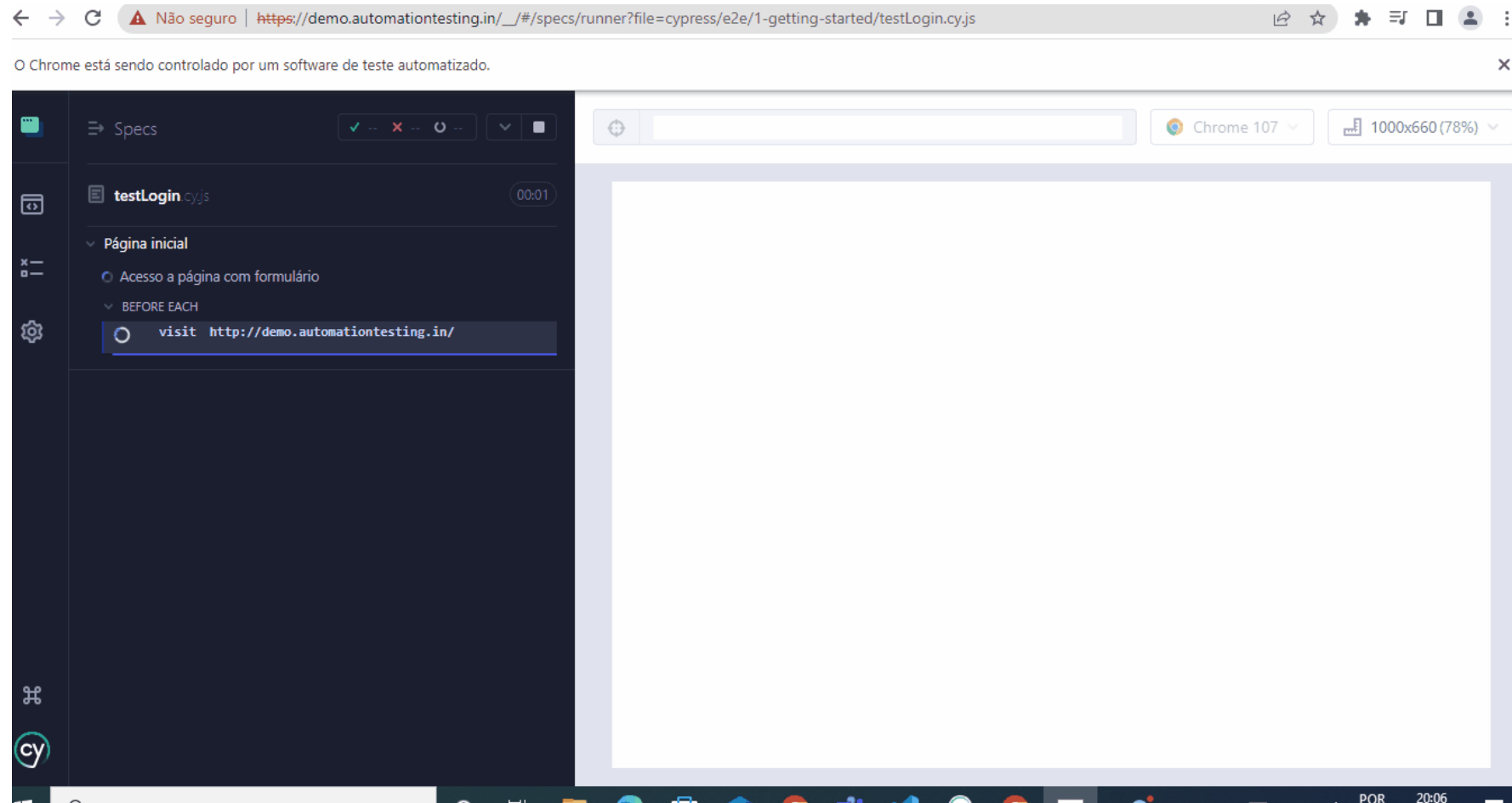


- **Observações:**

- Não permite a execução em dois ou mais superdomínios (Ex.: Abrir o site 1, e ao clicar um botão leva a um site 2) (Desenvolvimento em andamento);



The Test Runner (Desktop)



Setup

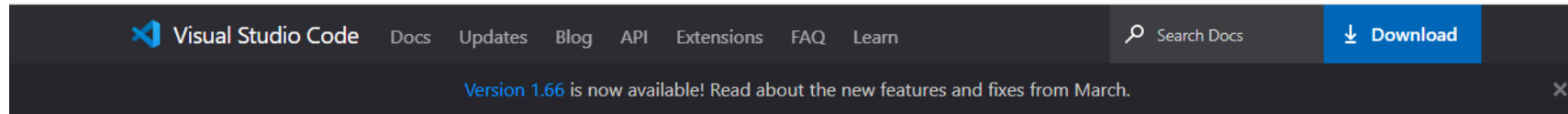
3

Preparando o ambiente de desenvolvimento

- ✓ Instalar o Chrome browser
- ✓ Instalar o Node js
- ✓ Instalar a IDE (Visual Studio Code)

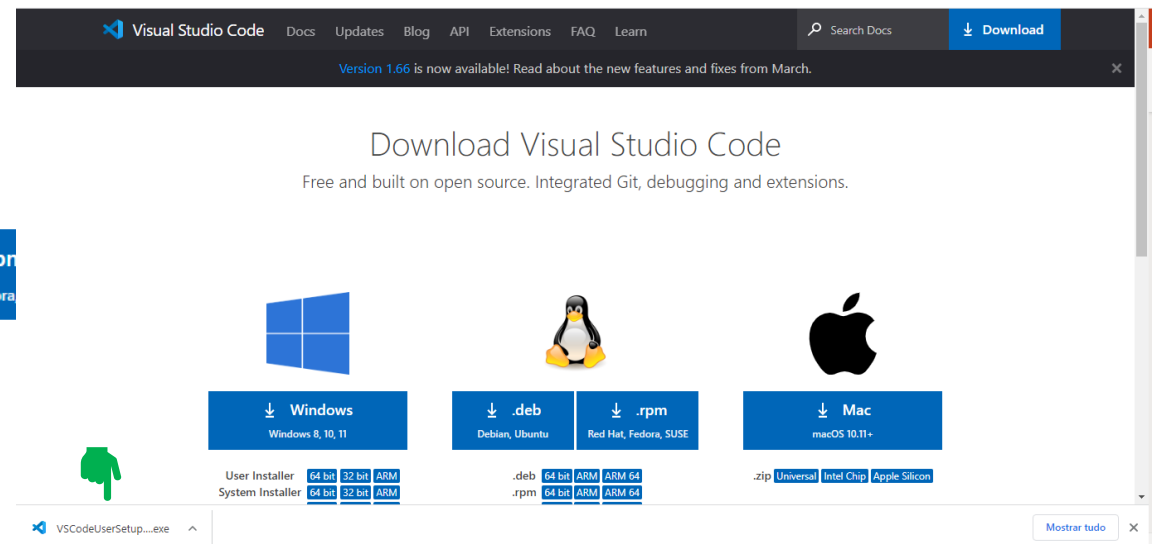
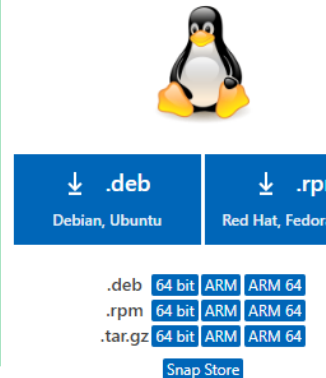
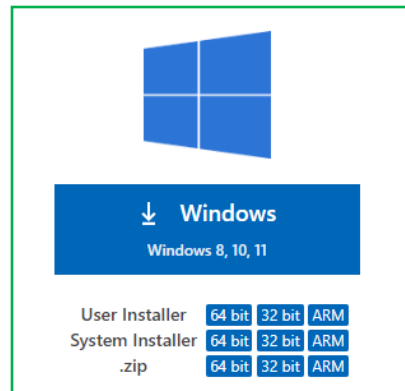


Download e Instalação do VsCode



Download Visual Studio Code

Free and built on open source. Integrated Git, debugging and extensions.



<https://code.visualstudio.com/download>

Download NodeJs



nodejs.org/pt-br/download/

node


INÍCIO | SOBRE | DOWNLOAD | DOCUMENTAÇÃO | PARTICIPE | SEGURANÇA | NOVIDADES | CERTIFICATION


Downloads


Versão LTS Mais Recente: 16.15.0 (includes npm 8.5.5)

Baixe o código fonte do Node.js ou um instalador pré-compilado para o seu sistema, e comece a desenvolver hoje.

LTS
Recomendado Para a Maioria dos Usuários


Instalador Windows
node-v16.15.0-x64.msi


Instalador macOS
node-v16.15.0.pkg


Código fonte
node-v16.15.0.tar.gz

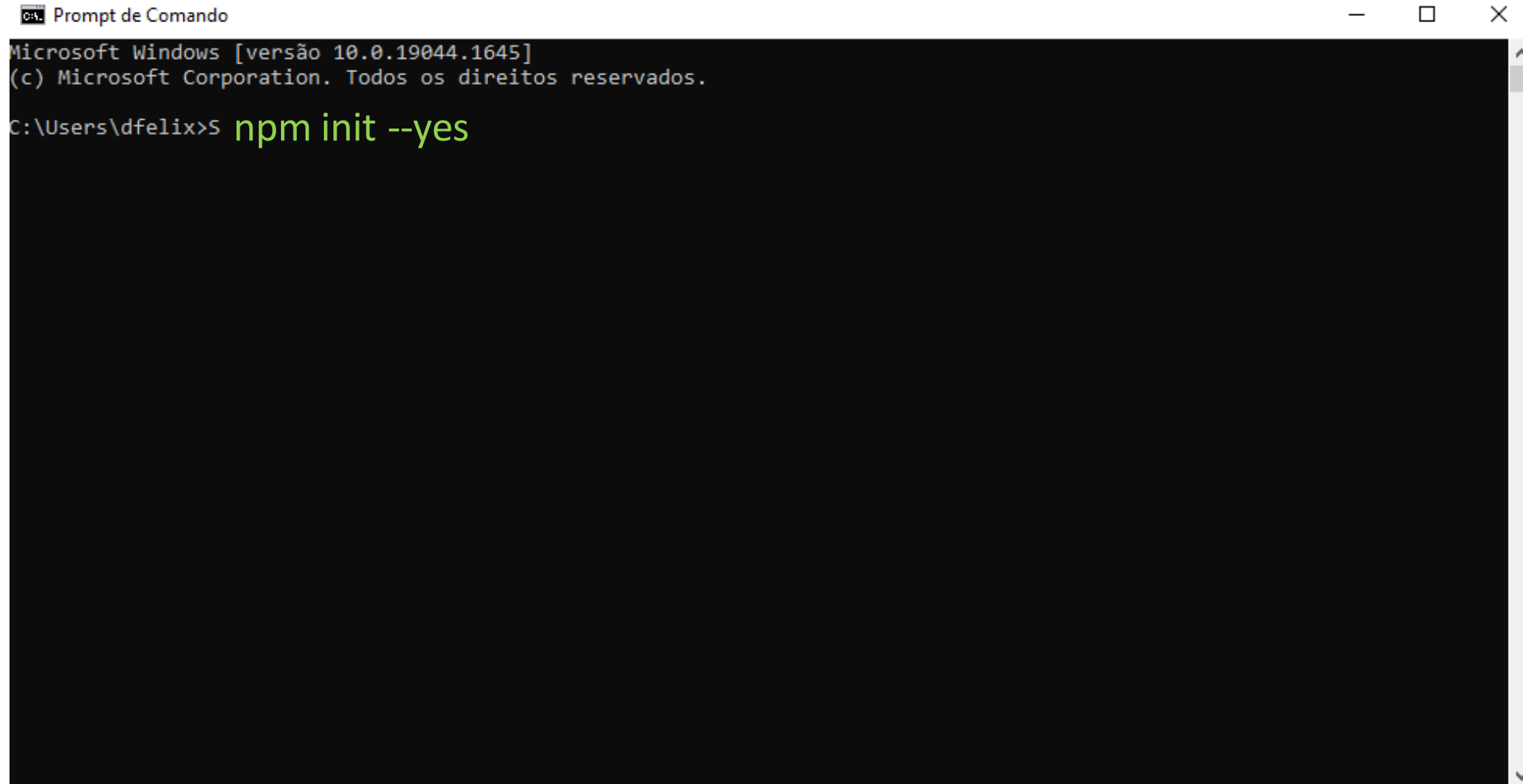
32-bit	64-bit
32-bit	64-bit
64-bit / ARM64	
64-bit	ARM64
64-bit	
ARMv7	ARMv8
node-v16.15.0.tar.gz	

<https://nodejs.org/dist/v16.15.0/node-v16.15.0-x64.msi>

ataformas adicionais

<https://nodejs.org/pt-br/download/>

Iniciando o projeto Node



```
Microsoft Windows [versão 10.0.19044.1645]  
(c) Microsoft Corporation. Todos os direitos reservados.  
C:\Users\dfelix>S npm init --yes
```



Instalando a dependência do Cypress



```
cmd Prompt de Comando
Microsoft Windows [versão 10.0.19044.1645]
(c) Microsoft Corporation. Todos os direitos reservados.
C:\Users\dfelix>S  npm install cypress --save-dev
```

Execução

4

Executando Cypress em modo *headed*

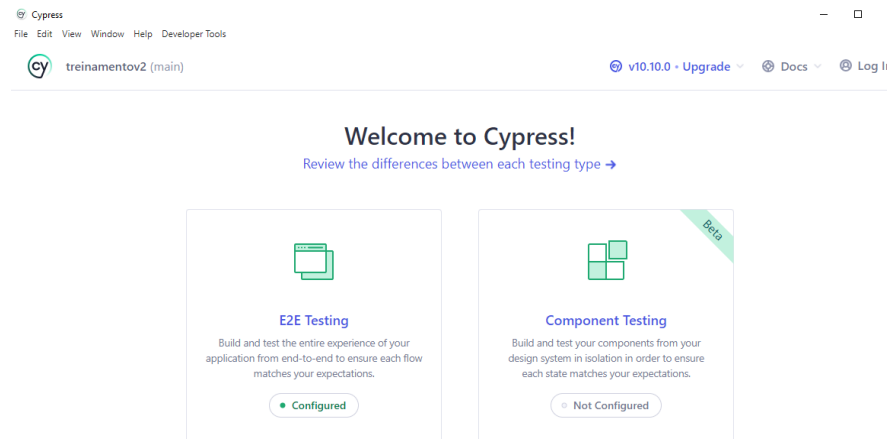


```
cmd Prompt de Comando
Microsoft Windows [versão 10.0.19044.1645]
(c) Microsoft Corporation. Todos os direitos reservados.

C:\Users\dfelix> npx cypress open
```

Modo *Headed*:

- O modo de *headed* é definido pela presença de interface do usuário (UI). No modo *headed*, um único aplicativo de interface do usuário será lançado.



Utilizando comando customizado



Modo *Headed*:

- Acessando o package.json do projeto, podemos customizar o comando de abrir o cypress dentro da propriedade "Scripts"

```
"scripts": {  
  "cy:browser": "cypress open --browser chrome --e2e",  
  "cy:open": "cypress open"  
},
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

```
PS C:\Users\dfelix\Desktop\Repositorios\treinamentov2> npm run cy:browser
```

Executando Cypress em modo *headless*



```
cmd Prompt de Comando
Microsoft Windows [versão 10.0.19044.1645]
(c) Microsoft Corporation. Todos os direitos reservados.

PS C:\Users\dfelix\Desktop\Repositorios\treinamentov2> npx cypress run
```

Modo *headless*:

- O modo *headless* não tem interface do usuário (UI). Não precisa da funcionalidade de interface do usuário. A interface do usuário está desabilitada e os aplicativos de interface do usuário não serão iniciados. Isso reduz a quantidade de recursos do sistema usados.

```
(Run Starting)

Cypress: 4.4.1
Browser: Electron 80 (headless)
Specs: 1 found (Login.spec.js)
Searched: cypress\integration\**\*.spec.js

Running: Login.spec.js (1 of 1)

Login screen
  Inputs credentials
```


The Test Runner (Executador dos Testes)



The screenshot shows the Cypress Test Runner interface. The left sidebar contains a dark navigation menu with the following items: 'treinamentov2 main', 'Specs', 'Runs', and 'Settings'. The main area displays a list of test specifications under the 'Specs' tab. The top of the interface shows the Cypress version 'v10.10.0' with an 'Upgrade' button, the browser 'Chrome 107', and links for 'Docs' and 'Log In'. The main content area shows a table of test runs with columns for 'Last updated', 'Latest runs', and 'Average duration'. The table lists several test files under the 'cypress\e2e' and '2-advanced-examples' folders, all showing '5 days ago' for the last update and '--' for the latest runs and average duration.

Visualizar os testes implementados rodando

Conexão com o dashboard, onde podemos visualizar os relatórios de execução dos testes, métricas, etc.

Para configurações no projeto, do dashboard e no dispositivo

Specs	Last updated ?	Latest runs ?	Average duration ?
▼ cypress\e2e			
▼ 2-advanced-examples			
actions.cy.js	5 days ago	--	--
aliasing.cy.js	5 days ago	--	--
assertions.cy.js	5 days ago	--	--
connectors.cy.js	5 days ago	--	--
cookies.cy.js	5 days ago	--	--

Estrutura de pastas



test - treinamentoov2 - Visual Studio Code

EXPLORER

... {} package.json ≡ test U X

≡ test 1 |

✓ TREINAMENTOV2

✓ cypress

> e2e ← E2e contém os arquivos de testes .cy.js

> fixtures ← Fixtures onde são guardados dados de testes como dados de login por exemplo

> support ← Na support são colocados os códigos reutilizáveis

> node_modules

JS cypress.config.js ← Arquivo de configuração geral do cypress, onde é possível ajustar os arquivos de testes, definir a url base do projeto, etc.

{ package-lock.json

{ package.json

i README.md

≡ test U

indra
At the core