Documento de Atividades sugeridas para as MasterClasses

Masterclass 1

Objetivo

O que é qualidade de software?
Time de Software Vs Squad
Plano de Teste de Software
BDD, o que é isso?
Given-When-The

Exercicios Sugeridos

Pergunta: Qual é a finalidade do Gherkin no contexto do Behavior-Driven Development (BDD)?
Resposta: O Gherkin é uma linguagem utilizada para escrever casos de teste em formato legível por humanos, facilitando a comunicação entre as partes interessadas e promovendo a colaboração no desenvolvimento de software.

Pergunta: Quais são as três principais palavras-chave usadas em um cenário de teste do Gherkin e qual é o propósito de cada uma delas?
Resposta: As três palavras-chave principais são:

1. Dado (Given): Define o estado inicial do cenário de teste.
2. Quando (When): Descreve a ação ou evento que ocorre no cenário.
3. Então (Then): Especifica o resultado esperado ou o comportamento desejado após a ocorrência da ação.

Pergunta: Qual é o objetivo do Cucumber na automação de testes?
Resposta: O Cucumber é uma ferramenta de automação de testes que permite a execução de casos de teste escritos em linguagem Gherkin. Seu objetivo é promover a automação de testes baseados em comportamento e facilitar a colaboração entre equipes de desenvolvimento e teste

Pergunta: Quais são os principais benefícios de utilizar o Cucumber como ferramenta de automação de testes?
Resposta: Alguns benefícios do uso do Cucumber são:

4. Permite a escrita de casos de teste em linguagem natural legível por humanos.
5. Facilita a colaboração entre desenvolvedores, testadores e analistas de negócio.
6. Promove a compreensão clara dos comportamentos esperados do software.

7. Possibilita a geração de documentação automatizada a partir dos casos de teste.

Pergunta: Explique como você definiria um cenário de teste utilizando a sintaxe do Gherkin.

Resposta: Um cenário de teste em Gherkin é definido da seguinte forma:

8. Cenário: [Descrição do cenário] Dado [Condição inicial] Quando [Ação realizada] Então [Resultado esperado]
   Onde as palavras-chave Dado, Quando e Então são preenchidas com informações relevantes para cada caso de teste específico.

Masterclass 2

Objetivo

Mais do BDD - agora no detalhe
Requisitos/Testes Funcionais
Requisitos/Testes Não Funcionais
Vamos fazer um plano de testes juntos?
Criar o plano
Escrever as features
Escrever os cenários

Exercicios Sugeridos

O que é o Cucumber-JS?

Resposta: O Cucumber-JS é uma ferramenta de automação de testes de software que permite a implementação da técnica de desenvolvimento orientado por comportamento (BDD) em projetos JavaScript, facilitando a escrita de casos de teste em linguagem natural legível por humanos (Gherkin) e a automação de sua execução.

Como você descreveria a estrutura básica de um arquivo de recurso (feature) do Cucumber?

Resposta: Feature: [Nome do recurso] [Descrição do recurso] Scenario: [Nome do cenário] Given [Dado] When [Quando] Then [Então]

Como você mapearia um passo de teste em um arquivo de recurso (feature) para sua implementação em JavaScript usando o Cucumber-JS?

Resposta:

```
GIVEN("DADO", function (variavel) {
```

```
  // Implementação do código em JavaScript
});

WHEN("when", function (variavel) {
  // Implementação do código em JavaScript
});

THEN("then", function (variavel) {
  // Implementação do código em JavaScript
});
```

O que é qualidade de software?

Resposta: Qualidade de software é a medida em que um software atende aos requisitos e expectativas do usuário, apresentando confiabilidade, desempenho, usabilidade e outras características desejáveis.

O que são testes funcionais?

Resposta: Testes funcionais são testes que visam verificar se o software atende aos requisitos funcionais especificados, ou seja, se ele executa corretamente as funções esperadas pelo usuário.

O que são requisitos funcionais?

Resposta: Requisitos funcionais são especificações que descrevem as funções e comportamentos que o software deve ser capaz de executar. Eles definem o que o software deve fazer.

Dê um exemplo de requisito funcional de um login de sistema.

Resposta: Exemplo: "O sistema deve permitir que o usuário faça login fornecendo um nome de usuário e senha válidos."

O que são requisitos não funcionais?

Resposta: Requisitos não funcionais são especificações que descrevem atributos do software, como desempenho, usabilidade, segurança, confiabilidade e outros aspectos que não estão diretamente relacionados às funcionalidades do sistema.

Dê um exemplo de requisito não funcional de um login de sistema.

Resposta: Exemplo: "O tempo de resposta do sistema de login deve ser inferior a 3 segundos em condições normais de uso."

O que são testes não funcionais?

Resposta: Testes não funcionais são testes que avaliam os atributos não funcionais de um software, como desempenho, segurança, usabilidade, escalabilidade, entre outros.

Descreva o teste do requisito não funcional: "O tempo de resposta do sistema de login deve ser inferior a 3 segundos em condições normais de uso."

Resposta: Esse teste envolve a medição do tempo de resposta do sistema de login em condições normais de uso e verificação se o tempo de resposta está dentro do limite estabelecido (inferior a 3 segundos).

O que é o STLC (Software Testing Life Cycle)?

Resposta: O STLC é uma abordagem sistemática para realizar atividades de teste em um projeto de desenvolvimento de software, compreendendo fases como planejamento de testes, análise de requisitos, desenho de testes, implementação de testes, avaliação de testes e encerramento de testes.

Exercícios práticos de requisitos funcionais e não funcionais.

…Wip…

Masterclass 3

Objetivo

Artefatos e métricas STLC
Ferramentas
Cucumber
Cypress

Exercicios Sugeridos

Apartir do github: https://github.com/rodrigoror/duckduckgo-cypress-preprocessor
Crie cenários de teste cucumber/cypress para os seguintes sites:

1. https://www.google.com/
2. https://aws.amazon.com/
3. https://azure.microsoft.com/en-us/
4. https://reactnative.dev/

Você pode fazer fork do projeto, ou sugerir aos alunos criarem pull requests para o projeto, incentivando o uso do GIT e o open-source.

Masterclass 4

Objetivo

Gates
Pirâmide de Testes
Ferramentas
Cypress

Exercicios Sugeridos

Dado o BDD a seguir, crie um código JS para atender os cenários.

Crie o projeto no Github e adicione todos os cenários dentro.

Cenário: Obter o tamanho da string
Dado uma string "Hello, World!"
Quando eu chamar o método length da string
Então o resultado deve ser 13
Resposta

```
const str = "Hello, World!";
const length = str.length;
console.log(length); // Output: 13
```

Cenário: Converter a string para letras maiúsculas
Dado uma string "Hello, World!"
Quando eu chamar o método toUpperCase da string
Então o resultado deve ser "HELLO, WORLD!"
Resposta

```
const str = "Hello, World!";
const upperCaseStr = str.toUpperCase();
console.log(upperCaseStr); // Output: HELLO, WORLD!
```

Cenário: Converter a string para letras minúsculas
Dado uma string "Hello, World!"
Quando eu chamar o método toLowerCase da string
Então o resultado deve ser "hello, world!"
Resposta

```
const str = "Hello, World!";
const lowerCaseStr = str.toLowerCase();
console.log(lowerCaseStr); // Output: hello, world!
```

Cenário: Dividir a string em substrings
Dado uma string "Hello, World!"

Quando eu chamar o método split da string, passando ", " como separador
Então o resultado deve ser um array contendo as substrings ["Hello", "World!"]
Resposta
```
const str = "Hello, World!";
const substrings = str.split(", ");
console.log(substrings); // Output: ["Hello", "World!"]
```

Cenário: Substituir uma substring na string
Dado uma string "Hello, World!"
Quando eu chamar o método replace da string, substituindo "World" por "John"
Então o resultado deve ser "Hello, John!"
Resposta
```
const str = "Hello, World!";
const newStr = str.replace("World", "John");
console.log(newStr); // Output: Hello, John!
```

Agora, gere métodos de teste para cada um dos cenários.

Resposta:
```
const request = require('supertest');
const app = require('./app'); // Substitua './app' pelo caminho do seu aplicativo

describe('Testes de tratamento de strings', () => {
  test('Obter o tamanho da string', async () => {
    const response = await request(app).get('/length');
    expect(response.status).toBe(200);
    expect(response.body.length).toBe(13);
  });

  test('Converter a string para letras maiúsculas', async () => {
    const response = await request(app).get('/toUpperCase');
    expect(response.status).toBe(200);
    expect(response.body).toBe('HELLO, WORLD!');
  });

  test('Converter a string para letras minúsculas', async () => {
    const response = await request(app).get('/toLowerCase');
    expect(response.status).toBe(200);
    expect(response.body).toBe('hello, world!');
  });

  test('Dividir a string em substrings', async () => {
    const response = await request(app).get('/split');
    expect(response.status).toBe(200);
    expect(response.body).toEqual(['Hello', 'World!']);
```

```
  });

  test('Substituir uma substring na string', async () => {
    const response = await request(app).get('/replace');
    expect(response.status).toBe(200);
    expect(response.body).toBe('Hello, John!');
  });
});
```

Jest + Supertest
https://github.com/rodrigoror/demo-express-jest-supertest/tree/master

Agora, adicione o cucumber para gerenciar estes testes.

Starter Answer:
https://github.com/rodrigoror/login-node-jest/tree/cucumber-app