Valderi Leithardt, Dr.

IPW (Class eight notes)

Summary

- Review of previous class *
- Application with users
- Tests
- Practical class
- Work
- Exercises

OpenAPI *(Review of previous class)*

- Visão Geral do Google Cloud API

- Trabalhar com definições do OpenAPI para APIs HTTP AWS (2023)

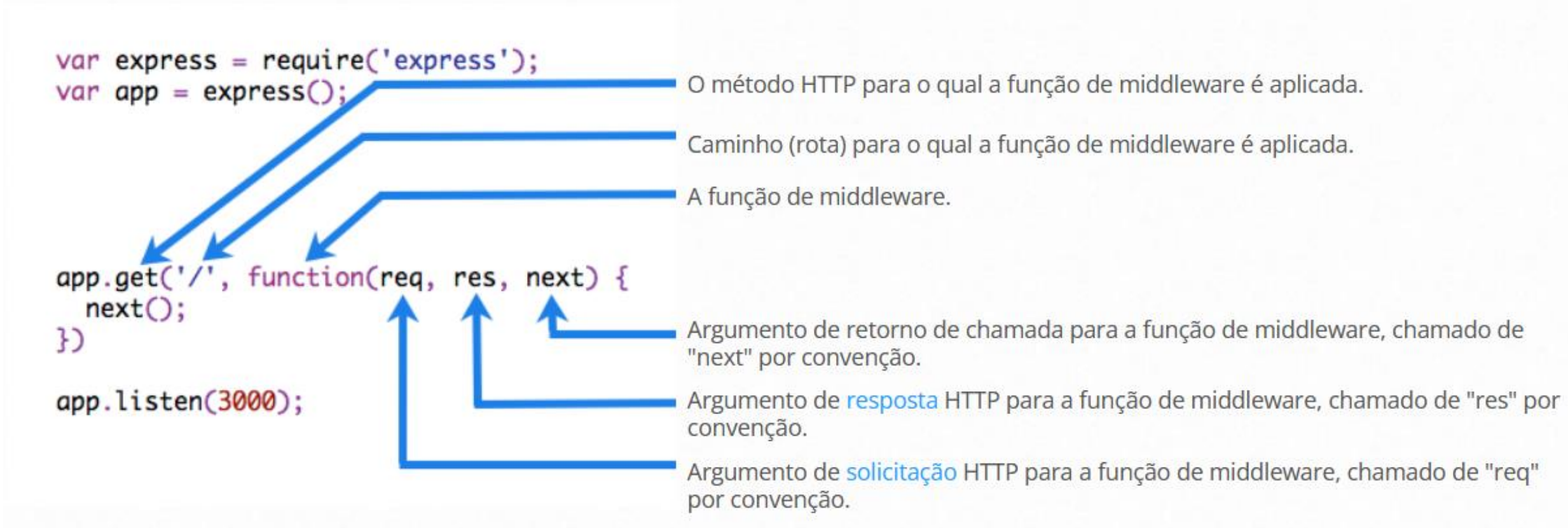
- Representação OpenAPI IBM (2023).

- Especificação OpenAPI, disponível em: https://swagger.io/specification/.

- A especificação Open API teve a sua última versão liberada no dia 16 de fevereiro de 2021. Houve algumas mudanças, inclusive relacionadas às quebras de compatibilidade, algo que não fica claro apenas olhando o número da versão acrescida (de 3.0.0 para 3.1.0). Conforme descreve GFT (2023).

Middleware *(Review of previous class)*

- O exemplo mostra os elementos de uma chamada de função de middleware:



Exemplo middleware Online (2023).
https://expressjs.com/en/starter/hello-world.html

Application with users

- Codificar páginas da web ou aplicações, uma das tarefas mais comuns é manipular documentos da web de alguma forma. Normalmente isso é feito usando o Document Object Model (DOM), um conjunto de APIs para controlar o HTML e a informação sobre os estilos que usa fortemente o objeto Document.
- O Document Object Model (DOM) é uma interface de programação para os documentos HTML e XML. Representa a página de forma que os programas possam alterar a estrutura do documento, alterar o estilo e conteúdo. O DOM representa o documento com nós e objetos, dessa forma, as linguagens de programação podem se conectar à página.

- Introduction DOM: https://developer.mozilla.org/en-US/docs/Web/API/Document_Object_Model/Introduction

Tests

O Conceitos de teste de Software

- O teste depende do contexto em que é solicitado, sendo assim existem diferentes tipos, técnicas e ferramentas específicas.
- Segundo (Santos, 2023), Diversos são os frameworks e ferramentas disponíveis para testes automatizados de API que estão disponíveis na literatura, porém frameworks mocha e chai, para criação de scripts de testes automatizados, que atendam as especificações da API REST a ser testada, são pouco explorados pela literatura.
- Existem várias técnicas de testes de Software que são descritas na literatura, porém duas delas são as abordagens estratégicas mais utilizadas: Teste Manual e Teste Automatizado.

Tests

- De acordo com o (PRESSMAN, 2011) as técnicas de testes têm como objetivo identificar as condições e informações dos dados. Escolher qual a técnica na qual será utilizada, depende de alguns fatores, podendo destacar:

✓ Nível de complexidade do componente ou sistema a ser testado;
✓ Requisitos contratuais ou requisitos solicitados pelo cliente;
✓ Níveis e tipos de riscos nos quais tendem a ser influências diretas na qualidade do Software;
✓ Documentação disponível para teste;
✓ Conhecimento e habilidade do analista de testes;
✓ Ferramentas disponíveis;
✓ Tempo e custo;
✓ Modelo do ciclo de vida utilizado no desenvolvimento.

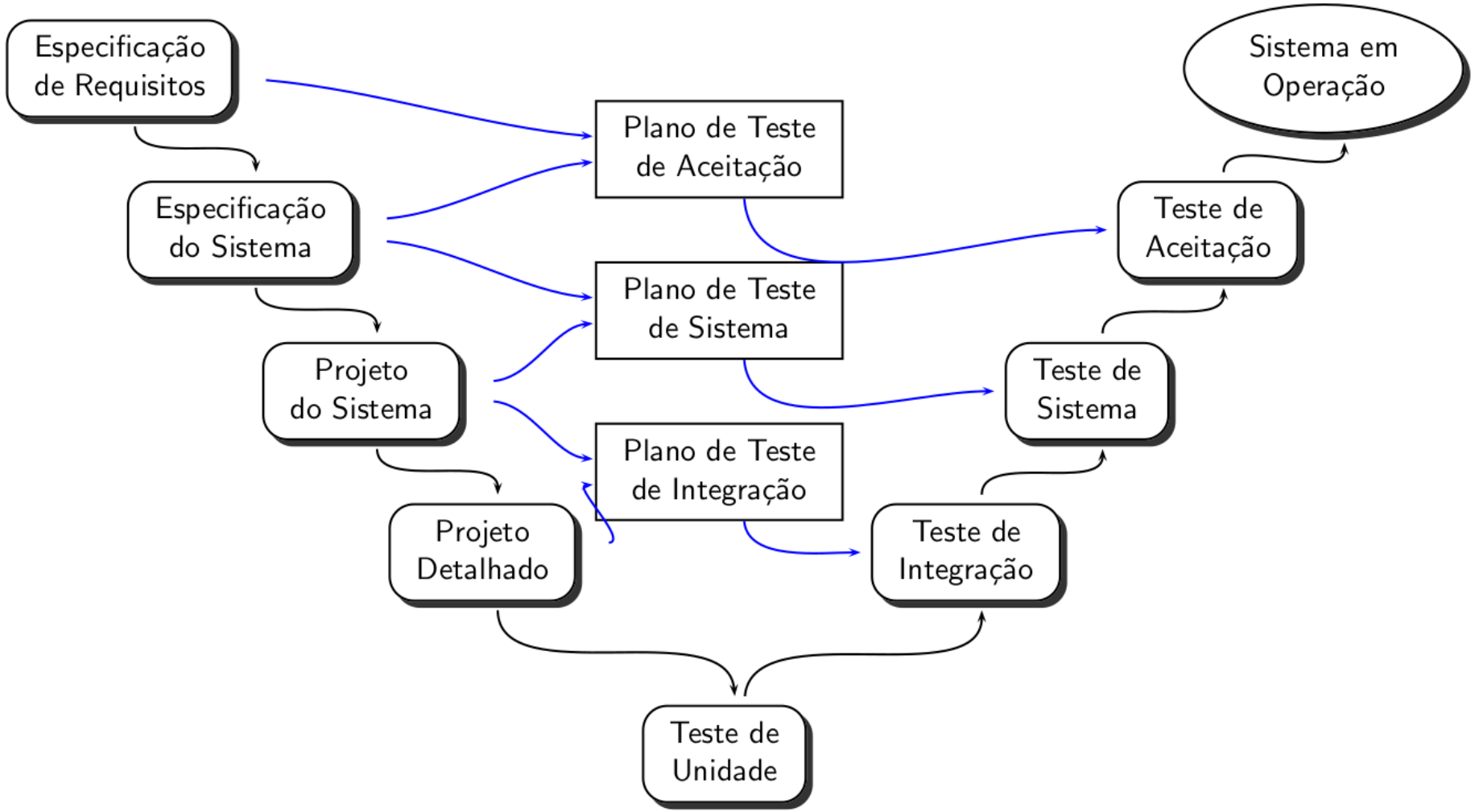
Tests

Tipos de testes

- Existem diferentes tipos de testes que podemos aplicar em conjuntos com Node.js.:
- Testes unitários: são utilizados para testar pequenas partes do código, como funções, para garantir que elas funcionam corretamente;
- Testes de integração: testam a interação entre diferentes partes do sistema;
- Testes de aceitação: garantem que o software atende aos requisitos especificados pelo cliente;
- Testes de desempenho/carga: testam a capacidade de processamento do sistema;
- Testes de segurança: testam a segurança da aplicação, buscando vulnerabilidades e falhas de segurança.

Tests - Fases

- Um modelo clássico, denominado Modelo-V, alinha as atividades de desenvolvimento com as atividades de teste.



Tests

Ferramentas de teste

- Há diversas ferramentas disponíveis para testar aplicações em Node.js. Algumas das mais populares são:
- Mocha: um framework de teste que suporta testes assíncronos e síncronos;
- Chai: uma biblioteca de asserções que funciona com o Mocha;
- Sinon: uma biblioteca de testes de unidade que permite testar funções isoladamente, (Tutorial);
- Supertest: uma biblioteca para testar APIs HTTP que pode ser usada com o Mocha;
- Jest: uma biblioteca para testar APIs HTTP que pode ser usada com o Mocha.

Tests - Mocha

É uma estrutura de testes do JavaScript, que possui diversos recursos que são executados com a utilização do Node.js e também em navegadores, possibilitando que os testes sejam executados em série e a geração de relatórios precisos durante o mapeamento de exceções que não são capturadas para cenários de testes corretos. *Disponível em: https://github.com/mochajs/mocha

```
describe("hooks", function() {

    before(function() {
    // Runs before all tests in this block
    });

    after(function() {
    // Runs after all tests in this block
    });

    beforeEach(function() {
    // Runs before each test in this block
    });

    afterEach(function() {
    // Runs after each test in this block
    });

    // Test cases
    });
```

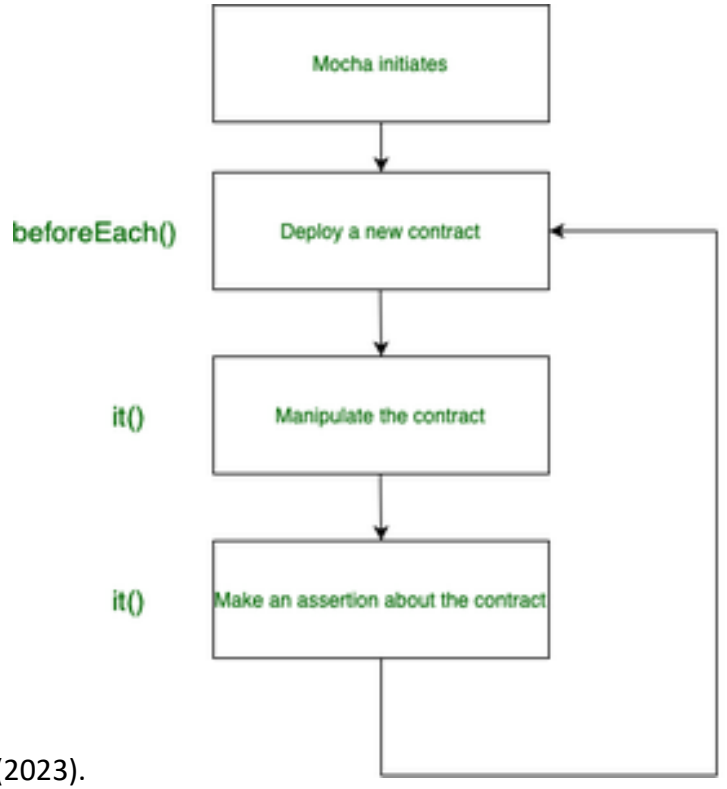


Figura Adaptada de geeksforgeeks (2023).

Tests - CHAI

- É descrito como uma biblioteca baseada em BDD/TDD que é executada em Node.js, também pode ser realizada em navegadores Web que utilizam aplicações JavaScript.

- Chai é uma biblioteca para a asserção dos resultados recebidos a os resultados esperados, ou seja, criamos uma relação entre os resultados, dessa forma caso a relação seja verdadeira o teste irá passar.

- TDD é uma metodologia de desenvolvimento, na qual o desenvolvedor começa a desenvolver a aplicação a partir dos testes.

✓ Tutorial: Testando Aplicações RESTFul API em Node.Js com Mocha & Chai

https://github.com/glaucia86/tutorial-crud-nodejs-mocha-chai

Tests

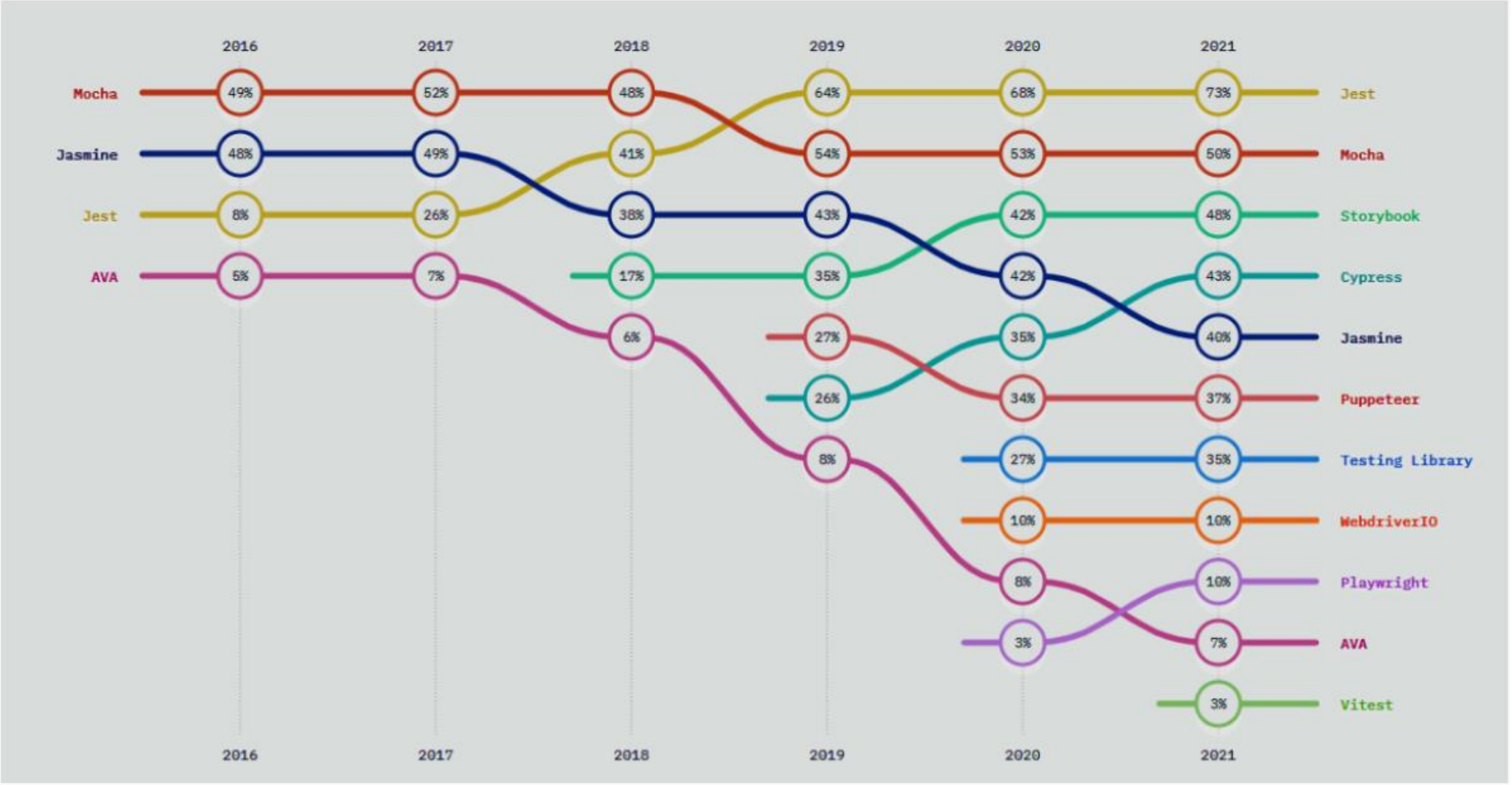


Figura extraída (2023), disponível em: [JavaScript unit testing frameworks in 2022: A comparison]

Tests

Que razão é importante testar aplicações em Node.js

- ✓ Garantir que o software está funcionando corretamente e que as alterações realizadas não afetaram o funcionamento da aplicação;
- ✓ Reduzir os custos de manutenção corrigindo erros no início do processo de desenvolvimento;
- ✓ Permitir que desenvolvedores trabalhem em equipe com segurança, com a garantia de que todas as mudanças são cuidadosamente testadas antes de serem implementadas;
- ✓ Recomenda-se que em testes, é necessário deletar a base de dados que foi criada anteriormente, para que novos testes sejam executados novamente. Evitando que a execução anterior não impacte sobre as próximas.

Exercises

- 2º Trabalho part I (**IPW_IP 2324 1 A2**) disponibilizado em:
https://github.com/isel-leic-ipw/2324i-IPW-LEIC31D/wiki/IPW_IP-2324-1-A2

- 3º Trabalho, part II (**IPW_IP 2324 1 A3**) disponibilizado em:
https://github.com/isel-leic-ipw/2324i-IPW-LEIC31D/wiki/IPW_IP-2324-1-A3

** Concluir exercícios da aulas anteriores.

References

- Santos, Taynara Luana Caetano dos. "Adherence of automated testing techniques in a Rest Api: approach in a marketplace integrator Hub application" Dissertação de Mestrado, disponível em: http://repositorio.utfpr.edu.br/jspui/handle/1/31733

- Rui Tiago Bugalho Monteiro " Arquitetura Orientada a Componentes para uma Web Responsiva", Dissertação de Mestrado disponível em: https://hdl.handle.net/10216/88459

- PRESSMAN, R. S. Engenharia de Software - Uma Abordagem Profissional 2011. Acesso 2023, em: https://www.academia.edu/42042370/Engenharia_de_Software_Uma_Abordagem_Profi_ssional

https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/RegExp/test - Acesso em 2023

- https://mochajs.org/ - Acesso em 2023

- Também foram realizadas adaptações e modificações com base no material disponibilizado por Professor Luís Falcão, acesso online em: https://github.com/isel-leic-ipw/
- Aulas gravadas Professor Falcão:
- Aula 10 (20/10): https://videoconf-colibri.zoom.us/rec/share/6D0EZ3zXSdMCfiUpjVwA7LJXoUH4ufPgTRO7-jOhBwFWLLUOGkInOs4gWSDMJkCm.gzOlP10sgmImgsA7
- Aula 11 (23/10): https://videoconf-colibri.zoom.us/rec/share/wdPvM1T2baNesdRsoZG2Bj7Hp_s1qHFKqPl97IvqaXP1BQvIYPWgXJ4WeJNDyoqb.JBAS33IhoihpT2Xa

* Todos os links foram acessado em 2023

Valderi Leithardt, Dr.

Professor IPW

valderi.leithardt@isel.pt