Valderi Leithardt, Dr.

IPW (Class seven notes)

Summary

- Review of previous class *
- Express: middlewares, aplicação, OpenAPI
- Prática
- Exercises

NPM *- Review of previous class*

Identificar um projeto por meio de metadados:

- ✓ O nome é x,z,y-npm;
- ✓ A versão é 1.0.0;
- ✓ A descrição informa npm guide for beginner;
- ✓ O ponto de entrada do projeto ou o arquivo principal é beginner-npm.js;
- ✓ As palavras chave ou tags para encontrar o projeto no repositório são npm, example e basic;
- ✓ O autor do projeto é X,Y, X;
- ✓ Este projeto está licenciado sob o MIT;
- ✓ As dependências ou outros módulos que esse módulo usa são express 4.16.4.

```
{
  "name": "xyz-npm",
  "version": "1.0.0",
  "description": "npm guide for beginner",
  "main": "beginner-npm.js",
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1"
  },
  "keywords": [
    "npm",
    "example",
    "basic"
  ],
  "author": "X,Y, Z",
  "license": "MIT",
  "dependencies": {
    "express": "^4.16.4"
  }
}
```

* Material adaptado conforme descrito NPM, 2023

File System do Node.js - Review of previous class

- Além de ler arquivos, podemos usar o módulo FS para criar e escrever em arquivos. Aqui está um exemplo simples de como criar um arquivo e escrever conteúdo nele usando o método fs.writeFile():

```
7
8   const content = "Este é o conteúdo que será escrito no arquivo.";
9
10  fs.writeFile("/caminho/do/novo-arquivo.txt", content, (err) => {
11    if (err) throw err;
12    console.log("O arquivo foi salvo com sucesso!");
13  });
```

- Neste exemplo, esta sendo criado um arquivo chamado novo-arquivo.txt e escrevendo o conteúdo da variável content no arquivo usando o método fs.writeFile().

Fs Promises API – Acesso em (2023)
https://nodejs.org/api/fs.html

Módulo fs do Node.js – Acesso em (2023)
https://nodejs.reativa.dev/0046-node-module-fs/index

Fetch API - Review of previous class

- O Fetch API é uma interface JavaScript moderna para fazer requisições HTTP/HTTPS de forma assíncrona. Permitindo os desenvolvedores criarem aplicações web mais interativas e dinâmicas, oferecendo uma maneira mais intuitiva e fácil de realizar chamadas de rede.

- Fetch são as abstrações da Interface do HTTP Request, Response, Headers (en-US), e Body payloads, juntamente com global fetch (en-US) método para iniciar requisições de recursos assíncronos. Como os componentes principais do HTTP são abstraidos como objetos de JavaScript, torna-se fácil APIs fazer uso das funcionalidades.

Material adaptado de MDN (2023).

Import - ***Review of previous class***

- A declaração estática import é usada para importar vínculos que são exportados por um outro módulo. Os módulos importados estão em strict mode, declarado como tal ou não. A declaração import não pode ser usada em scripts embutidos, a menos que tal script tenha um type="module". Há também uma função dinâmica import(), que não requer scripts de type="module".

- A compatibilidade com versões anteriores pode ser garantida usando o atributo nomodule na tag de script.

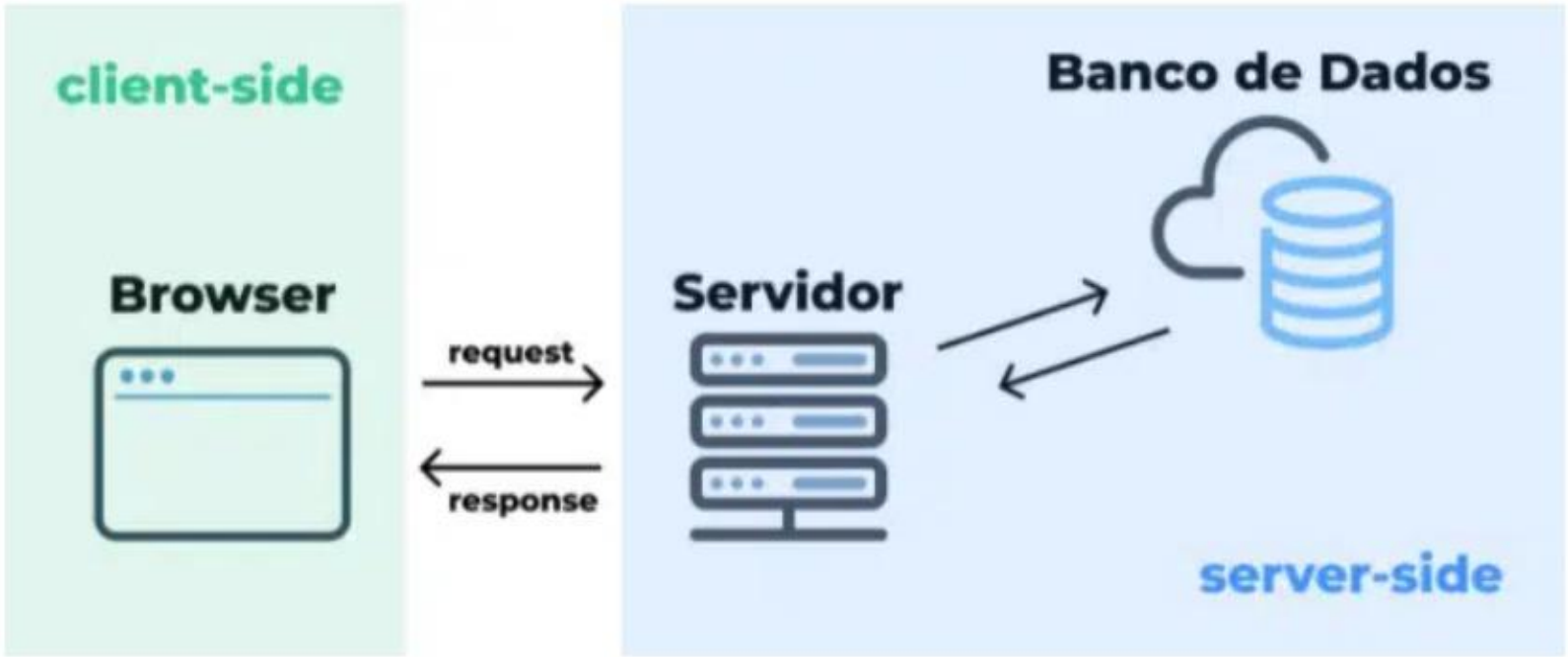
Import (MDN) – Acesso em (2023).

https://developer.mozilla.org/pt-BR/docs/Web/JavaScript/Reference/Statements/import

Import – Acesso em (2023).

https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Statements/import

Aplicação


client-side
Browser
request
response
Servidor
Banco de Dados
server-side

OpenAPI

- Uma API (Application Programming Interface), é um conjunto de estruturas de programação para acesso a um determinado sistema. As APIs foram criadas para permitir que máquinas pudessem se comunicar entre si. Portanto, uma API nada mais é do que uma maneira para permitir que um sistema troque informações com outro sistema.

- Uma OpenAPI (API aberta) é uma API disponível para desenvolvedores externos. Isto não significa que seja uma API livre sem autenticação, sem controle ou sem custos.

- Uma determinada organização pode criar uma API aberta gratuita ou uma API aberta que seja paga. O conceito do “Open” tem a ver com o fato da API estar disponível para outras empresas e desenvolvedores. Do contrário, caso um sistema tenha uma API disponível, mas que seja acessível apenas a sistemas da própria empresa, então é uma API fechada.

* OpenAPI 2023, disponível em: https://www.openapis.org/

OpenAPI

REST API

- REST API (Representational State Transfer Application Programming Interface) é um estilo de arquitetura que define um conjunto de restrições e propriedades baseadas em HTTP fornecendo interoperabilidade entre sistemas de computadores na internet (ou rede local). REST permite que os sistemas consumidores acessem e manipulem representações textuais de recursos - as "coisas" da vida real em REST se chamam recursos - usando um conjunto pré definido de operações.

- Por colocar mais restrições do que WebServices SOAP e utilizar mais recursos já nativos do protocolo HTTP, acabou padronizando bastante a forma de comunicação e isso agilizou o desenvolvimento das integrações.

O REST enfatiza:

Guia de Design REST

- Simplicidade através de uma comunicação bem definida, com algumas restrições que facilitam a padronização e stateless, ou seja, todas as informações necessárias para o processamento de uma requisição devem estar contidas naquela requisição.

- Extensibilidade dado que um determinado recurso pode ser representado em diferentes formatos e até mesmo versões, além da possibilidade de se adicionar novos recursos sem precisar alterar os já existentes.

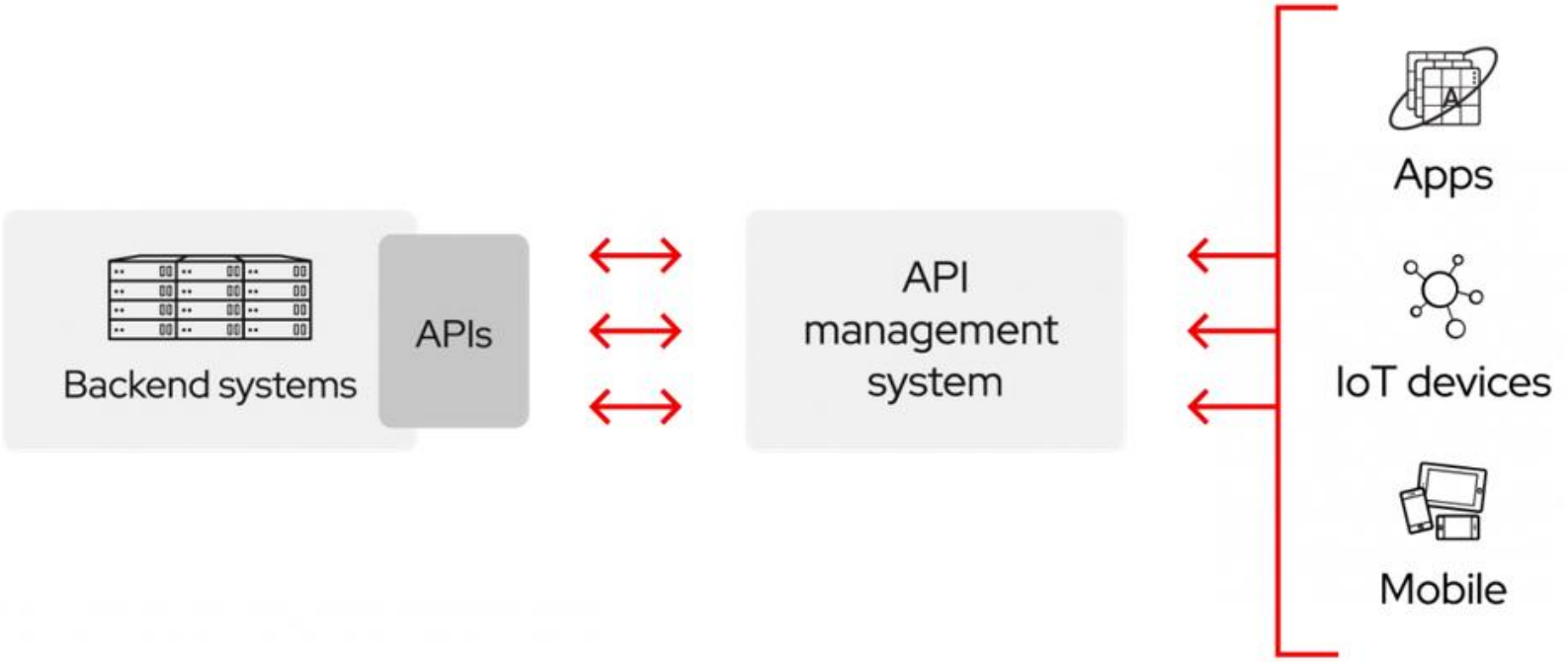
- Confiabilidade porque existe uma definição clara das ações que podem ser feitas e dos seus efeitos colaterais.

- Performance porque faz parte dos principais pilares do REST o uso de cache em todas as camadas possíveis através de uma semântica bem definida, além de uma arquitetura baseada em separação de camadas, permitindo que partes diferentes do sistema possam ser escaladas de forma independente e isolada.

OpenAPI

- ❖ APIs funcionam como se fossem contratos, com documentações que representam um acordo entre as partes interessadas. Se uma dessas partes enviar uma solicitação remota estruturada de uma forma específica, isso determinará como a aplicação da outra parte responderá.
- ❖ APIs simplificam a forma como os desenvolvedores integram novos componentes de aplicações a uma arquitetura preexistente.

* Figura extraída e texto descrito conforme em RedHat (2023).



API do Google Maps, Conforme RedHat (2023).

OpenAPI

- 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

- Segundo AWS (2023), O middleware começou como uma ponte entre novas aplicações e sistemas herdados antes de ganhar popularidade na década de 1980. Os desenvolvedores inicialmente o usavam para integrar novos programas com sistemas anteriores sem reescrever o código. O middleware se tornou uma importante ferramenta de comunicação e gestão de dados em sistemas distribuídos.
- Os desenvolvedores usam middleware para dar suporte ao desenvolvimento de aplicações e simplificar os processos de design. Assim, eles ficam livres para se concentrar na lógica e nos recursos de negócios em vez da conectividade entre diferentes componentes de software.
- Sem o middleware, seria necessário criar um módulo de troca de dados para cada componente de software que se conecta à aplicação. Isso é desafiador, porque as aplicações modernas consistem em vários microsserviços ou pequenos componentes de software que interagem entre si.

Middleware

- Segundo (artigo, 2023), middlewares são funções que têm como parâmetros o request, response e next, ou seja, a próxima função middleware a ser executada. No escopo dessas funções, podemos escrever código para:
- Realizar mudanças na requisição HTTP;
- Encerrar o ciclo request-response;
- Chamar o próximo middleware;
- Outras ações.

- OBS.: Se a middleware atual não chamar a next(), a solicitação ficará suspensa.

Middleware

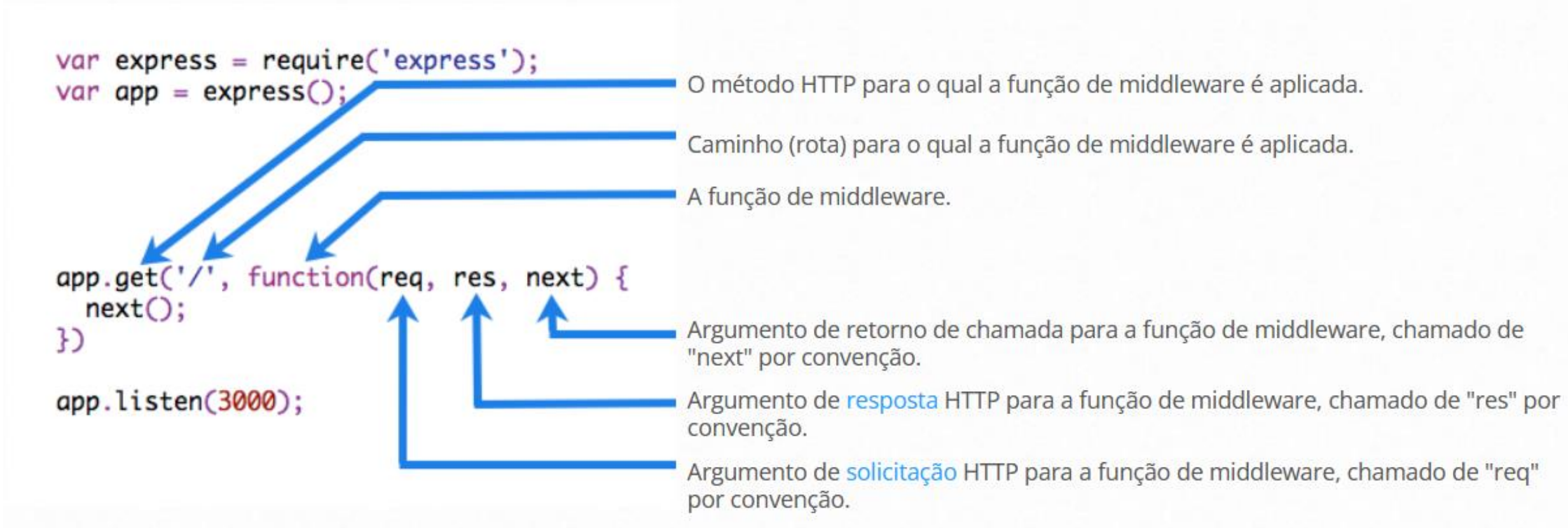
- Funções de Middleware são funções que tem acesso ao objeto de solicitação (req), o objeto de resposta (res), e a próxima função de middleware no ciclo solicitação-resposta do aplicativo. A próxima função middleware é comumente denotada por uma variável chamada next.

- Funções de middleware podem executar as seguintes tarefas:

- Executar qualquer código.
- Fazer mudanças nos objetos de solicitação e resposta.
- Encerrar o ciclo de solicitação-resposta.
- Chamar o próximo middleware na pilha.

Middleware

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



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

Middleware - Aplicação

- Um aplicativo Express pode usar os seguintes tipos de middleware:

- Middleware de nível do aplicativo
- Middleware de nível de roteador
- Middleware de manipulação de erros
- Middleware integrado
- Middleware de Terceiros

- É possível carregar middlewares de nível de roteador e de nível do aplicativo com um caminho de montagem opcional.
- É possível também carregar uma série de funções de middleware juntas, o que cria uma sub-pilha do sistema de middleware em um ponto de montagem.

Middleware - Aplicação

- Um **aplicativo Express** pode usar os seguintes tipos de middleware:
- Middleware de nível do aplicativo
- Middleware de nível de roteador
- Middleware de manipulação de erros
- Middleware integrado
- Middleware de Terceiros

- É possível carregar middlewares de nível de roteador e de nível do aplicativo com um caminho de montagem opcional.
- É possível também carregar uma série de funções de middleware juntas, o que cria uma sub-pilha do sistema de middleware em um ponto de montagem.
- Para obter uma lista parcial de funções de middleware de terceiros que são comumente utilizadas com o Express, consulte: https://expressjs.com/pt-br/resources/middleware.html

Exercises

- 2º Trabalho part 1 disponibilizado em:

https://github.com/isel-leic-ipw/2324i-IPW-LEIC31D/wiki

Cap 20 – livro ENG: https://eloquentjavascript.net/20_node.html

* Leitura https://www.w3.org/TR/webarch/

** Concluir exercícios da aulas anteriores.

References

- https://eloquentjavascript.net/20_node.html
- https://www.tutorialspoint.com/expressjs/index.htm
- https://expressjs.com/en/starter/installing.html
- https://www.w3cschool.cn/doc_express/express-guide-writing-middleware.html
- 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:

- **Lesson 09: Simple fetch wrapper to overcome the ticketmaster API Rate… · isel-leic-ipw/2324i-IPW-LEIC31D@d544cbc**
- Aula 11 (20/10): https://videoconf-colibri.zoom.us/rec/share/VK_Z9n-pAwG_mgBcQqnQALsKCAnrOs2zvCN5K1yL_P19pOI9kTBmYzCuiIAlAblL.cdl0vx74ZBQHzWd2

Valderi Leithardt, Dr.

Professor IPW

valderi.leithardt@isel.pt