

Desenvolvimento de APIs

Capítulo 1. Introdução

Prof. Guilherme Assis



Aula 1.1. Backend vs Frontend

☐ Backend.

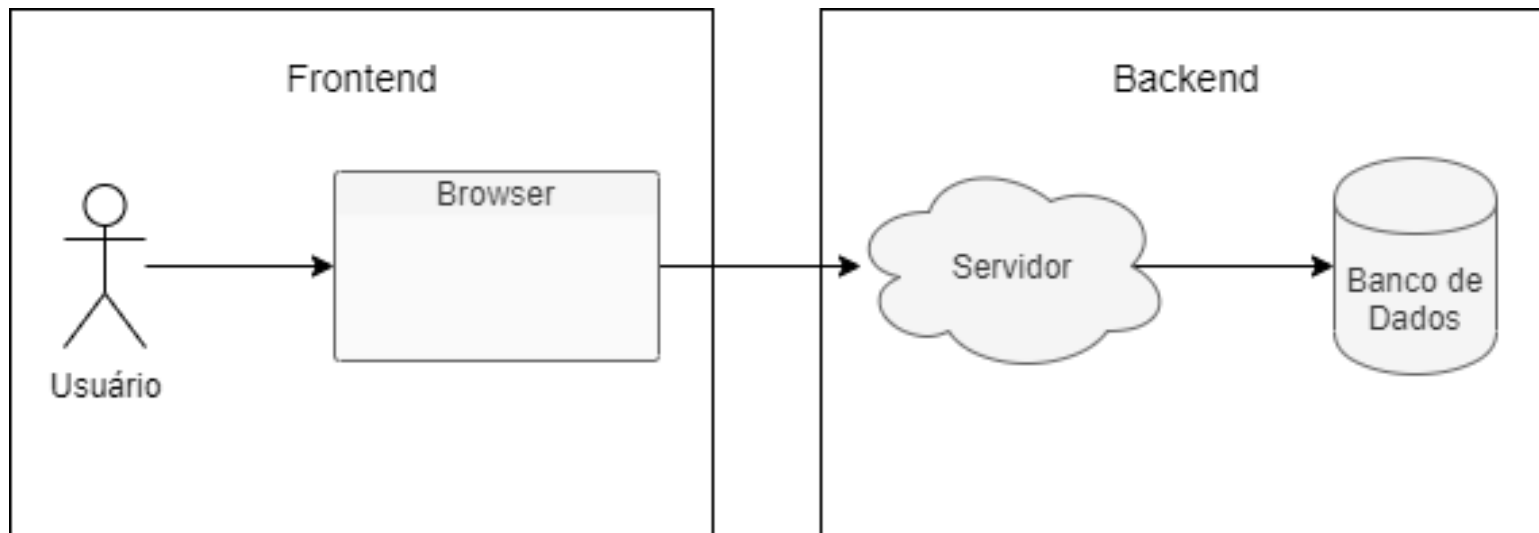
☐ Frontend.

- O backend se refere a parte que fica hospedada no servidor, focando principalmente em como a aplicação funciona;
 - Responsável por interagir com o banco de dados, gravando e buscando registros.
- Exemplos de linguagens:
 - Java;
 - C#;
 - PHP;
 - JavaScript (Node.js).

- Frontend é parte da aplicação com a qual o usuário interage:
 - HTML;
 - CSS;
 - JavaScript.
- As páginas podem ser montadas no servidor e devolvidas prontas para o usuário ou montadas no próprio browser do usuário;
 - Server side rendering e client side rendering.

Backend vs Frontend

- Exemplo: usuário navegando por um e-commerce na web.



Conclusão

☒ Backend.

☒ Frontend.

■ Próxima aula

□ APIs.



Aula 1.2. APIs

☐ API.

☐ REST.

☐ URI.

- API: Application Programming Interface.
- Conjunto de serviços que são expostos de forma a permitir a comunicação entre sistemas.
- Uma aplicação acessa recursos da outra sem saber como foram implementados;
 - Quem está fornecendo tem um maior controle sobre o que está sendo feito.
- Pode ser vista como um contrato, representado pela documentação.

- Serviços que fazem parte de uma API são chamados de webservices.
- Um webservice somente transmite as informações, não sendo por si só uma aplicação possível de ser acessada pela web.
- Muitas empresas estão criando APIs de suas aplicações, de forma a possibilitar a fácil integração de outras aplicações.

- Facilidade na integração de sistemas;
 - Depende do protocolo HTTP e um formato, como o JSON.
- Favorece a reutilização de software;
 - Integração entre aplicações desenvolvidas em linguagens e plataformas diferentes.
- Segurança na integração;
 - Evita que aplicações integrem diretamente pelo banco de dados.

- REST é a sigla para Representational State Transfer e foi descrito por Roy Fielding, um dos criadores do protocolo HTTP.
- Utiliza uma URI (Uniform Resource Identifier) para realizar uma chamada de serviço
 - URIs são interfaces de utilização do serviço, servindo como um contrato;
 - Exemplo: <http://www.teste.com.br/clientes/2>.
- URIs que uma API disponibiliza também são conhecidas como seus endpoints.

- Através dos endpoints é possível realizar várias operações.
- Além do endpoint, o cliente precisa informar o método HTTP:
 - GET: obter os dados de um recurso.
 - POST: criar um novo recurso.
 - PUT: substituir os dados de um determinado recurso.
 - DELETE: excluir um determinado recurso.

- Exemplos de chamadas:
 - GET /clientes: recuperar os dados de todos os clientes;
 - GET /clientes/id: recuperar os dados de um determinado cliente;
 - POST /clientes: criar um novo cliente;
 - PUT /clientes: atualizar os dados de um determinado cliente;
 - DELETE /clientes/id: excluir um determinado cliente.
- Geralmente aplicações web que trabalham com REST utilizam o formato JSON.
 - { "id": 3, "nome": "João", "idade": 32, "sexo": "Masculino" }

Conclusão

☑ API.

☑ REST.

☑ URI.

■ Próxima aula

□ Node.js.



Aula 1.3. Node.js

☐ Node.js.

- O Node.js foi criado em 2009 na tentativa de resolver o problema de arquiteturas bloqueantes.



- Plataformas como .NET, Java ou PHP paralisam um processamento enquanto realizam um processo de I/O no servidor.

- Esta paralisação é o chamado modelo bloqueante (Blocking-Thread).
- Enquanto uma requisição é processada, as demais ficam ociosas em espera.
- Esses servidores criam várias threads para darem vazão a fila de espera, podendo ser necessário fazer upgrade nos hardwares.

- O Node.js possui uma arquitetura não bloqueante (non-blocking thread).
- Ele apresenta uma boa performance em consumo de memória e utilizando ao máximo o poder de processamento dos servidores.
- Nele as aplicações são single-thread, ou seja, cada aplicação possui um único processo.
- Utiliza bastante a programação assíncrona, com o auxílio das funções de callback do JavaScript.

- Em uma arquitetura bloqueante, o jeito de lidar com essa concorrência seria criar múltiplas threads para lidar com as diversas requisições.
- O NodeJS foi criado utilizando o V8, que é um motor JavaScript de código aberto, criado pela Google e utilizado no Google Chrome.
- Com o NodeJS é possível executar o código JavaScript no servidor.
- Ele mantém um serviço rodando no servidor, que faz a interpretação e execução de códigos JavaScript.

- A criação do Node.js está muito ligada com a crescente utilização das SPAs.
- Com o Node.js também é possível criar aplicações desktop, com o auxílio de ferramentas como Electron por exemplo.
- Node.js pode ser utilizado em aplicações Real-Time, como aplicações colaborativas como aplicativos de mensagens e jogos on-line.

- O Node.js não é muito recomendado para aplicações que lidam com algoritmos complexos e que consumam muita CPU.
- Esta limitação pode ser contornada com a utilização de Workers.

- ☑ Node.js utiliza o motor JavaScript V8 do Google para rodar código JavaScript no servidor.
- ☑ O Node.js possui uma arquitetura não bloqueante.
- ☑ Muito utilizado no mercado.

- ❑ Node.js Event Loop.



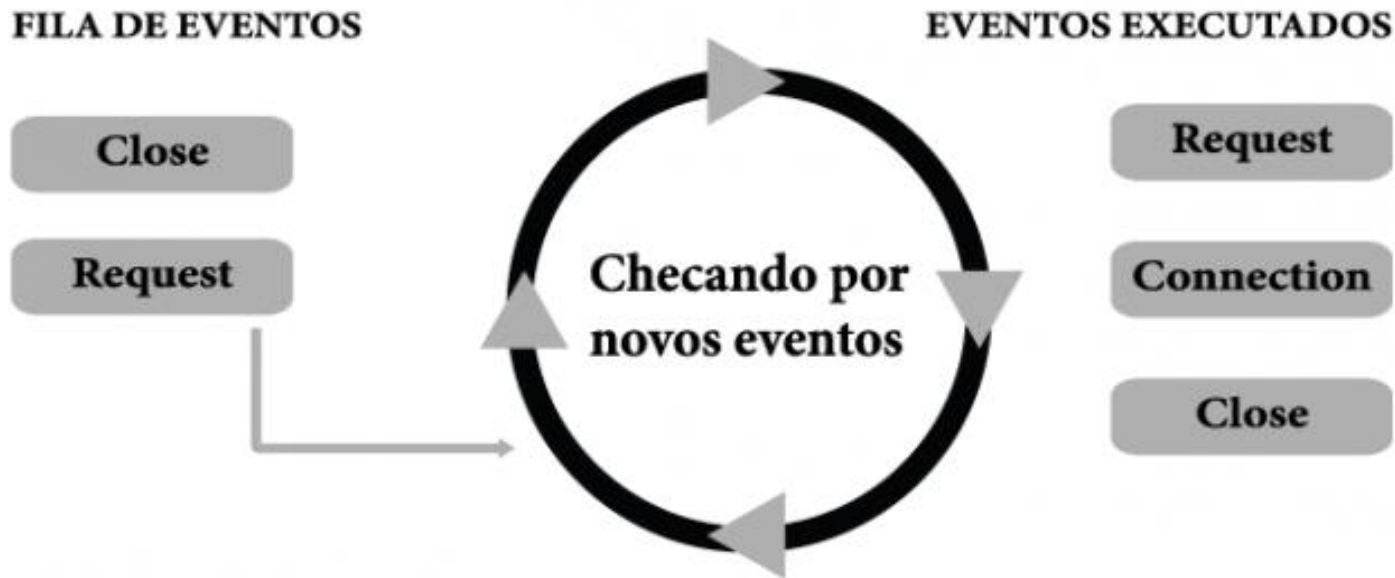
Aula 1.4. Node.js Event Loop

- ❑ Node.js Event Loop.

- O NodeJS é uma plataforma baseada em eventos. Isso significa que tudo que acontece no NodeJS é uma reação a um evento.
- Ele segue a mesma filosofia de orientação de eventos do JavaScript.
- Uma transação processada passa por várias callbacks.
- O NodeJS trabalha dessa forma porque operações de I/O e de rede são muito lentas.
- NodeJS trabalha com assincronismo, permitindo que seja desenvolvido uma aplicação orientada a eventos, graças ao Event Loop.

- O Event Loop basicamente é um loop infinito, que a cada iteração verifica se existem novos eventos em sua fila de eventos.
- O módulo responsável por emitir eventos é o EventEmitter.
- Quando um evento é emitido, ele é enviado para a fila de eventos, para que o Event Loop possa executá-lo e depois retornar seu callback.

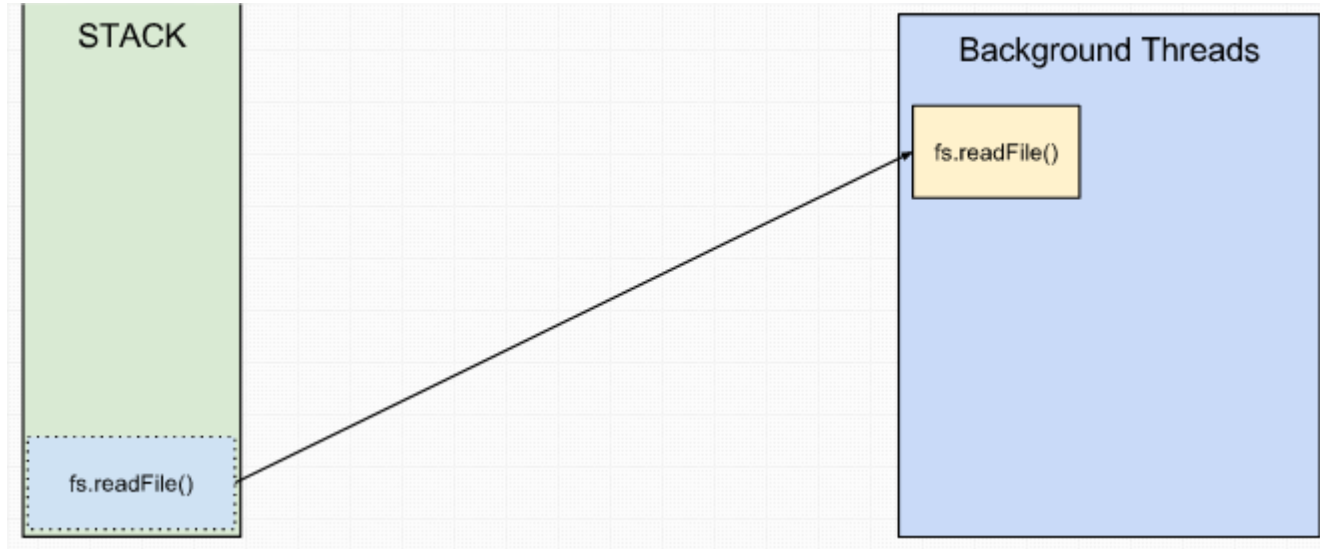
Node.js Event Loop



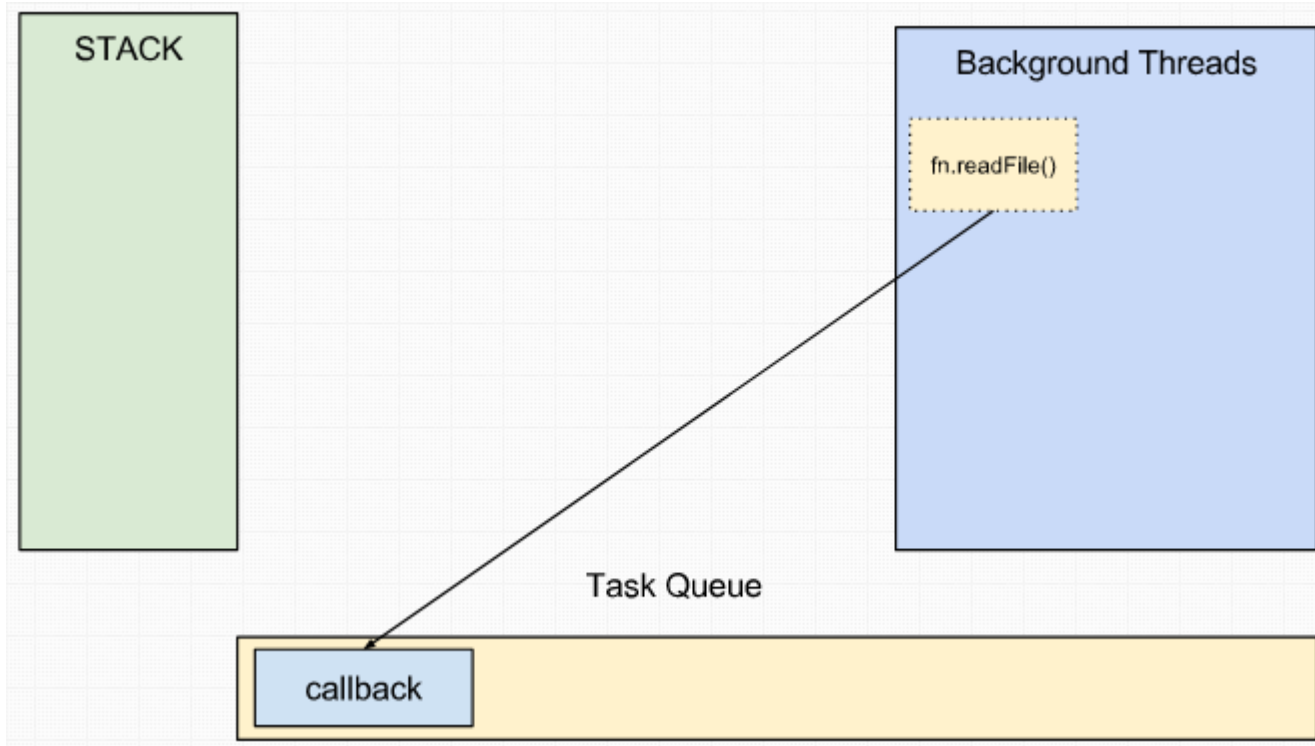
OS EVENTOS SÃO PROCESSADOS UM POR VEZ

- O V8, que é a base do Node.js, é single-threaded.
- Quando são executadas ações de I/O que demandaram tempo, o Node.js envia essas operações para outra thread do sistema.
- O Event Loop possui uma stack, e sempre que um método é chamado ele entra na stack para aguardar seu processamento.
- Após a outra thread do sistema executar a tarefa I/O, ele envia essa tarefa para a Task Queue.

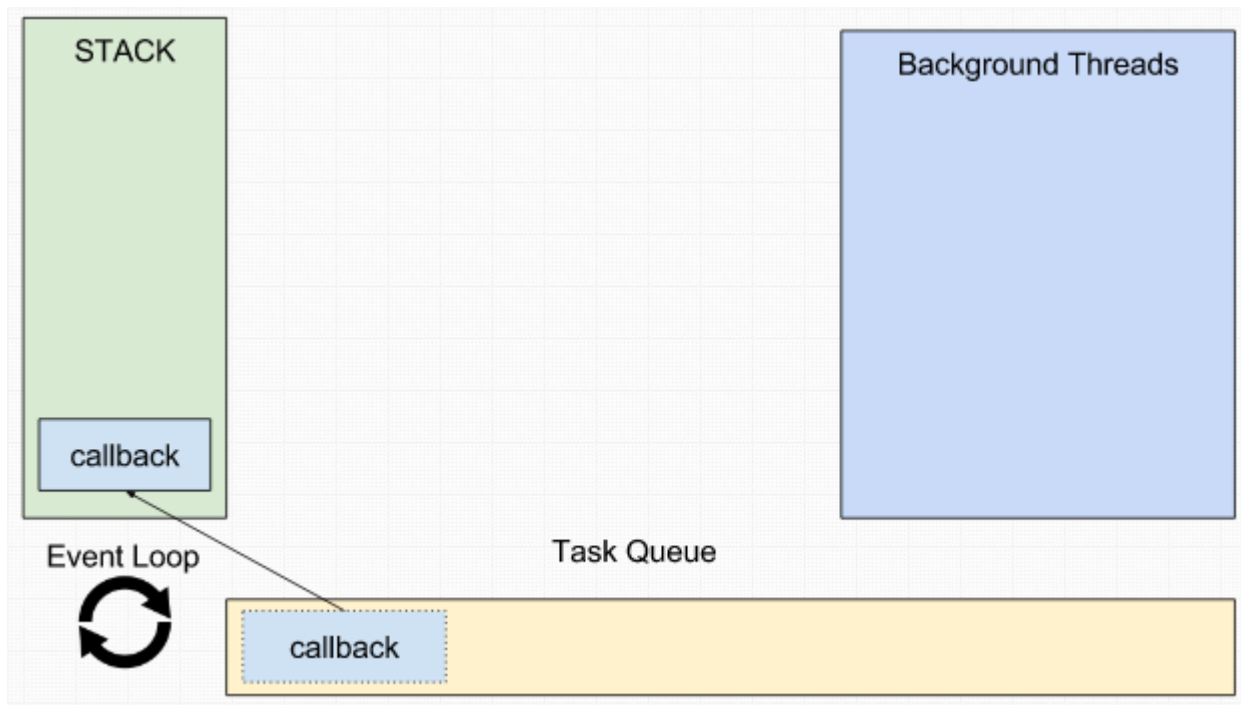
Node.js Event Loop



Node.js Event Loop

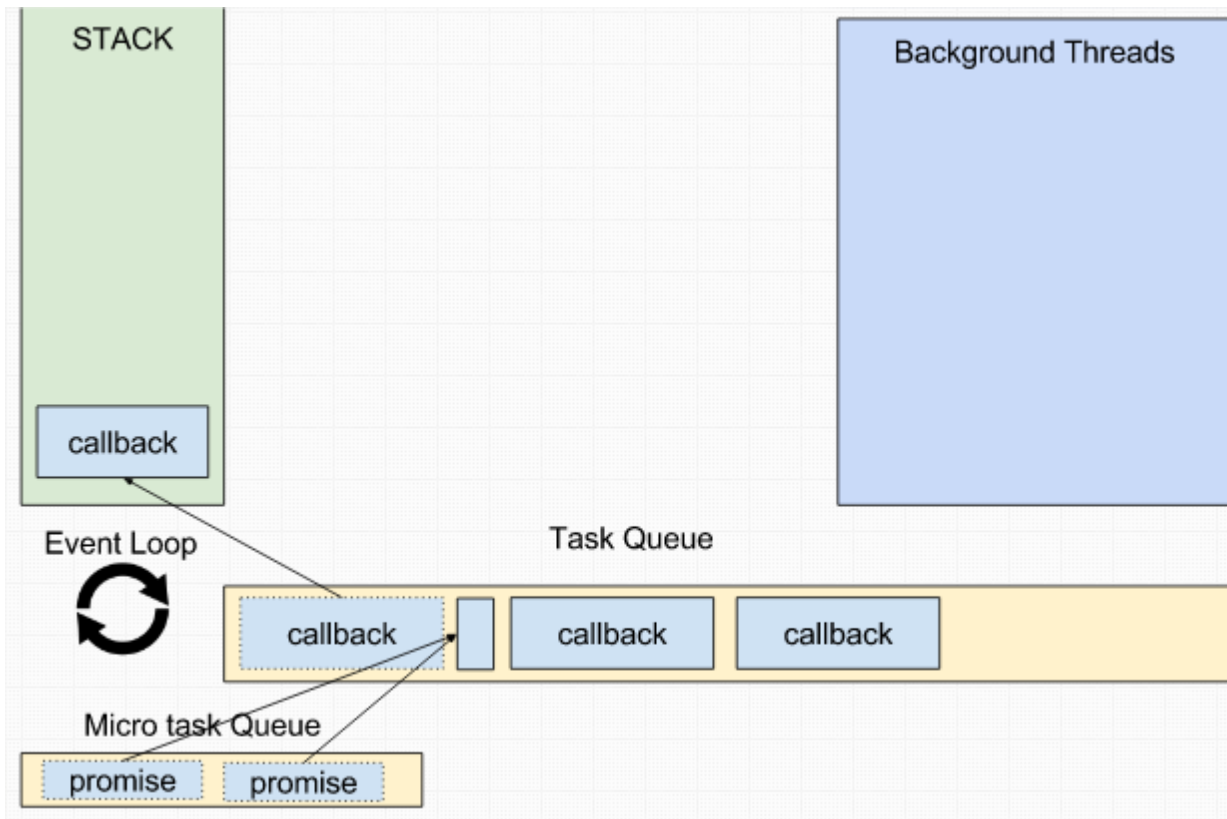


Node.js Event Loop



- Na Task Queue há dois tipos de tasks, as micro tasks e as macro tasks.
- Somente as macro tasks devem ser processadas em um ciclo do Event Loop.
- As micro tasks são tarefas que devem ser executadas rapidamente após alguma ação
- Após o Event Loop processar uma macro task da Task Queue, ele deve processar todas as micro tasks disponíveis antes de chamar outra macro task.

Node.js Event Loop



- ☑ NodeJS trabalha com assincronismo.
- ☑ Event Loop é quem coordena a execução das tasks e callbacks.
- ☑ Com isso o processador pode sempre trabalhar em sua máxima capacidade.

Próxima aula

☐ Módulos do Node.js.



Aula 1.5. Módulos do Node.js

Nesta aula

☐ Módulos do Node.js.

- Um módulo no Node.js é o mesmo que uma biblioteca no JavaScript.
- É um conjunto de funções que podem ser incluídas em uma aplicação.
- O Node.js segue o CommonJS, uma especificação de ecossistemas para o JavaScript.

- Assim é possível incluir um módulo que está em outro arquivo utilizando a função chamada `require`.
- É possível criar um módulo e importa-lo em outro arquivo facilmente.

```
JS test-module.js > ...  
1  exports.testFunction = function () {  
2    return "Test function done";  
3  };  
  
JS index.js > ...  
1  var test = require('./test-module');  
2  console.log(test.testFunction());
```

- O módulo nativo HTTP permite transferir dados através do protocolo HTTP.
- Este módulo consegue criar um servidor HTTP capaz de escutar portas do servidor e enviar respostas de volta ao cliente.

```
var http = require('http');
http.createServer(function (req, res) {
  if ((req.method === 'GET') && (req.url === '/test')) {
    res.write('GET /test');
  } else {
    res.write('Hello World!');
  }
  res.statusCode = 200;
  res.end();
}).listen(8080);
```

- O módulo nativo HTTP permite transferir dados através do protocolo HTTP.
- Este módulo consegue criar um servidor HTTP capaz de escutar portas do servidor e enviar respostas de volta ao cliente.

```
var http = require('http');
http.createServer(function (req, res) {
  if ((req.method === 'GET') && (req.url === '/test')) {
    res.write('GET /test');
  } else {
    res.write('Hello World!');
  }
  res.statusCode = 200;
  res.end();
}).listen(8080);
```

- File System permite trabalhar com arquivos, fazendo ações como ler, criar, atualizar, excluir e renomear arquivos.
- Possui os métodos `readFile()`, `appendFile()`, `writeFile()`, `unlink()` e `rename()`.

```
var fs = require('fs');
fs.readFile('./test-file.txt', 'utf-8', function (err, data) {
  if (err) {
    console.log(err.message);
  } else {
    console.log(data);
  }
});
```


- O módulo Events permite criar, disparar e escutar eventos.
- O objeto EventEmitter permite que sejam adicionadas call-backs para os eventos.
- Para disparar um evento basta chamar o método emit().

```
var events = require('events');  
var EventEmitter = new events.EventEmitter();  
  
eventEmitter.on('testEvent', function () {  
  console.log('Test event done');  
});  
  
eventEmitter.emit('testEvent');
```

- ☑ O NodeJS possui diversos módulos nativos além dos aqui citados, como por exemplo para enviar e-mails e fazer upload de arquivos.
- ☑ Auxiliam bastante no desenvolvimento por prover funcionalidades utilizadas frequentemente.

■ Próxima aula

□ NPM.



Aula 1.6. NPM

☐ NPM.

- NPM é a sigla para Node Package Manager, é o gerenciador de pacotes do Node.js.
- Ele é um repositório on-line para publicação de projetos de código aberto.
- Ferramenta de linha de comando auxilia na instalação de pacotes, gerenciamento de versão e gestão de dependências.

- Ele possui milhares de bibliotecas publicadas, algumas bem famosas e que são utilizadas por muitas aplicações.
- Adicionar biblioteca:
 - `npm install nome-da-biblioteca`
- Desinstalar biblioteca:
 - `npm uninstall nome-da-biblioteca`

- Grande partes dos frameworks JavaScript funcionam em conjunto com o NPM.
- Uma aplicação que utiliza o NPM como gerenciador de dependências deve possuir um arquivo chamado package.json.
- Ao lado do nome do pacote também é informado sua versão que será baixada.
- Instalar versão específica:
 - `npm install nome-da-biblioteca@1.5.3`

- A partir da versão 5 do NPM foi adicionado um arquivo chamado `package-lock.json`.
- Ele serve para garantir a consistência das dependências entre as máquinas.
- `package-lock.json` mantém a última versão que foi instalada, a localização do pacote e um código hash para verificar sua integridade.

- Geralmente, ao versionar os projetos no GIT ou SVN por exemplo, a pasta `node_modules` não é versionada.
- Ao instalar uma nova dependência no projeto ela é automaticamente incluída no `package.json`.

- name: nome do pacote.
- version: versão do pacote.
- description: descrição do pacote.
- homepage: site do pacote.
- author: nome do autor do pacote.
- repository: o tipo do repositório e a URL do pacote.
- main: o ponto de entrada do pacote.
- keywords: palavras chave do pacote.

```
{  
  "name": "hello-world",  
  "version": "1.0.0",  
  "description": "",  
  "main": "index.js",  
  "author": "",  
  "license": "ISC",  
  "dependencies": {  
    "express": "^4.17.1"  
  }  
}
```

- dependencies: lista de todas as dependências do pacote, que serão instaladas na pasta node_modules do projeto pelo comando npm install.
- devDependencies: dependências somente de desenvolvimento. Um pacote é listado aqui quando instalado com a flag --save-dev.

- ☑ NPM é muito utilizado pelas aplicações JavaScript.
- ☑ Provê fácil gerenciamento de dependências e instalação de pacotes.

☐ IDE de desenvolvimento.




Aula 1.7. IDE de desenvolvimento

Nesta aula

☐ IDE de desenvolvimento.

- Não existe uma IDE ou editor de texto específico que precisa ser utilizado.
- O desenvolvedor pode escolher o que preferir.
- O Visual Studio Code ou VS Code é um dos mais utilizados.
- Ferramenta open-source desenvolvida pela Microsoft.
- Possui vários puglins para facilitar o desenvolvimento.

IDE de desenvolvimento



Visual Studio Code

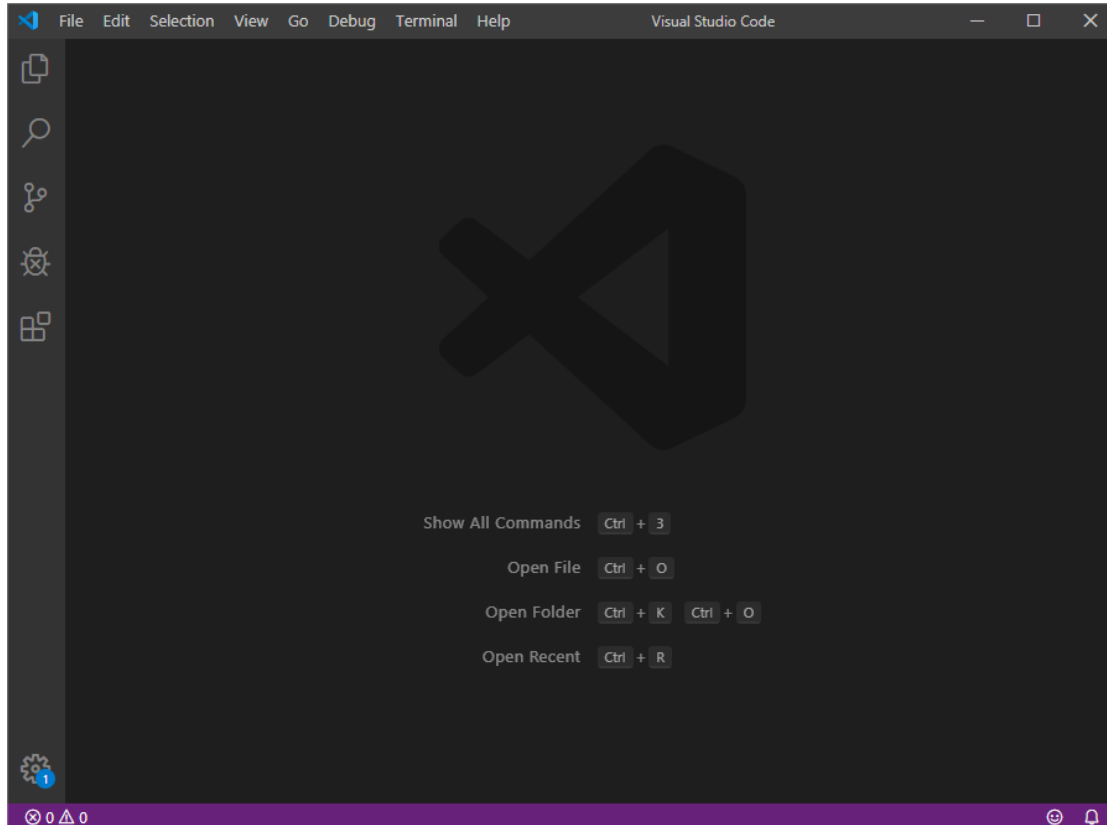


ATOM



Brackets

Visual Studio Code



☒ IDE de desenvolvimento.

- ❑ Ferramentas para consumo de endpoints.



Aula 1.8. Ferramentas para consumo de endpoints

- ☐ Ferramentas para consumo de endpoints.

- É comum ir testando os endpoints a medida que vão sendo desenvolvidos.
- Ferramentas para consumo de endpoints REST facilitam bastante o trabalho.
- Existem diversas ferramentas que tem este objetivo, nesta disciplina foi utilizada para testes a Insomnia.
- É possível organizar as requisições em pastas.



Ferramentas para consumo de endpoints

The screenshot displays the Insomnia application window titled "Insomnia (Grades Control) - grade". The interface includes a menu bar (Application, Edit, View, Window, Tools, Help) and a toolbar with "Send" and "Preview" buttons. The main area is divided into several panels:

- Left Panel:** Contains a "Filter" input, a folder named "grades-control-api", and a list of endpoints. The selected endpoint is "POST grade". Other endpoints include "PUT grade", "GET grade", "GET grade/:id", and "DEL grade/:id".
- Top Bar:** Shows the request method "POST", the URL "http://localhost:3000/grade", and the status "200 OK" with response time "93 ms" and size "37 B".
- JSON Panel:** Displays the request body as a JSON object:

```
1 {  
2   "student": "José Silva 6",  
3   "subject": "Matemática",  
4   "type": "Prova final",  
5   "value": 90.2  
6 }
```
- Preview Panel:** Displays the response body as a JSON object:

```
1 {  
2   "message": "Nota salva com sucesso!"  
3 }
```

- ☑ Ferramentas para consumo de endpoints.

☐ Capítulo 2 – Express.