**# Exercícios aula 1**

**Criação de rotas e métodos http (Rest API)**

* **Exercício 1 - Fazer uma pesquisa (resumo de 1 página) sobre a V8 JavaScript engine.**

V8 foi primeiro projetado para aumentar a performance de execução do Javascript dentro de navegadores web. A fim de obter velocidade. V8 é o interpretador JavaScript, também chamado de máquina virtual Javascript (ou engine), é o mecanismo JavaScript e WebAssembly de código aberto de alto desempenho do Google, escrito em C ++. É usado no Chrome e no Node.js, entre outros. O V8 pode ser executado de forma autônoma ou pode ser incorporado a qualquer aplicativo C ++.

A proposta do V8 é acelerar o desempenho de uma aplicação compilando o código Javascript para o formato nativo de máquina antes de executá-lo, lida com a alocação de memória para objetos e coleta objetos de que não precisa mais. O coletor de lixo preciso, geracional e de parar o mundo do V8 é uma das chaves para o desempenho do V8, permitindo que rode a velocidade de um código binário compilado.

A engine V8 também usa vários segmentos(threads) internamente:

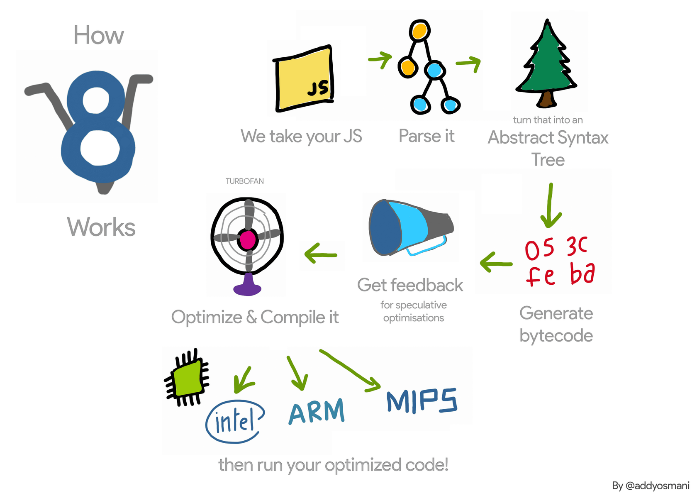
A principal threads faz o que espera: busca seu código, compila; e então executa.

Também há um thread separada para compilar, de modo que a thread principal pode se manter executando enquanto o código está sendo otimizado.

Um Profiler thread que vai contar ao runtime quais métodos nós gastamos mais tempo para que o Crankshaft possa otimizá-lo.

Algumas thread para lidar com varreduras de garbage collection(coleta de lixo)

Ao executar pela primeira vez o código Javascript, V8 aproveita o full-codegen que traduz diretamente o Javascript parseado em código de máquina sem qualquer transformação. Isso permite que ele comece a executar o código de máquina muito rápido. Note que a V8 não usa representação bytecode intermediário, dessa maneira removendo a necessidade por um interpretador.



* **Exercício 2 - Escolher 3 frameworks do Node.js e fazer um resumo**

Com a explosão da popularidade do NodeJS, foi criado o vasto número de estruturas de backend de JavaScript existentes hoje. Eles diferem em complexidade, recursos disponíveis ou padrões de arquitetura, segue alguns frameworks e ferramentas:

**Feathersjs:**

Feathers é uma estrutura da web leve para a criação de aplicativos em tempo real e APIs REST usando JavaScript ou TypeScript.

Feathers pode interagir com qualquer tecnologia de back-end, suporta mais de uma dúzia de bancos de dados e funciona com qualquer tecnologia de front-end como React, VueJS, Angular, React Native, Android ou iOS. Em sua essência, o Feathers é um conjunto de ferramentas e um padrão de arquitetura que facilita a criação de APIs REST escaláveis ​​e aplicativos em tempo real. Você pode construir protótipos em minutos e aplicativos prontos para produção em dias.

Os módulos individuais e APIs do Feathers. Existem três seções principais da API:

Core: a funcionalidade central do Feathers que pode ser usada no servidor e no cliente

Servidor: modula os módulos do lado do servidor usados ​​com o Core ao criar um servidor API em NodeJS

Cliente: Módulos usados ​​no cliente (NodeJS, navegador ou React Native) junto com o Core ao conectar a um servidor API Feathers.

**Meteor:**

Meteor é uma plataforma JavaScript full-stack para o desenvolvimento de aplicativos móveis e da web modernos. O Meteor inclui um conjunto-chave de tecnologias para a construção de aplicativos reativos de cliente conectado, uma ferramenta de construção e um conjunto curado de pacotes da comunidade Node.js e JavaScript em geral.

o framework cria uma “espécie de API”, cujo cliente monitora qualquer mudança, sendo que apenas dados trafegam de um lado ao outro. Quando algum dado é modificado no lado do cliente, essa mudança se reflete automaticamente no servidor e vice-versa. Além da enorme facilidade que isso traz ao trabalho do desenvolvedor, a aplicação fica bastante responsiva, pois não é mais necessário baixar a página inteira novamente e renderizar todo o conteúdo simplesmente porque uma pequena parte precisa ser atualizada. Nunca foi tão simples fazer AJAX.

O Meteor permite que você desenvolva em uma linguagem, JavaScript, em todos os ambientes: servidor de aplicativos, navegador da web e dispositivo móvel.

O Meteor usa dados na rede, o que significa que o servidor envia dados, não HTML, e o cliente os renderiza.

O Meteor abraça o ecossistema, trazendo as melhores partes da comunidade JavaScript extremamente ativa para você de uma forma cuidadosa e ponderada.

O Meteor fornece reatividade total da pilha, permitindo que sua IU reflita perfeitamente o verdadeiro estado do mundo com o mínimo de esforço de desenvolvimento.

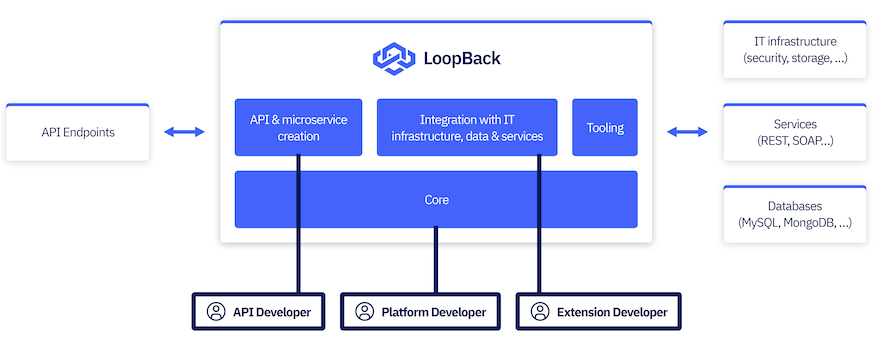
**LoopBack:**

O framework LoopBack é um conjunto de módulos em Node.js que podem ser usados de forma independente ou em conjunto para construir rapidamente APIs REST.

Uma aplicação LoopBack interage com fonte de dados por meio da API de modelo LoopBack, disponível localmente no Node.js, remotely over REST, e via cliente nativo APIs para iOS, Android e HTML5. Usando essas APIs, apps podem consultar bancos de dados, armazenar dados, upload de arquivos, enviar e-mails, criar notificações push, registrar usuários e executar outras ações fornecidas pelos serviços e fonte de dados.

Clientes podem chamar APIs LoopBack diretamente usando Strong Remoting, uma conexão automática da camada de transporte, permite que você forneça métodos back-end da API em REST, WebSockets, e outros transportes.

O diagrama abaixo demonstra como o LoopBack serve como uma ponte de composição entre as solicitações de entrada e as integrações de saída. Ele também mostra as diferentes personas que estão interessadas em vários recursos fornecidos pelo LoopBack.



* **Exercício 3 - Instalar e criar a aplicação hello world do node.js**

**(** [**https://github.com/Mileneapm/nodejs-up/tree/master/exercicio1**](https://github.com/Mileneapm/nodejs-up/tree/master/exercicio1) **)**

* **Exercício 4 - Fazer um resumo (1 página) sobre REST API.**

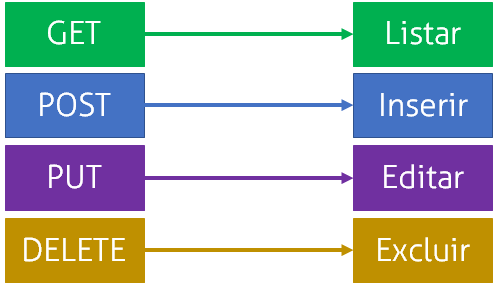
É uma interface de programação de aplicações (API ou API web) que está em conformidade com as restrições do estilo de arquitetura REST, permitindo a interação com serviços web RESTful.

Os benefícios deste modelo de API REST é que podemos servir múltiplos clientes com o mesmo back-end, ou seja, um único código fornecido para Web Mobile ou até mesmo uma API pública.

Em geral, a API torna o trabalho muito mais simples e fácil. Ele permite que o desenvolvedor integre a funcionalidade de serviços de terceiros, em vez de construí-los do zero. Por exemplo, Uber & Ola está usando o Google Map como sistema de navegação. Isso os ajuda a economizar tempo em vez de construir um sistema de navegação do zero.

**Diagrama

Descrição gerada automaticamente**

Serviços RESTful: verbos HTTP

Exemplo fluxo de requisição:

Requisição é feita por um cliente;

Resposta retornada através de uma estrutura de dados (ex: JSON);

Cliente recebe a resposta e processa o resultado.

Estas respostas utilizam métodos HTTP, que são:

* **GET**  http://minhaapi.com/*users*➔ Buscar alguma informação no back-end
* **POST**  http://minhaapi.com/*users*➔ Criar alguma informação no back-end
* **PUT**  http://minhaapi.com/*users*/1 ➔ Editar alguma informação no back-end
* **DELETE**  http://minhaapi.com/*users*/1 ➔ Deletar alguma informação no back-end

Negrito ➔ Método HTTP  
Itálico ➔ Recurso/Rota  
Número ➔ Parâmetro

[**https://github.com/Mileneapm/nodejs-up/tree/master/exercicio1e2**](https://github.com/Mileneapm/nodejs-up/tree/master/exercicio1e2)

* **Exercício 5 - Criar um CRUD rest API que suporte os métodos HTTP get, post, put e delete.**

**Criar uma aplicação específica. Ex: Controle de estacionamento, LanHouse etc.**

* **Exercício 6 - Evoluir o CRUD rest API para que seja possível cadastrar e recuperar informações de veículos (GET e POST) (Salvar a informações em Array)**

**Desafio é fazer o alterar e o deletar.**

**# Exercícios aula 2**

**Criação de rotas e métodos http (Rest API)**

[**https://github.com/Mileneapm/nodejs-up/tree/master/exercicio1e2**](https://github.com/Mileneapm/nodejs-up/tree/master/exercicio1e2)

* **Exercício 1 - Fazer uma pesquisa usando o filter.**
* **Exercício 2 - Usar o reduce para realizar um cálculo na aplicação (ex: número de horas gastas em serviços)**
* **Exercício 3 - Realizar a alteração de um item utilizando o map.**
* **Exercício 4 - Realizar a criação de um novo crud rest API.**

**Desafio: Realizar a exclusão de um determinado item por ID.**

**# Exercícios aula 3**

* **Exercício 1 - Realizar a criação de módulos por funcionalidade (CRUD)**
* **Exercício 2 - Criação do js de rotas**
* **Exercício 3 - Realizar uma pesquisa sobre os tipos de retorno http (status code)**

Os códigos de status das respostas HTTP indicam se uma requisição HTTP foi corretamente concluída.

Das cinco classes, segue os tipos de Status de Retorno mais comuns:

* [**Respostas informativas**](https://developer.mozilla.org/pt-BR/docs/Web/HTTP/Status#respostas_informativas) **(100-199):**

[100 Continue](https://developer.mozilla.org/pt-BR/docs/Web/HTTP/Status/100) - Essa resposta provisória indica que tudo ocorreu bem até agora e que o cliente deve continuar com a requisição ou ignorar se já concluiu o que gostaria.

[101 Switching Protocol](https://developer.mozilla.org/pt-BR/docs/Web/HTTP/Status/101) - Esse código é enviado em resposta a um cabeçalho de solicitação [Upgrade (en-US)](https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/Upgrade) pelo cliente, e indica o protocolo a que o servidor está alternando.

* [**Respostas de sucesso**](https://developer.mozilla.org/pt-BR/docs/Web/HTTP/Status#respostas_de_sucesso) **(200-299):**

200 OK - A requisição foi bem-sucedida.

201 Created - A solicitação foi atendida e um novo recurso foi criado.

[202 Accepted](https://developer.mozilla.org/pt-BR/docs/Web/HTTP/Status/202) - A requisição foi recebida, mas nenhuma ação foi tomada sobre ela

[203 Non-Authoritative Information](https://developer.mozilla.org/pt-BR/docs/Web/HTTP/Status/203) - Esse código de resposta significa que o conjunto de meta-informações retornadas não é o conjunto exato disponível no servidor de origem, mas coletado de uma cópia local ou de terceiros. Exceto essa condição, a resposta de 200 OK deve ser preferida em vez dessa resposta.

[204 No Content](https://developer.mozilla.org/pt-BR/docs/Web/HTTP/Status/204) - Não há conteúdo para enviar para esta solicitação, mas os cabeçalhos podem ser úteis. O user-agent pode atualizar seus cabeçalhos em cache para este recurso com os novos.

* [**Mensagens de redirecionamento**](https://developer.mozilla.org/pt-BR/docs/Web/HTTP/Status#mensagens_de_redirecionamento) **(300-399):**

[300 Multiple Choice](https://developer.mozilla.org/pt-BR/docs/Web/HTTP/Status/300);

[301 Moved Permanently](https://developer.mozilla.org/pt-BR/docs/Web/HTTP/Status/301);

[302 Found](https://developer.mozilla.org/pt-BR/docs/Web/HTTP/Status/302);

[303 See Other](https://developer.mozilla.org/pt-BR/docs/Web/HTTP/Status/303);

[304 Not Modified](https://developer.mozilla.org/pt-BR/docs/Web/HTTP/Status/304);

305 Use Proxy;

306 unused;

[307 Temporary Redirect](https://developer.mozilla.org/pt-BR/docs/Web/HTTP/Status/307);;

[308 Permanent Redirect](https://developer.mozilla.org/pt-BR/docs/Web/HTTP/Status/308);

* [**Respostas de erro do Cliente**](https://developer.mozilla.org/pt-BR/docs/Web/HTTP/Status#respostas_de_erro_do_cliente) **(400-499):**

400 - BAD REQUEST - A requisição não foi atendida pelo servidor devido à sintaxe incorreta.

[401 Unauthorized](https://developer.mozilla.org/pt-BR/docs/Web/HTTP/Status/401) - Embora o padrão HTTP especifique "unauthorized", semanticamente, essa resposta significa "unauthenticated". Ou seja, o cliente deve se autenticar para obter a resposta solicitada.

[402 Payment Required](https://developer.mozilla.org/pt-BR/docs/Web/HTTP/Status/402) - Este código de resposta está reservado para uso futuro. O objetivo inicial da criação deste código era usá-lo para sistemas digitais de pagamento, porém ele não está sendo usado atualmente.

[403 Forbidden](https://developer.mozilla.org/pt-BR/docs/Web/HTTP/Status/403) - O cliente não tem direitos de acesso ao conteúdo portanto o servidor está rejeitando dar a resposta. Diferente do código 401, aqui a identidade do cliente é conhecida.

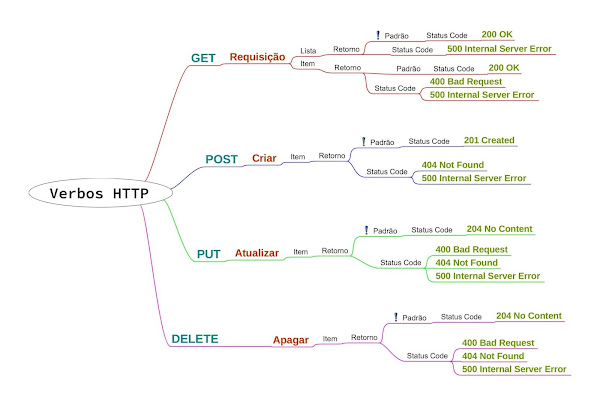
404 - NOT FOUND - O servidor não encontrou recurso da solicitação.

* [**Respostas de erro do Servidor**](https://developer.mozilla.org/pt-BR/docs/Web/HTTP/Status#respostas_de_erro_do_servidor) **(500-599):**

[500 Internal Server Error](https://developer.mozilla.org/pt-BR/docs/Web/HTTP/Status/500) - O servidor encontrou uma situação com a qual não sabe lidar.

[501 Not Implemented](https://developer.mozilla.org/pt-BR/docs/Web/HTTP/Status/501) - O método da requisição não é suportado pelo servidor e não pode ser manipulado. Os únicos métodos exigidos que servidores suportem (e, portanto, não devem retornar este código) são GET e HEAD.

[502 Bad Gateway](https://developer.mozilla.org/pt-BR/docs/Web/HTTP/Status/502) - Esta resposta de erro significa que o servidor, ao trabalhar como um gateway a fim de obter uma resposta necessária para manipular a requisição, obteve uma resposta inválida.



**# Exercícios aula 4**

* **Exercício 1 - Aplicar o mysql no projeto e criar as rotinas para salvar, alterar e listar nas funcionalidades.**
* **Exercício 2 - Realizar a criação da funcionalidade de excluir**
* **Exercício 3 - Realizar o exercício 1 e 2 para duas funcionalidades**
* **Exercício 4 - Modularizar a forma de conexão de uma maneira mais adequada.**
* **Exercício 5 - Formar dois grupos que deverão realizar uma pesquisa sobre o NPM e o YARN do Node.js. Realizar um debate para discutir as vantagens e desvantagens de cada um.**

**NPM** é o maior registro de software do mundo. Os desenvolvedores de código aberto de todos os continentes usam o npm para compartilhar e pegar pacotes emprestados, e muitas organizações também usam o npm para gerenciar o desenvolvimento privado.

O npm consiste em três componentes distintos:

* o site
* a interface de linha de comando (CLI)
* o registro

Use o site para descobrir pacotes, configurar perfis e gerenciar outros aspectos de sua experiência com o npm. Por exemplo, você pode configurar organizações para gerenciar o acesso a pacotes públicos ou privados.

A CLI é executada a partir de um terminal e é como a maioria dos desenvolvedores interage com o npm.

O registro é um grande banco de dados público de software JavaScript e as meta-informações que o cercam.

**Yarn** é um gerenciador de pacotes para o seu código. Ele permite que você use e compartilhe código com outros desenvolvedores de todo o mundo. O Yarn faz isso de forma rápida, segura e confiável para que você não precise se preocupar.

Yarn permite que você use soluções de outros desenvolvedores para problemas diferentes, tornando mais fácil para você desenvolver seu software. Se tiver problemas, você pode relatá-los ou contribuir de volta e, quando o problema for corrigido, você pode usar o Yarn para mantê-los atualizados.

O código é compartilhado por meio de algo chamado pacote. Um pacote contém todo o código que está sendo compartilhado, bem como um package.json arquivo (chamado de manifesto ) que descreve o pacote.

**Qual utilizar?**

O melhor de tudo é que tanto o npm quanto o Yarn utilizam o package.json, dando a você a escolha sobre qual melhor se adequa à sua necessidade.

**# Trabalho final**

**Estrutura pastas:**

**Interface gráfica do usuário, Texto, Aplicativo

Descrição gerada automaticamente**

*Node*, por ser *javascript*, tende a liberdade para estrutura, podendo usar da mais diversas, procurando sempre uma estrutura é legível e organizada.

De acordo que a aplicação vai crescendo, é necessário separar os diretórios. Um padrão comum em diversas linguagens é colocar o código da aplicação em um diretório *source* normalmente chamado *src, consorme imagem acima.*

Dessa maneira o código da aplicação é isolado em um diretório deixando o *root* mais limpo e acabando com a mistura de diretórios de código.

Dessa forma, a estrutura fornece maior facilidade ao incrementar novas funções, além de estar “separadinho” e bem definido na descrição.