#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](https://pt.wikipedia.org/wiki/Interpretador_JavaScript), também chamado de [máquina virtual](https://pt.wikipedia.org/wiki/M%C3%A1quina_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 várias segmentos(threads) internamente:

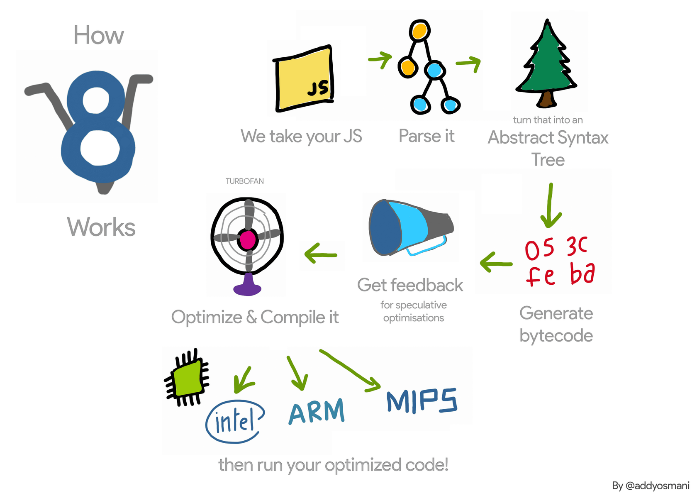
A principal threads faz o que espera: busca seu código, compila; e então executa.

Também há uma 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 (resumo de 1 página)

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.

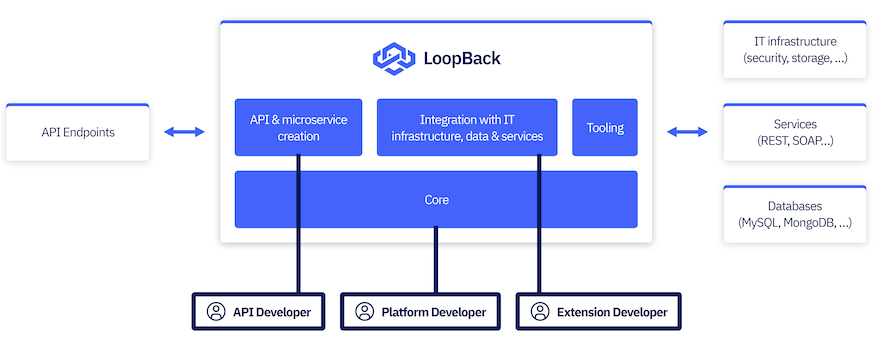
LoopBack:

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](https://loopback.io/doc/pt-br/lb2/Built-in-models-REST-API), e via cliente nativo APIs para [iOS, Android, e HTML5](https://loopback.io/doc/pt-br/lb2/Client-SDKs). 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](https://loopback.io/doc/pt-br/lb2/Strong-Remoting.html), 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 (Pasta exc-1.3)**
* **Exercício 4 - Fazer um resumo (1 página) sobre REST API.**

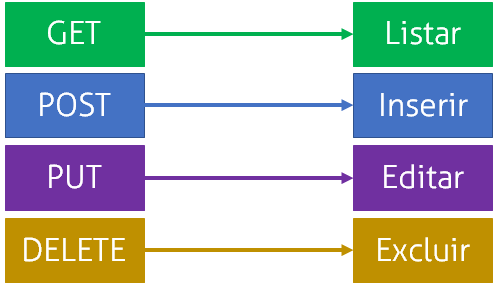
É uma interface de programação de aplicações (API ou API web) que está em conformidade com as restrições do estilo de arquitetura REST, permitindo a interação com serviços web RESTful.

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

* **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.**