Back end com NODEJS e EXPRESS – Samsung OCEAN

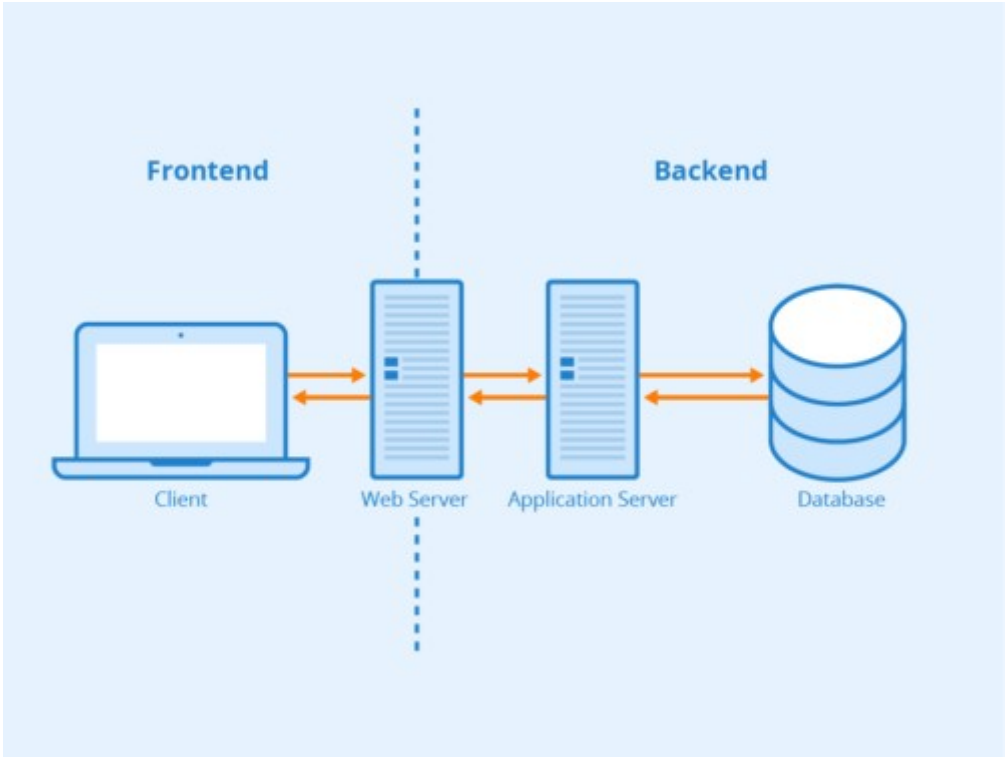
Desenvolver um CRUD com express

Node é um motor que processa o javascript no backend

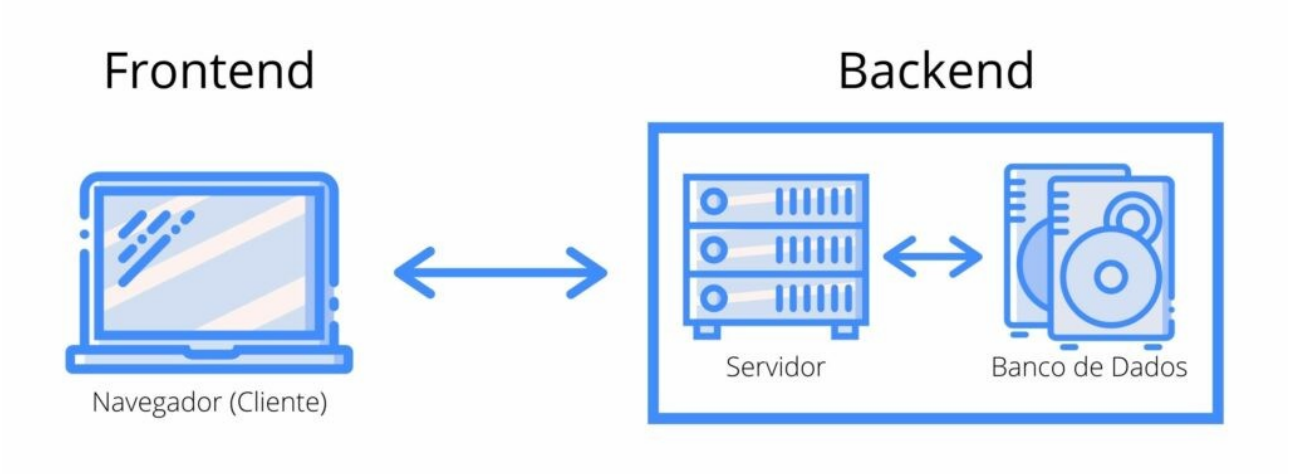
Backend está na camada do servidor na web que é acessada via IP

IP – Endereço da aplicação na internet

O backend tem foco na lógica do software e administra as formas de comunicação e armazenamento



http://bettatecnologia.com.br/blog/front-end-vs-back-end-entenda-as-diferencas/



mainframe = central de processamentos

Se comunicar com o servidor direto pelo terminal

Não tem interface gráfica no backend, usa-se terminal e editor de texto para os comandos e escrita dos códigos utilizados no back

Servidor com 2 camadas

Front end = é a interface gráfica que exibe os dados vindos do backend. Parte que o usuário interage com os dados

Lógica de apresentação – é o uso da lógica de programação para construir a regra de negócio. presentation logic

Linguagens de frontend – javascript(parte lógica- comportamento), HTML(estrutura) e css(estilo) que dão corpo as páginas web.

Javascript manipula a estrutura do HTML. Ex: reload da página

Linguagem de script interpretada no tempo de execução;

V8 – Engine que interpreta o Javascript e o executa. Feito em C++

Front end faz a requisição de dados no backend

Renderiza o site

Validação dos dados no front end é voltada para questões de usabilidade

Back end = recebe a requisição e fornece ao backend de forma estruturada; Serve o frontend. Acessa os dados, processa e manda para o front entd

É a camada de proteção do código, via uso de autenticação de usuários e senhas para comunicação com o banco de dados, API's e outros servidores.

Backend trabalha com os dados do front

Camada de segurança com o banco de dados

Lógica/regra de negócios – uso da lógica de programação para estruturar e organizar os dados em informações

Requisições HTTP (Hyper text transfer protocol)

Biblioteca x Framework

Biblioteca: conjunto de arquivos com a premissa de resolver um problema **especifico**

Framework: conjunto de diversas bibliotecas organizadas com o proposito de resolver questões de determinada implementação

Podem oferecer padrões de arquitetura

NojeJS: express

Python: Django, Flask

Exemplos de framewordk com API:

Uma camada

estão no mesmo servidor

faz a requisição no banco de dados e preenche o HTML com os dados

Duas camadas

Estão em outros servidores

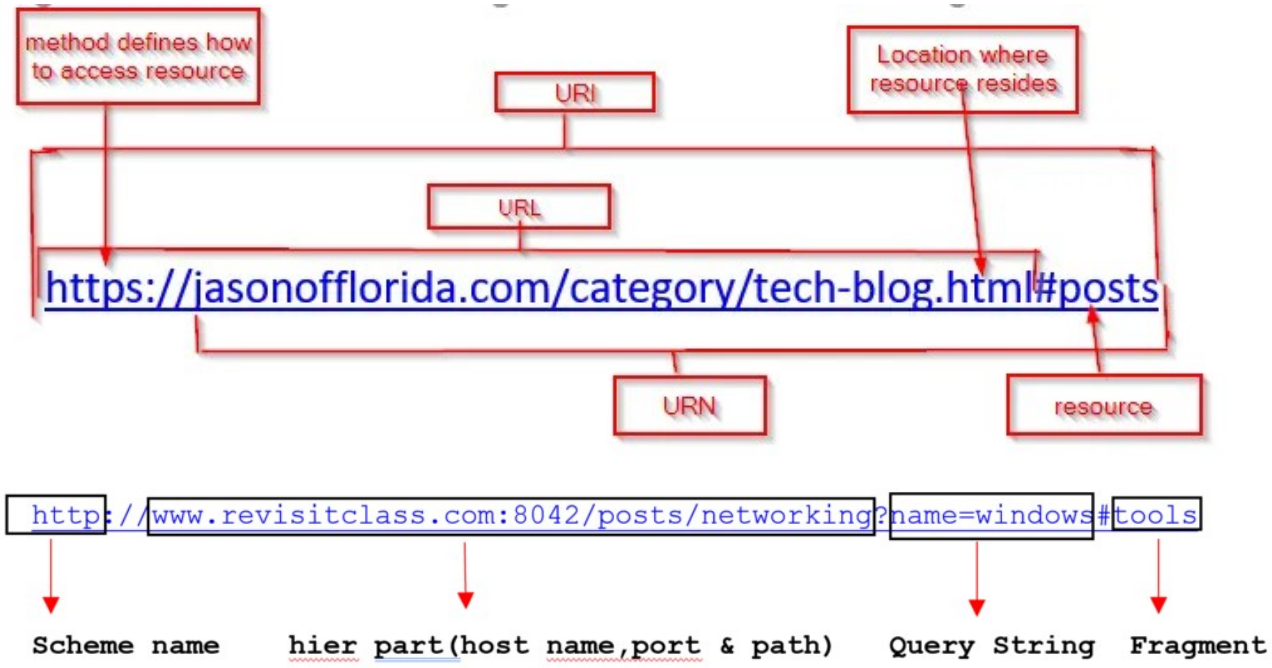
recebe o HTML com o javascript

devolve a estrutura dos dados

Dominio é uma mascara de um IP – localização da aplicação na internet

Protocolo de comunicação HTTP ou HTTPS:

Funciona através de uma URL



Verbos HTTP

Os tipos de verbos HTTP mais utilizados são:

- GET – solicitar informação;
- POST – Enviar informações novas;
- PUT – Atualizar informações existentes;
- DELETE – Remover informações.

➔ Body é o corpo da requisição e a forma que os dados são colocados num JSON como objetos javascript {}

➔

Rotas

As rotas são representadas pelas barras “/” que recebem chamadas de acordo com o verbo HTTP

Recebe a requisição e a resposta

Pode-se usar a mesma rota com verbos diferentes

Serialização

É o processo de enviar o dado de acordo coma estrutura que a linguagem aceita ou adapta-la.

Ex: json.Parse(variavel) → serializar o conteúdo da variável.

Pasta do projeto

Na pasta do projeto, digitar no terminal “npm init -y” para inicializar o repositório node que gera o arquivo package.json com os dados iniciais do projeto entre chaves e valores no objeto.

```
{
  "name": "API_JS_Ocean",
  "version": "1.0.0",
  "description": "",
  "main": "index.js",
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1"
  },
  "keywords": [],
  "author": "",
  "license": "ISC"
}
```

Express

O express é um framework web utilizado no node para aplicações backend

Instalar o express: npm install express | npm I express

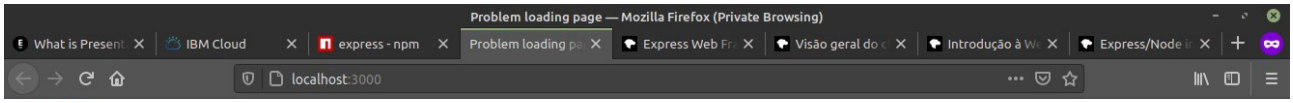
Gera a pasta node modules com todos os arquivos do framework

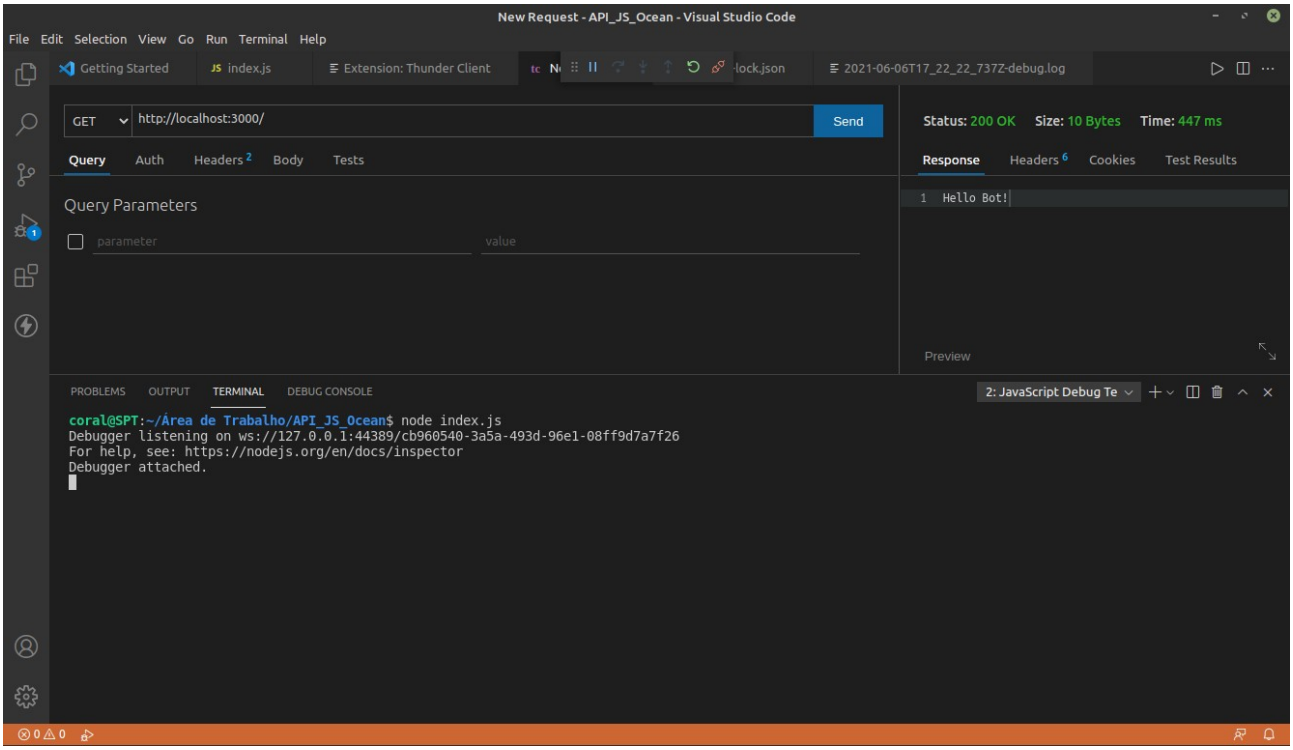
```
// importa o express
const express = require('express')
// chama o express para ser usado através da variável ‘app’
const app = express()
 // faz a requisição get no servidor e envia a mensagem de resposta contida no res.send()
app.get('/', function (req, res) {
  res.send('Hello World')
})
 // servidor usará a porta 3000
app.listen(3000)
```

Iniciar a aplicação: npm start

node index.js

Ver o servidor na URL: http://localhost:3000





a cada mudança no arquivo index.js é necessário reiniciar no terminal com o comando node index.js.

"scripts": {

"start": "node index.js",

"dev": "nodemoon"

}

nodemoon: biblioteca que atualiza a cada modificação na chave “dev” – Instalado via NPM dentro do ambiente de desenvolvimento do projeto

> npm i -D nodemon

Executar o script de dev com nodemon:

> npm run dev

Variável app

Toda vez que for se referir a aplicação com express usa-se a variável “app”.

Ex: app.get()

Escopo dos endpoints utilizados no projeto

- GET com a rota “/messages”
- GET – *messages*/{id} – retorna uma mensagem de id especifico
- POST – cria uma nova mensagem
- PUT *messages*{id} – atualiza a mensagem
- DELETE *messages*/{id} – deletar uma mensagem de ID especifico
- Toda requisição possui um header ou um body. - GET é mais comum ter a URL e o Headers
- Enviar a mensagem pelo body da mensagem
- Para capturar e exibir o corpo da requisição é necessário instalar a biblioteca “body-parser” do NPM(Dependência de execução do projeto)

Express – Biblioteca

// criar endpoint da rota POST

app.post(/endpoint, (req, res) => {

}

Para obter os dados das requisições post e persistir esses dados, faz-se necessário utilizar o body-parser, Caso contrário o valor retornado é undefined.