**BackEnd**

Iniciar projeto via NPM:

***(npm init -y)***

***FERRAMENTA Express*** para configuração de rotas!

(**npm install express**)

Nodemon é dependência que faz o servidor rodar automaticamente

(npm install **nodemon** -D)

Ao instalar é necessário ir na pasta do package.json alterar o teste e mudar para:

"scripts": {

    "start": "nodemon index.js"

  },

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Ao se instalar uma dependência, para usa-la no código, deve se a importar para pagina

Importação do express (configura as rotas automaticamente)

const express = require('express');

**const** é uma **variável** que se chama “express” e está recebendo **require** que é um importador e dentro colchetes tem o nome do pacote instalado.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Serviço: de portas http

const express = require('express');

const app = express(); // app agr tem poder do express

app.listen(3333); //aqui eu solicito que o express excute a porta 3333

1° const **express** requere a ferramenta **express** responsável pela configuração das rotas

2° const app recebe todas as funções da const **express**

3° app passa a escutar(listen), a porta a qual eu selecionar.

Serviço de rotas específicas e recursos

routes.get('/',(*request*, *response*) => {

    return response.json({

     evento: 'omnistak 11.0',

     Aluno: 'Alef Oliveira'

    });

});

app.listen(3333); //Rota raiz!

1° routes é uma variável que recebe todas as config do express.

2° routes.get pega(get) da rota raiz que seria ‘localhost:3333/’ antes da barra de recursos.

Os parâmetros definidos na função (req, res) {aqui viram as requisições e repostas} definindo os recursos solicitados após o ‘/’ por **quary parans** ou **routes paramns**

*Dentro da função iremos recupera sempre um Json pois é o método de comunicação entre o front e o backend.*

A Rota é o caminha completo, mas o que vem após a barra se chama recurso, o que eu estou querendo recuperar, pegar, incluir, acessar e etc.

**MÉTODOS HTTP**

**GET**

Serve quando queremos buscar uma informação do backend

Ex: retorno de informação, listagem e etc.

**POST**

Quando queremos criar uma informação no backend

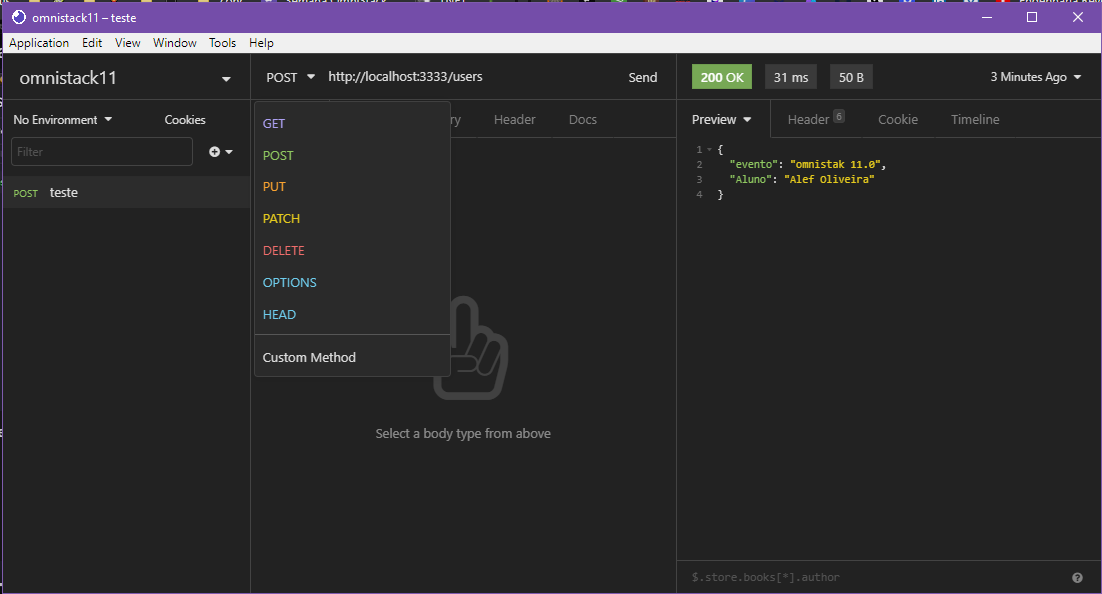
**PUT**

Altera informações no backend

**DELETE**

Remove informação no backend

O browser não aceita teste com outro método http se não o GET, então usamos o ISONMNIA que é um app que simula o browser parra requisições e podemos utilizar todos os tipos de Métodos HTTP.



**Teste de post no insomnia**

**TIPOS DE PARAMETROS** que podemos enviar para uma rota (**só request)**

**Query params**

Paramentos nomeados na rota após o (?) **servem para filtros nas buscar**

Ex: **no lado do cliente:**

3333/users**?**name=Alef (solicitar na rota usuários o usuário com o nome Alef)

**No servidor:**

routes.post('/users',(*request*, *response*) => {

    const params = request.query;

**Route params**

Paramentos utilizados para identificar recursos (é tudo após a /, onde identificar qual banco ou tabela buscar as informações)

Ex: no lado do cliente

...:3333/users/1 (Estou buscando no banco **Users** o usuário com ID=1)

**No servidor:**

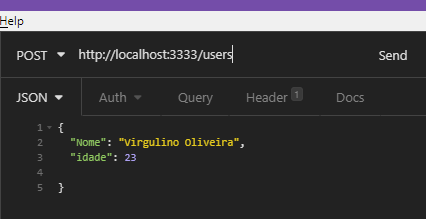
routes.get('/users/:id',

**Request body**

Corpo da requisição, utilizado para **criar** ou **alterar** um usuário

**EX:**

**Lado do Cliente enviando uma (Request) para o servidor**



**Lado do servidor:**

routes.post('/users',(*request*, *response*) => {

    const data = request.body;

*console*.log(body);

Neste momento a aplicação não vai entender que liguagem vc esta utilizando para passar as informações, como estamos usando o Json

Adiciona-se o seguinte código.

**Banco de Dados**

SQL e NoSQL

**Comunicação do Banco de Dados**

Driver: SELECT \* FROM users

Query Builder: teble(‘users’).select.(‘\*’).where()

Ferramenta **KNEX**

**Instalação (**npm install knex) após isso

Instala o banco de dados SQLite (npm install sqlite3)

Conecção com o banco de Dados

1° executar um pacote na raiz do projeto (npx knex intit) ele vai criar um arquivo

Na raiz do projeto para controlar todos os ambientes do projeto

Dentro do arquivo do knex configuramos para crie o diretório na pasta ‘rsc/database...’



**Pensamentos sobre aplicação**

**Entidades**

* ONG’s
* Casos(incidentes) Cada entidade uma tabela

**Funcionalidades**

* Login/logout de ong’s
* Cadastro de ong
* Cadastro de novos casos
* Deletar casos
* Listar casos específicos de uma ONG
* Listar todos os casos
* Entrar em contato com a ONG(wpp e e-mail)

**Migrations – knex – aprender!!!!**

**Code para criar migração**

(npx knex migrate:make creat\_ongs)

O nome creat\_ongs, é somente um exemplo.

Ele cria uma pasta e nesta pasta nos criamos a nossa tabela



Após isso executa com “npx**” para criar a tabela**

(npx knex migrate:latest)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Rotas e banco de dados**

Gerendo ID automático

**Dependencia do express crypto**

const crypto = require('crypto');

**código dentro da rota**

const id = crypto.randomBytes(4).toString('HEX');

**Método de descontrução antes e dps**

const data = request.body;

const { name, email, whatsapp, city, uf } = request.body;

**concecção com o banco de dados**

é feita uma pastar “connections” importa knex e as configurações do sqlite e cria

const connection = knex(configuration.development);

Após feita a conecção, importa a mesma para os códigos que precisam

Ex: da rota de cadastro de ong

routes.post('/ongs', async (*request*, *response*) => {

    const { name, email, whatsapp, city, uf } = request.body;

    const id = crypto.randomBytes(4).toString('HEX');

    await connection('ongs').insert({

        id,

        name,

        email,

        whatsapp,

        city,

        uf,

    });

    return response.json({ id });

});

Fez a conecção e selecionou a tabela ‘ongs’ e concatena com método insert para inserir os dados requeridos pelo lado do cliente.

Criou-se a pasta controllers onde a lógica das rotas fica lá!

Ex: rota POST

*module*.*exports* = {

    async create(*request*, *response*) {

      const { name, email, whatsapp, city, uf } = request.body;

      const id = crypto.randomBytes(4).toString('HEX');

      await connection('ongs').insert({

        id,

        name,

        email,

        whatsapp,

        city,

        uf,

      });

      return response.json({ id });

    }

};

**E Na pasta das rotas se utilizou o nome da função “create”**

**Ex:**

routes.post('/ongs', OngController.create);

**Login**

A função da rota de login é verificar se existe a Ong cadastrada no banco.

**FrontEnd**