Knex.js

Labenu_

dotenv

Labenu_

dotenv 

- O dotenv gerencia **variáveis de ambiente**
- Guarda informações **sensíveis** ou de **configuração**
- Esses dados ficam em um arquivo oculto chamado **.env** que deve ser criado na pasta raiz do projeto
- Colocamos o arquivo oculto **.env** dentro do **.gitignore** para os dados não serem enviados à nuvem

dotenv 

- Dentro do arquivo oculto **.env**, definimos a **chave** para cada uma das constantes junto com seu respectivo **valor**

```
PORT = 3003
DB_HOST = “35.226.146.116”
DB_USER = “turma-aluno”
DB_PASSWORD = “senha123”
DB_DATABASE = “turma-aluno”
```

dotenv 🏕️

- Precisamos chamar o método **dotenv.config()**
- A partir de então, as variáveis de ambiente poderão ser acessadas nos arquivos .ts

```
import dotenv from “dotenv”

dotenv.config()

console.log(process.env.NOME_DA_CHAVE)
```

Knex

Labenu_

Knex

- ***Knex*** é uma biblioteca de Javascript que permite fazer conexões com vários bancos SQL

- Definimos nas configurações um cliente de SQL, ou seja:
  - MySQL
  - PostgreSQL
  - MariaDB
  - etc...

Knex

- Instalamos o **knex** junto com o **mysql** como dependências de produção

- O knex não possui tipagens nativas, então precisamos instalar seu **@types/knex** como dependência de desenvolvimento

- Para configurar o knex precisamos criar uma conexão.
  Não precisa decorar a estrutura, sempre utilize o template

Knex

```
import knex from 'knex'
import dotenv from 'dotenv'

dotenv.config()

const connection = knex({  // Estabelece conexão com o banco
  client: "mysql",
  connection: {
      host: process.env.DB_HOST,
      port: 3306,
      user: process.env.DB_USER,
      password: process.env.DB_PASSWORD,
      database: process.env.DB_DATABASE,
      multipleStatements: true
  }
})

export default connection
```

Knex: raw 🥩

- O método **connection.raw()** nos permite criar e executar uma **query** no banco de dados utilizando a linguagem **SQL**
- Usamos **template strings** para montar as queries do ***raw***
- Agora podemos integrar nossa API com um banco de dados

Knex: raw 🥩

- Os callbacks dos endpoints agora serão **assíncronos**
- Toda função assíncrona devolve uma **Promise<>**
- Sempre que existir uma conexão com o banco de dados precisamos utilizar o **await**

Knex: raw - Coding together 🥩

- Criar um endpoint **GET Produtos**

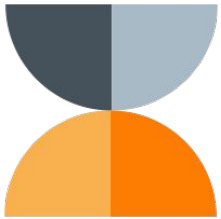
  query params ‘busca’ opcional

  caso não seja enviado, retornar todos os produtos

- Criar um endpoint **POST Produtos**

  body com ‘nome’, ‘preco’ e ‘categoria’ obrigatórios

  em caso de sucesso retorna os dados do produto cadastrado



Pausa para relaxar 😴

10 min

- O knex é uma biblioteca que permite fazer conexões com **bancos SQL**
- Como queremos conectar com um MySQL, devemos baixar o ***client* específico** dele para node
- O jeito mais direto de fazer queries é usando o método **raw** que devolve diretamente a resposta do banco

Knex: raw - Coding together 🥩

- Criar um endpoint **PUT Produtos** para **editar preço**

  body com ‘preco’ obrigatório

  em caso de sucesso retorna o status e mensagem

- Criar um endpoint **DELETE Produtos**

  em caso de sucesso retorna o status e mensagem

Knex: Query Builder

Labenu_

Knex: Query Builder 👷🏼‍♀️

- O ***Query Builder*** é uma funcionalidade do Knex que tende a **facilitar a criação das queries**

- Além da escrita, ele facilita na hora de tratar os dados

- Existem várias sintaxes relacionadas ao *Query Builder*, por isso é sempre bom confirmar na documentação se o que você está fazendo está certo

É isso por hoje!

Labenu_

Obrigado!