



PP II - PRISMA E HTTP

Semestre	4º Semestre
Tópicos Abordados	Códigos
Prioridade de Revisão	Baixa
Status de Compreensão	Compreendido
Matéria	UCE – Paradigmas de Programação II
Data da Aula	@11 de agosto de 2025
Professor	Daniel
Conteúdo	Explicação para github Luis Felipe Ferracine

Protocolo HTTP

O protocolo HTTP é utilizado em sistemas web, é o modelo de comunicação entre Cliente e Servidor. O protocolo consiste em dois lados, a pergunta (REQUEST) e a resposta (RESPONSE).

O Request é do Cliente para o Servidor, e o Response é do Servidor para o Cliente.

HTTP em Node.js

Para facilitar o uso do protocolo HTTP, usamos a Biblioteca express, já instalada anteriormente.

```
import express from 'express';
import cors from 'cors'
// importação de dependencias em nosso projeto

const app = express();
app.use(cors())
app.use(express.json());
```

Express

Para declarar rotas usando express, existem palavras reservadas que representam funções

- app.get()
- app.post()
- app.delete()
- app.put()
- app.listen()
- app.use()

Essas são as mais usadas, porem, existem outras

Declarando rotas

No protocolo HTTP, existe o conceito de **CRUD**.

- C - **Create** - Criar
- R - **Read** - Ler

- U - **Update** - Atualizar
- D - **Delete** - Deletar

```
// Declarando rotas no express

app.get("/get", (Request, Response) => {
  // .get representa "Read" no ceneito do CRUD
  // PARAMETROS DA FUNÇÃO
  // "/get" é o caminho da url que representa esse requisição
  // Request e Response são os modelos de comunicação entre Cliente e Servidor
})

app.post("/post", (Request, Response) => {
  // .post representa "Create" no ceneito do CRUD
  // PARAMETROS DA FUNÇÃO
  // "/post" é o caminho da url que representa esse requisição
  // Request e Response são os modelos de comunicação entre Cliente e Servidor
})

app.put("/put", (Request, Response) => {
  // .put representa "Update" no ceneito do CRUD
  // PARAMETROS DA FUNÇÃO
  // "/put" é o caminho da url que representa esse requisição
  // Request e Response são os modelos de comunicação entre Cliente e Servidor
})

app.delete("/delete", (Request, Response) => {
  // .delete representa "Delete" no ceneito do CRUD
  // PARAMETROS DA FUNÇÃO
  // "/delete" é o caminho da url que representa esse requisição
  // Request e Response são os modelos de comunicação entre Cliente e Servidor
})

app.listen(PORT, () => {
  // .listen representa em qual porta virtual, o servidor será aberto
})
```

Porta Virtual

conceito de porta virtual é muito utilizado não apenas na programação, mas em muitas áreas, como redes, etc.

No nosso caso, vamos usar sempre a porta 3000

Como estamos fazendo um servidor em nossa maquina, para testar as requisições da API, a URL fica da seguinte forma

http://localhost:3000

para utilizar as requisições, só colocar as rotas de casa

http://localhost:3000/get

Prisma

O prisma é um ORM que facilita fazer a conexão com nosso banco de dados.

Tabelas criadas pelo ORM

```
model TipoDeficiencia {
  id      Int      @id @default(autoincrement())
```

```

nome    String        @unique
subtipos SubtipoDeficiencia[]
createdAt DateTime     @default(now())
updatedAt DateTime     @updatedAt
}

model SubtipoDeficiencia {
  id      Int          @id @default(autoincrement())
  nome    String
  tipold  Int
  tipo    TipoDeficiencia @relation(fields: [tipold], references: [id], onDelete: Cascade)
  barreiras SubtipoBarreira[]
  createdAt DateTime    @default(now())
  updatedAt DateTime    @updatedAt

  @@unique([tipold, nome]) // evita subtipo duplicado dentro do mesmo tipo
}

model Barreira {
  id      Int          @id @default(autoincrement())
  descricao String      @unique
  subtipos SubtipoBarreira[]
  acessibilidades BarreiraAcessibilidade[]
  createdAt DateTime    @default(now())
  updatedAt DateTime    @updatedAt
}

model Acessibilidade {
  id      Int          @id @default(autoincrement())
  descricao String      @unique
  barreiras BarreiraAcessibilidade[]
  createdAt DateTime    @default(now())
  updatedAt DateTime    @updatedAt
}

model SubtipoBarreira {
  subtipold Int
  barreirald Int
  subtipo SubtipoDeficiencia @relation(fields: [subtipold], references: [id], onDelete: Cascade)
  barreira Barreira          @relation(fields: [barreirald], references: [id], onDelete: Cascade)

  @@id([subtipold, barreirald])
  @@index([barreirald])
}

model BarreiraAcessibilidade {
  barreirald Int
  acessibilidadeld Int
  barreira Barreira @relation(fields: [barreirald], references: [id], onDelete: Cascade)
  acessibilidade Acessibilidade @relation(fields: [acessibilidadeld], references: [id], onDelete: Cascade)

  @@id([barreirald, acessibilidadeld])
  @@index([acessibilidadeld])
}

```