**Estrutura sugerida**

src/

controllers/

tipos.controller.ts

subtipos.controller.ts

barreiras.controller.ts

acessibilidades.controller.ts

vinculos.controller.ts

repositories/

prisma.ts

tipos.repo.ts

subtipos.repo.ts

barreiras.repo.ts

acessibilidades.repo.ts

vinculos.repo.ts

routes/

tipos.routes.ts

subtipos.routes.ts

barreiras.routes.ts

acessibilidades.routes.ts

vinculos.routes.ts

index.ts

services/

tipos.service.ts

subtipos.service.ts

barreiras.service.ts

acessibilidades.service.ts

vinculos.service.ts

server.ts

**Controllers – falando com o Prisma**

**src/repositories/prisma.ts**

import { PrismaClient } from "@prisma/client";

const globalForPrisma = globalThis as unknown as { prisma?: PrismaClient };

export const prisma =

globalForPrisma.prisma ?? new PrismaClient({ log: ["warn", "error"] });

if (process.env.NODE\_ENV !== "production") globalForPrisma.prisma = prisma;

**src/repositories/tipos.repo.ts**

import { prisma } from "./prisma";

export const TiposRepo = {

list() {

return prisma.tipoDeficiencia.findMany({ orderBy: { id: "asc" } });

},

listWithSubtipos() {

return prisma.tipoDeficiencia.findMany({

orderBy: { id: "asc" },

include: { subtipos: { orderBy: { id: "asc" } } },

});

},

create(nome: string) {

return prisma.tipoDeficiencia.create({ data: { nome } });

},

findById(id: number) {

return prisma.tipoDeficiencia.findUnique({ where: { id } });

},

};

**src/repositories/subtipos.repo.ts**

import { prisma } from "./prisma";

export const SubtiposRepo = {

findById(id: number) {

return prisma.subtipoDeficiencia.findUnique({ where: { id } });

},

// usado pelo GET /subtipos/:id (com joins + ordenações)

findDeepById(id: number) {

return prisma.subtipoDeficiencia.findUnique({

where: { id },

include: {

tipo: true,

barreiras: {

include: {

barreira: {

include: {

acessibilidades: {

include: { acessibilidade: true },

orderBy: { acessibilidadeId: "asc" },

},

},

},

},

orderBy: { barreiraId: "asc" },

},

},

});

},

create(nome: string, tipoId: number) {

return prisma.subtipoDeficiencia.create({ data: { nome, tipoId } });

},

};

**src/repositories/barreiras.repo.ts**

import { prisma } from "./prisma";

export const BarreirasRepo = {

list() {

return prisma.barreira.findMany({ orderBy: { id: "asc" } });

},

create(descricao: string) {

return prisma.barreira.create({ data: { descricao } });

},

findById(id: number) {

return prisma.barreira.findUnique({ where: { id } });

},

// opcional: listar acessibilidades já diretamente pela barreira

listAcessibilidades(id: number) {

return prisma.barreira.findUnique({

where: { id },

include: {

acessibilidades: {

include: { acessibilidade: true },

orderBy: { acessibilidadeId: "asc" },

},

},

});

},

};

**src/repositories/acessibilidades.repo.ts**

import { prisma } from "./prisma";

export const AcessRepo = {

list() {

return prisma.acessibilidade.findMany({ orderBy: { id: "asc" } });

},

create(descricao: string) {

return prisma.acessibilidade.create({ data: { descricao } });

},

findById(id: number) {

return prisma.acessibilidade.findUnique({ where: { id } });

},

};

**src/repositories/vinculos.repo.ts**

import { prisma } from "./prisma";

export const VinculosRepo = {

vincularBarreirasSubtipo(subtipoId: number, barreiraIds: number[]) {

return prisma.subtipoBarreira.createMany({

data: barreiraIds.map((barreiraId) => ({ subtipoId, barreiraId })),

skipDuplicates: true,

});

},

vincularAcessBarreira(barreiraId: number, acessibilidadeIds: number[]) {

return prisma.barreiraAcessibilidade.createMany({

data: acessibilidadeIds.map((acessibilidadeId) => ({ barreiraId, acessibilidadeId })),

skipDuplicates: true,

});

},

};

**Services – regras / validações simples**

**src/services/tipos.service.ts**

import { TiposRepo } from "../repositories/tipos.repo";

export const TiposService = {

list() {

return TiposRepo.list();

},

listWithSubtipos() {

return TiposRepo.listWithSubtipos();

},

async create(nome: string) {

const final = (nome ?? "").trim();

if (!final) throw Object.assign(new Error("O campo 'nome' é obrigatório"), { status: 400 });

return TiposRepo.create(final);

},

};

**src/services/subtipos.service.ts**

import { SubtiposRepo } from "../repositories/subtipos.repo";

import { TiposRepo } from "../repositories/tipos.repo";

export const SubtiposService = {

async findDeep(id: number) {

const subtipo = await SubtiposRepo.findDeepById(id);

if (!subtipo) throw Object.assign(new Error("Subtipo não encontrado"), { status: 404 });

// “achata” a resposta como no server.ts

const barreiras = subtipo.barreiras.map((sb) => ({

id: sb.barreira.id,

descricao: sb.barreira.descricao,

acessibilidades: sb.barreira.acessibilidades.map((ba) => ({

id: ba.acessibilidade.id,

descricao: ba.acessibilidade.descricao,

})),

}));

return {

id: subtipo.id,

nome: subtipo.nome,

tipo: { id: subtipo.tipo.id, nome: subtipo.tipo.nome },

barreiras,

};

},

async create(nome: string, tipoId: number) {

const final = (nome ?? "").trim();

if (!final) throw Object.assign(new Error("Campos 'nome' e 'tipoId' são obrigatórios"), { status: 400 });

if (!Number.isInteger(tipoId)) throw Object.assign(new Error("tipoId inválido"), { status: 400 });

const tipo = await TiposRepo.findById(tipoId);

if (!tipo) throw Object.assign(new Error("Tipo não encontrado"), { status: 404 });

return SubtiposRepo.create(final, tipoId);

},

};

**src/services/barreiras.service.ts**

import { BarreirasRepo } from "../repositories/barreiras.repo";

export const BarreirasService = {

list() {

return BarreirasRepo.list();

},

async create(descricao: string) {

const final = (descricao ?? "").trim();

if (!final) throw Object.assign(new Error("O campo 'descricao' é obrigatório"), { status: 400 });

return BarreirasRepo.create(final);

},

};

**src/services/acessibilidades.service.ts**

import { AcessRepo } from "../repositories/acessibilidades.repo";

export const AcessService = {

list() {

return AcessRepo.list();

},

async create(descricao: string) {

const final = (descricao ?? "").trim();

if (!final) throw Object.assign(new Error("O campo 'descricao' é obrigatório"), { status: 400 });

return AcessRepo.create(final);

},

};

**src/services/vinculos.service.ts**

import { SubtiposRepo } from "../repositories/subtipos.repo";

import { BarreirasRepo } from "../repositories/barreiras.repo";

import { AcessRepo } from "../repositories/acessibilidades.repo";

import { VinculosRepo } from "../repositories/vinculos.repo";

export const VinculosService = {

async vincularBarreiras(subtipoId: number, barreiraIds: number[]) {

if (!Array.isArray(barreiraIds) || barreiraIds.length === 0) {

throw Object.assign(new Error("barreiraIds deve ser um array com pelo menos 1 id"), { status: 400 });

}

const subtipo = await SubtiposRepo.findById(subtipoId);

if (!subtipo) throw Object.assign(new Error("Subtipo não encontrado"), { status: 404 });

for (const id of barreiraIds) {

const b = await BarreirasRepo.findById(id);

if (!b) throw Object.assign(new Error(`Barreira ${id} não encontrada`), { status: 404 });

}

await VinculosRepo.vincularBarreirasSubtipo(subtipoId, barreiraIds);

return { ok: true };

},

async vincularAcessibilidades(barreiraId: number, acessibilidadeIds: number[]) {

if (!Array.isArray(acessibilidadeIds) || acessibilidadeIds.length === 0) {

throw Object.assign(new Error("acessibilidadeIds deve ser um array com pelo menos 1 id"), { status: 400 });

}

const barreira = await BarreirasRepo.findById(barreiraId);

if (!barreira) throw Object.assign(new Error("Barreira não encontrada"), { status: 404 });

for (const id of acessibilidadeIds) {

const a = await AcessRepo.findById(id);

if (!a) throw Object.assign(new Error(`Acessibilidade ${id} não encontrada`), { status: 404 });

}

await VinculosRepo.vincularAcessBarreira(barreiraId, acessibilidadeIds);

return { ok: true };

},

};

**Controllers (HTTP ↔ services)**

**src/controllers/tipos.controller.ts**

import { Request, Response } from "express";

import { TiposService } from "../services/tipos.service";

export const TiposController = {

async list(\_req: Request, res: Response) {

const data = await TiposService.list();

res.json(data);

},

async listWithSubtipos(\_req: Request, res: Response) {

const data = await TiposService.listWithSubtipos();

res.json(data);

},

async create(req: Request, res: Response) {

const { nome } = req.body ?? {};

const created = await TiposService.create(nome);

res.status(201).json(created);

},

};

**src/controllers/subtipos.controller.ts**

import { Request, Response } from "express";

import { SubtiposService } from "../services/subtipos.service";

export const SubtiposController = {

async getOne(req: Request, res: Response) {

const id = Number(req.params.id);

const data = await SubtiposService.findDeep(id);

res.json(data);

},

async create(req: Request, res: Response) {

const { nome, tipoId } = req.body ?? {};

const created = await SubtiposService.create(nome, Number(tipoId));

res.status(201).json(created);

},

};

**src/controllers/barreiras.controller.ts**

import { Request, Response } from "express";

import { BarreirasService } from "../services/barreiras.service";

export const BarreirasController = {

async list(\_req: Request, res: Response) {

const data = await BarreirasService.list();

res.json(data);

},

async create(req: Request, res: Response) {

const { descricao } = req.body ?? {};

const created = await BarreirasService.create(descricao);

res.status(201).json(created);

},

};

**src/controllers/acessibilidades.controller.ts**

import { Request, Response } from "express";

import { AcessService } from "../services/acessibilidades.service";

export const AcessibilidadesController = {

async list(\_req: Request, res: Response) {

const data = await AcessService.list();

res.json(data);

},

async create(req: Request, res: Response) {

const { descricao } = req.body ?? {};

const created = await AcessService.create(descricao);

res.status(201).json(created);

},

};

**src/controllers/vinculos.controller.ts**

import { Request, Response } from "express";

import { VinculosService } from "../services/vinculos.service";

export const VinculosController = {

async vincularBarreiras(req: Request, res: Response) {

const subtipoId = Number(req.params.id);

const { barreiraIds } = req.body ?? {};

const result = await VinculosService.vincularBarreiras(subtipoId, barreiraIds);

res.json(result);

},

async vincularAcessibilidades(req: Request, res: Response) {

const barreiraId = Number(req.params.id);

const { acessibilidadeIds } = req.body ?? {};

const result = await VinculosService.vincularAcessibilidades(barreiraId, acessibilidadeIds);

res.json(result);

},

};

Rotas

**src/routes/tipos.routes.ts**

import { Router } from "express";

import { TiposController } from "../controllers/tipos.controller";

const router = Router();

router.get("/", TiposController.list); // /tipos

router.get("/com-subtipos", TiposController.listWithSubtipos); // /tipos/com-subtipos

router.post("/", TiposController.create); // /tipos

export default router;

**src/routes/subtipos.routes.ts**

import { Router } from "express";

import { SubtiposController } from "../controllers/subtipos.controller";

const router = Router();

router.get("/:id", SubtiposController.getOne); // /subtipos/:id

router.post("/", SubtiposController.create); // /subtipos

export default router;

**src/routes/barreiras.routes.ts**

import { Router } from "express";

import { BarreirasController } from "../controllers/barreiras.controller";

const router = Router();

router.get("/", BarreirasController.list); // /barreiras

router.post("/", BarreirasController.create);

export default router;

**src/routes/acessibilidades.routes.ts**

import { Router } from "express";

import { AcessibilidadesController } from "../controllers/acessibilidades.controller";

const router = Router();

router.get("/", AcessibilidadesController.list); // /acessibilidades

router.post("/", AcessibilidadesController.create);

export default router;

**src/routes/vinculos.routes.ts**

import { Router } from "express";

import { VinculosController } from "../controllers/vinculos.controller";

const router = Router();

// POST /subtipos/:id/barreiras (N:N)

router.post("/subtipos/:id/barreiras", VinculosController.vincularBarreiras);

// POST /barreiras/:id/acessibilidades (N:N)

router.post("/barreiras/:id/acessibilidades", VinculosController.vincularAcessibilidades);

export default router;

**src/routes/index.ts**

import { Router } from "express";

import tipos from "./tipos.routes";

import subtipos from "./subtipos.routes";

import barreiras from "./barreiras.routes";

import acessibilidades from "./acessibilidades.routes";

import vinculos from "./vinculos.routes";

const router = Router();

router.use("/tipos", tipos);

router.use("/subtipos", subtipos);

router.use("/barreiras", barreiras);

router.use("/acessibilidades", acessibilidades);

router.use("/", vinculos); // rotas de vínculo

export default router;

// src/server.ts

import express from "express";

import { PrismaClient } from "@prisma/client";

// importa suas rotas

import tiposRoutes from "./routes/tipos.routes";

import subtiposRoutes from "./routes/subtipos.routes";

import barreirasRoutes from "./routes/barreiras.routes";

import acessibilidadesRoutes from "./routes/acessibilidades.routes";

const app = express();

const prisma = new PrismaClient();

app.use(express.json());

// usa os módulos de rotas

app.use("/tipos", tiposRoutes);

app.use("/subtipos", subtiposRoutes);

app.use("/barreiras", barreirasRoutes);

app.use("/acessibilidades", acessibilidadesRoutes);

// middleware de erro genérico

app.use((err: any, \_req: any, res: any, \_next: any) => {

  console.error(err);

  res.status(err.status || 500).json({ error: err.message || "Erro interno" });

});

// sobe o servidor

const PORT = process.env.PORT || 3000;

app.listen(PORT, () => {

  console.log(`API rodando em http://localhost:${PORT}`);

});

**O que mapeia cada endpoint que você tinha**

* GET /tipos → routes/tipos.routes.ts → TiposController.list() → TiposService.list() → TiposRepo.list()
* GET /subtipos (tipos com subtipos) → agora GET /tipos/com-subtipos
* GET /barreiras → BarreirasController.list()
* GET /acessibilidades → AcessibilidadesController.list()
* GET /subtipos/:id (deep) → SubtiposController.getOne() → SubtiposService.findDeep()
* POST /tipos → TiposController.create()
* POST /subtipos → SubtiposController.create()
* POST /barreiras → BarreirasController.create()
* POST /acessibilidades → AcessibilidadesController.create()
* POST /subtipos/:id/barreiras → VinculosController.vincularBarreiras()
* POST /barreiras/:id/acessibilidades → VinculosController.vincularAcessibilidades()