Passo 1. criar a pasta estoque em Documentos

Passo 2. criar a pasta backend em estoque

Passo 3. na pasta backend, digite os comandos abaixo:

Passo 4. npm init -y

Passo 5. npm install typescript -D

Passo 6. npx tsc --init

Passo 7. npm install tsx -D

Passo 8. crie arquivo src/server.ts

Passo 9. no package.json, alterar test para dev: tsx watch src/server.ts

Passo 10. crie arquivo src/routes.ts

Passo 11. npm install fastify

Passo 12. altere o arquivo routes.ts para

import { FastifyInstance } from 'fastify'

export async function AppRoutes(app: FastifyInstance) {

}

Passo 13. npm install @fastify/cors

Passo 14. altere o arquivo server.ts para

import Fastify from 'fastify'

import cors from '@fastify/cors'

import { AppRoutes } from './routes'

// instanciar o objeto da classe Fastify

const app = Fastify()

// registra o uso do cors

app.register(cors) // permite que qualquer IP utilize as rotas

// registra todas as rotas no servidor HTTP

app.register(AppRoutes)

// vamos subir o servidor, vamos executá-lo, ele ficará ouvindo

// e aguardando as requisições

app.listen({

port: 3333

})

.then(() => {

console.log('HTTP server running and listening requests')

})

Passo 15. npm install prisma -D

Passo 16. npm install @prisma/client

Passo 17. npx prisma init --datasource-provider SQLite

Passo 18. alterar o arquivo prisma/schema.prisma para

generator client {

provider = "prisma-client-js"

}

datasource db {

provider = "sqlite"

url = env("DATABASE\_URL")

}

model User {

id Int @id @default(autoincrement())

username String

password String

email String

created\_at DateTime

products Product[]

controls Control[]

@@map("user")

}

model Product {

id Int @id @default(autoincrement())

name String

description String

quantity Float

price Float

created\_at DateTime

userId Int

user User @relation(fields: [userId], references: [id])

controls Control[]

@@map("product")

}

model Control {

id Int @id @default(autoincrement())

type String

quantity Float

price Float

created\_at DateTime

userId Int

user User @relation(fields: [userId], references: [id])

productId Int

product Product @relation(fields: [productId], references: [id])

@@map("control")

}

Passo 19. npx prisma migrate dev

Passo 20. coloque o nome da migração como product, user and control

Passo 21. npm install zod

Passo 22. crie o arquivo src/lib/prisma.ts com o conteúdo

// importa o prisma

import { PrismaClient } from '@prisma/client'

// instanciar o objeto da classe PrimaClient

export const prisma = new PrismaClient()

Passo 23. npm install dayjs

Passo 24. abrir o arquivo src/routes.ts e insira:

import { FastifyInstance } from 'fastify'

import {z} from 'zod'

import {prisma} from './lib/prisma'

import dayjs from 'dayjs'

export async function AppRoutes(app: FastifyInstance) {

// rota para criar um user

app.post('/user', async (request) => {

const postBody = z.object({

username: z.string(),

password: z.string(),

email: z.string()

})

const {username, password, email} = postBody.parse(request.body)

const created\_at = dayjs().startOf('day').toDate() // sem hora, minuto e segundo

const newUser = await prisma.user.create({

data: {

username,

password,

email,

created\_at

}

})

return newUser

})

// rota para recuperar um user

app.get('/user/:username/:password', async (request) => {

const getParam = z.object({

username: z.string(),

password: z.string(),

})

const {username, password } = getParam.parse(request.params)

const user = await prisma.user.findMany({

where: {

username: username,

password: password

}

})

return user

})

}