Roteiro da aula 2

1. Alterar o arquivo schema. prisma, adicionando as tabelas:

model Day {

id String @id @default(uuid())

date DateTime

@@map("days")

@@unique([date])

}

model DayHabit {

id String @id @default(uuid())

day\_id String

habit\_id String

@@map("day\_habits")

@@unique([day\_id, habit\_id])

}

1. Alterar o arquivo schema. prisma, adicionando as tabelas:

model HabitWeekDay {

id String @id @default(uuid())

habit\_id String

week\_day Int

@@unique([habit\_id, week\_day])

@@map("habit\_week\_days")

}

1. npx prisma migrate dev
2. alterar o arquivo schema.prisma, adicionando os relacionamentos entre as tabelas:

model DayHabit {

id String @id @default(uuid())

day\_id String

habit\_id String

day Day @relation(fields: [day\_id], references: [id])

habit Habit @relation(fields: [habit\_id], references: [id])

@@unique([day\_id, habit\_id])

@@map("day\_habits")

}

model Day {

id String @id @default(uuid())

date DateTime

dayHabits DayHabit[]

@@unique([date])

@@map("days")

}

model HabitWeekDay {

id String @id @default(uuid())

habit\_id String

week\_day Int

habit Habit @relation(fields:[habit\_id], references: [id])

@@unique([habit\_id, week\_day])

@@map("habit\_week\_days")

}

model Habit {

id String @id @default(uuid())

title String

created\_at DateTime

dayHabits DayHabit[]

weekDays HabitWeekDay[]

@@map("habits")

}

1. npx prisma migrate dev
2. npm i -D prisma-erd-generator @mermaid-js/mermaid-cli
3. altere o arquivo schema.prisma com:

generator erd {

provider = "prisma-erd-generator"

}

1. npx prisma generate
2. na pasta prisma, crie um arquivo seed.ts

import { PrismaClient } from '@prisma/client'

const prisma = new PrismaClient()

const firstHabitId = '0730ffac-d039-4194-9571-01aa2aa0efbd'

const firstHabitCreationDate = new Date('2022-12-31T03:00:00.000')

const secondHabitId = '00880d75-a933-4fef-94ab-e05744435297'

const secondHabitCreationDate = new Date('2023-01-03T03:00:00.000')

const thirdHabitId = 'fa1a1bcf-3d87-4626-8c0d-d7fd1255ac00'

const thirdHabitCreationDate = new Date('2023-01-08T03:00:00.000')

async function run() {

await prisma.habit.deleteMany()

await prisma.day.deleteMany()

/\*\*

\* Create habits

\*/

await Promise.all([

prisma.habit.create({

data: {

id: firstHabitId,

title: 'Beber 2L água',

created\_at: firstHabitCreationDate,

weekDays: {

create: [

{ week\_day: 1 },

{ week\_day: 2 },

{ week\_day: 3 },

]

}

}

}),

prisma.habit.create({

data: {

id: secondHabitId,

title: 'Exercitar',

created\_at: secondHabitCreationDate,

weekDays: {

create: [

{ week\_day: 3 },

{ week\_day: 4 },

{ week\_day: 5 },

]

}

}

}),

prisma.habit.create({

data: {

id: thirdHabitId,

title: 'Dormir 8h',

created\_at: thirdHabitCreationDate,

weekDays: {

create: [

{ week\_day: 1 },

{ week\_day: 2 },

{ week\_day: 3 },

{ week\_day: 4 },

{ week\_day: 5 },

]

}

}

})

])

await Promise.all([

/\*\*

\* Habits (Complete/Available): 1/1

\*/

prisma.day.create({

data: {

/\*\* Monday \*/

date: new Date('2023-01-02T03:00:00.000z'),

dayHabits: {

create: {

habit\_id: firstHabitId,

}

}

}

}),

/\*\*

\* Habits (Complete/Available): 1/1

\*/

prisma.day.create({

data: {

/\*\* Friday \*/

date: new Date('2023-01-06T03:00:00.000z'),

dayHabits: {

create: {

habit\_id: firstHabitId,

}

}

}

}),

/\*\*

\* Habits (Complete/Available): 2/2

\*/

prisma.day.create({

data: {

/\*\* Wednesday \*/

date: new Date('2023-01-04T03:00:00.000z'),

dayHabits: {

create: [

{ habit\_id: firstHabitId },

{ habit\_id: secondHabitId },

]

}

}

}),

])

}

run()

.then(async () => {

await prisma.$disconnect()

})

.catch(async (e) => {

console.error(e)

await prisma.$disconnect()

process.exit(1)

})

1. Atualizar o arquivo package.json com

"prisma": {

"seed": "tsx prisma/seed.ts"

}

1. npx prisma db seed
2. npx prisma studio
3. na pasta src, crie uma pasta lib
4. crie o arquivo prisma.ts com o conteúdo

import { PrismaClient } from '@prisma/client'

export const prisma = new PrismaClient()

1. crie o arquivo routes.ts na pasta src

import {FastifyInstance} from 'fastify'

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

export async function AppRoutes(app: FastifyInstance){

app.get('/hello2', async () => {

const habits = await prisma.habit.findMany({

where: {

title: {

startsWith: 'beber'

}

}

})

return habits

})}

1. alterar o arquivo src/server.ts como:

import Fastify from 'fastify'

import cors from '@fastify/cors'

import { AppRoutes } from './routes'

const app = Fastify()

app.register(cors)

app.register(AppRoutes)

app.listen({

port: 3333,

})

.then( () => {

console.log('Http Server running')

})

1. npm install zod
2. npm install dayjs
3. alterar arquivo routes.ts, adicionando

import dayjs from "dayjs"

import { z } from "zod"

1. alterar arquivo routes.ts, adicionando

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

const createHabitBody = z.object({

title: z.string(),

weekDays: z.array(

z.number().min(0).max(6)

)

})

const { title, weekDays } = createHabitBody.parse(request.body)

const today = dayjs().startOf('day').toDate()

await prisma.habit.create({

data: {

title,

created\_at: today,

weekDays: {

create: weekDays.map(weekDay => {

return {

week\_day: weekDay

}

})

}

}

})

})

1. alterar arquivo routes.ts, adicionando

app.get('/day', async (request) => {

const getDayParams = z.object({

date: z.coerce.date() // converte string para date

})

const {date} = getDayParams.parse(request.query)

const parsedDate = dayjs(date).startOf('day')

const weekDay = parsedDate.get('day')

// todos hábitos possívels

const possibleHabits = await prisma.habit.findMany({

where: {

created\_at:{

lte: date

},

weekDays: {

some: {

week\_day: weekDay

}

}

}

})

// todos hábitos realizados

const day = await prisma.day.findUnique({

where: {

date: parsedDate.toDate()

},

include: {

dayHabits: true

}

})

const completedHabit = day?.dayHabits.map(dayHabit => {

return dayHabit.habit\_id

})

return {

possibleHabits,

completedHabit

}

})

**FAZENDO AGORA O FRONTEND**

1. alterar aquivo tailwind.config.cjs

theme: {

extend: {

colors: {

background: '#09090A'

},

1. alterar arquivo index.html

<body class="bg-background text-white" >

<div id="root"></div>

<script type="module" src="/src/main.tsx"></script>

</body>

1. alterar arquivo App.tsx

export function App() {

return (

)

}

1. alterar arquivo main.tsx

import React from 'react'

import ReactDOM from 'react-dom/client'

import { App } from './App'

ReactDOM.createRoot(document.getElementById('root') as HTMLElement).render(

<React.StrictMode>

<App />

</React.StrictMode>,

)

1. Alterar arquivo App.tsx

export function App() {

return (

<div className="w-screen h-screen flex justify-center items-center">

<div className="w-full max-w-5xl px-6 flex flex-col gap-16">

</div>

</div>

)

}

1. Adicionar a dependência Phosphor

npm install phosphor-react

1. Criar arquivo Header.tsx

import { Plus } from "phosphor-react";

import LogoImage from '../assets/logo.svg'

export function Header() {

return (

<div className="w-full max-w-3xl mx-auto flex items-center justify-between">

<img src={LogoImage} alt="" />

<button type="button"

className="border border-violet-500 font-semibold rounded-lg px-6 py-4 flex items-center gap-3 hover:border-violet-300">

<Plus size={20} className="text-violet-500" />

Novo hábito

</button>

</div>

);

}

1. Alterar arquivo App.tsx

Alterar arquivo App.tsx

import { Header } from "./components/Header";

import { SummaryTable } from "./components/SummaryTable";

export function App() {

return (

<div className="w-screen h-screen flex justify-center items-center">

<div className="w-full max-w-5xl px-6 flex flex-col gap-16">

<Header/>

<SummaryTable/>

</div>

</div>

)

}

1. Alterar arquivo tailwind.config.cjs

theme: {

extend: {

colors: {

background: '#09090A'

},

gridTemplateRows: {

7: 'repeat(7, minmax(0, 1fr))',

}

},

},

1. Criar arquivo SummaryTable.tsx

const weekDays = [

'D',

'S',

'T',

'Q',

'Q',

'S',

'S',

];

export function SummaryTable() {

return (

<div className="w-full flex">

<div className="grid grid-rows-7 grid-flow-row gap-3">

{weekDays.map((weekDay, i) => {

return (

<div

key={`${weekDay}-${i}`}

className="text-zinc-400 text-xl h-10 w-10 font-bold flex items-center justify-center"

>

{weekDay}

</div>

)

})}

</div>

</div>

);

}

1. Criar arquivo HabitDay

export function HabitDay(){

return (

<div className={ “w-10 h-10 border-2 rounded-lg bg-zinc-900 border-zinc-800”}

)

}

1. Alterar arquivo SummaryTable.tsx, adicionando depois do fechamento da div mais interna

<div className="grid grid-rows-7 grid-flow-col gap-3">

<HabitDay/>

<HabitDay/>

<HabitDay/>

<HabitDay/>

<HabitDay/>

<HabitDay/>

<HabitDay/>

<HabitDay/>

<HabitDay/>

</div>

1. npm install dayjs
2. Criar pasta utils e dentro criar arquivo generate-dates-from-year-beginning.ts

import dayjs from 'dayjs'

export function generateDatesFromYearBeginning() {

const firstDayOfTheYear = dayjs().startOf('year')

const today = new Date()

const dates = []

let compareDate = firstDayOfTheYear

while (compareDate.isBefore(today)) {

dates.push(compareDate.toDate())

compareDate = compareDate.add(1, 'day')

}

return dates

}

1. Alterar SummaryTable para:

import { generateDatesFromYearBeginning } from '../utils/generate-dates-from-year-beginning'

1. E antes da função SummaryTable, adicione

const summaryDates = generateDatesFromYearBeginning()

1. E substitua a chamada dos componentes <HabitDay> por

{summaryDates.map(date => {

return <HabitDay key={data.toString()}/>

})}

1. Alterar SummaryTable, antes da variável summaryDates

const minimumSummaryDatesSize = 18 \* 7 // 18 semanas

const amountOfDaysToFill = minimumSummaryDatesSize - summaryDates.length

1. E finalmente depois da instrução:

{summaryDates.map(date => {

return <HabitDay key={data.toString()}/>

})}

Coloque o código abaixo

{amountOfDaysToFill > 0 && Array.from({ length: amountOfDaysToFill }).map((\_, i) => {

return (

<div

key={i}

className="w-10 h-10 bg-zinc-900 border-2 border-zinc-800 rounded-lg opacity-40 cursor-not-allowed"

/>

)

})}