Roteiro da Aula 3

**BACKEND**

1. Adicionar a rota no arquivo routes.ts

app.patch('/habits/:id/toggle', async (request) => {

const toggleHabitParams = z.object({

id: z.string().uuid()

})

const { id } = toggleHabitParams.parse(request.params)

// recupera o dia de hoje, sem hora, minuto e segundo

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

// verifica se o dia já existe

let day = await prisma.day.findUnique({

where: {

date: today

}

})

// caso o dia não tenha sido criado ainda, pois apenas dias com hábitos

// completados, existem

if (!day) {

day = await prisma.day.create({

data: {

date: today

}

})

}

// procura para saber se o hábito já foi completado

const dayHabit = await prisma.dayHabit.findUnique({

where: {

day\_id\_habit\_id: {

day\_id: day.id,

habit\_id: id

}

}

})

// se já foi completado

if (dayHabit) { // remove marcação

await prisma.dayHabit.delete({

where: {

id: dayHabit.id

}

})

} else { //não foi completada, então cria

await prisma.dayHabit.create({

data: {

day\_id: day.id,

habit\_id: id

}

})

}

})

1. Adicionar a rota no arquivo routes.ts

// [{day\_id, date, completed, ammount},

// {day\_id, date, completed, ammount},

// {day\_id, date, completed, ammount} ]

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

const summary = await prisma.$queryRaw`

SELECT

D.id,

D.date,

(

SELECT

cast(count(\*) as float)

FROM day\_habits DH

WHERE DH.day\_id = D.id

) as completed,

(

SELECT

cast(count(\*) as float)

FROM habit\_week\_days HDW

JOIN habits H

ON H.id = HDW.habit\_id

WHERE

HDW.week\_day = cast(strftime('%w', D.date/1000.0, 'unixepoch') as int)

AND H.created\_at <= D.date

) as amount

FROM days D

`

return summary

})

**FRONTEND**

1. Instalar dependência radix

npm install @radix-ui/react-dialog

1. Criar componente NewHabitForm.tsx

import { Check } from "phosphor-react";

export function NewHabitForm() {

<form className="w-full flex flex-col mt-6">

<label htmlFor="title" className="font-semibold leading-tight">

Qual seu comprometimento?

</label>

<input

type="text"

id="title"

placeholder="ex.: Exercícios, dormir bem, etc..."

className="p-4 rounded-lg mt-3 bg-zinc-800 text-white"

autoFocus

/>

<label htmlFor="" className="font-semibold leading-tight mt-4">

Qual a recorrência?

</label>

<button

type="submit"

>

<Check size={20} weight="bold" />

Confirmar

</button>

</form>

)

}

1. Alterar arquivo Header.tsx para:

import { Plus, X } from "phosphor-react";

import \* as Dialog from '@radix-ui/react-dialog';

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

import { NewHabitForm } from "./NewHabitForm";

export function Header() {

return (

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

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

<Dialog.Root>

<Dialog.Trigger

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

</Dialog.Trigger>

<Dialog.Portal>

<Dialog.Overlay className="w-screen h-screen bg-black/80 fixed inset-0" />

<Dialog.Content className="absolute p-10 bg-zinc-900 rounded-2xl w-full max-w-md top-1/2 left-1/2 -translate-x-1/2 -translate-y-1/2">

<Dialog.Close className="absolute right-6 top-6 text-zinc-400 rounded-lg hover:text-zinc-200 ">

<X size={24} aria-label="Fechar" />

</Dialog.Close>

<Dialog.Title className="text-3xl leading-tight font-extrabold">

Criar bábito

</Dialog.Title>

<NewHabitForm />

</Dialog.Content>

</Dialog.Portal>

</Dialog.Root>

</div>

);

}

1. Instalar a dependência popover

npm install @radix-ui/react-popover

1. Criar arquivo ProgressBar.tsx

interface ProgressBarProps {

progress: number

}

export function ProgressBar(props: ProgressBarProps) {

return (

<div className='h-3 rounded-xl bg-zinc-700 w-full mt-4'>

<div

role="progressbar"

aria-label="Progresso de hábitos completados nesse dia"

aria-valuenow={props.progress}

className="h-3 rounded-xl bg-violet-600 transition-all"

style={{

width: `${props.progress}%`

}}

/>

</div>

)

}

1. Alterar HabitDay.tsx na parte de return para

return (

<Popover.Root>

<Popover.Trigger

className="w-10 h-10 bg-zinc-900 border-2 border-zinc-800 rounded-lg” />

<Popover.Portal>

<Popover.Content className="min-w-[320px] p-6 rounded-2xl bg-zinc-900 flex flex-col">

<span className="font-semibold text-zinc-400">terça-feira</span>

<span className="mt-1 font-extrabold leading-tight text-3xl">17/01 </span>

<ProgressBar progress={75} />

<Popover.Arrow height={8} width={16} className='fill-zinc-900' />

</Popover.Content>

</Popover.Portal>

</Popover.Root>

);

1. Instalar a dependência clsx

npm install clsx

1. Alterar arquivo HabitDay adicionando:

interface HabitDayProps {

completed: number

amount: number

}

1. Criar variável como:

const completedPercentage = Math.round((completed / amount) \* 100);

1. Alterar Popover.Trigger para

<Popover.Trigger

className={clsx("w-10 h-10 bg-zinc-900 border-2 border-zinc-800 rounded-lg ", {

'bg-zinc-900 border-zinc-800': comlpetedPercentage === 0,

'bg-violet-900 border-violet-500': comlpetedPercentage > 0 && comlpetedPercentage < 20,

'bg-violet-800 border-violet-500': comlpetedPercentage >= 20 && comlpetedPercentage < 40,

'bg-violet-700 border-violet-500': comlpetedPercentage >= 40 && comlpetedPercentage < 60,

'bg-violet-600 border-violet-500': comlpetedPercentage >= 60 && comlpetedPercentage < 80,

'bg-violet-500 border-violet-400': comlpetedPercentage >= 80,

})}

/>

1. Alterar SummaryTable.tsx para:

return (

<HabitDay

key={date.toString()}

amount={5}

completed={Math.round(Math.random() \* 5)}

/>