Roteiro da Aula 5

FrontEnd

1. Criar componente HabitList.tsx

import \* as Checkbox from '@radix-ui/react-checkbox'

import { Check } from 'phosphor-react'

import { useEffect } from 'react'

export function HabitsList() {

useEffect(() => {

console.log('Carregou')

}, [])

return (

<div className="mt-6 flex flex-col gap-3">

<Checkbox.Root

className="flex items-center gap-3 group"

>

<div

className="h-8 w-8 rounded-lg flex items-center justify-center bg-zinc-900 border-2 border-zinc-800 group-data-[state=checked]:bg-green-500 group-data-[state=checked]:border-green-500">

<Checkbox.Indicator>

<Check size={20} className="text-white" />

</Checkbox.Indicator>

</div>

<span className="font-semibold text-xl text-white leading-tight group-data-[state=checked]:line-through group-data-[state=checked]:text-zinc-400">

Beber 2L de água

</span>

</Checkbox.Root>

</div>

)

}

1. Alterar HabitDay.tsx para

...

<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">{dayOfWeek}</span>

<span className="mt-1 font-extrabold leading-tight text-3xl">{dayAndMonth}</span>

<ProgressBar progress={comlpetedPercentage} />

<HabitsList date={date} onCompletedChanged={handleCompletedChaged} />

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

</Popover.Content>

</Popover.Portal>

...

1. Alterar HabitList.tsx para:

interface HabitLisProps {

date: Date

}

interface HabitsInfo {

possibleHabits: {

id: string;

title: string;

created\_at: string

}[]

completedHabits: string[]

}

1. Alterar HabitList.tsx para

export function HabitsList({ date, onCompletedChanged }: HabitLisProps) {

const [habitsInfo, setHabitsInfo] = useState<HabitsInfo>()

useEffect(() => {

api.get('day', {

params:{

date: date.toISOString()

}

}).then(response => {

setHabitsInfo(response.data)

})

},[])

1. Alterar HabitList.tsx para

return (

<div className='mt-6 flex flex-col gap-3'>

{habitsInfo?.possibleHabits.map(habit => {

return (

<Checkbox.Root

key={habit.id}

checked={habitsInfo.completedHabits.includes(habit.id)}

className='flex items-center gap-3 group'

>

<div className='h-8 w-8 rounded-lg flex items-center justify-center bg-zinc-900 border-2 border-zinc-800 group-data-[state=checked]:bg-green-500 group-data-[state=checked]:border-green-50 transition-colors group-focus:ring-2 group-focus:ring-violet-600 group-focus:ring-offset-2 group-focus:ring-offset-background'>

<Checkbox.Indicator>

<Check size={20} className="text-white" />

</Checkbox.Indicator>

</div>

<span className='font-semibold text-xl text-white leading-tight group-data-[state=checked]:line-through group-data-[state=checked]:text-zinc-400'>

{habit.title}

</span>

</Checkbox.Root>

)

})}

</div>

)

1. Alterar HabitList.tsx para

...

const isDateInPast = dayjs(date).endOf('day').isBefore(new Date())

...

<Checkbox.Root

...

disabled={isDateInPast}

...

>

7. Alterar HabitList.tsx para

async function handleToggleHabit(habitId: string) {

const isHabitAlreadyCompleted = habitsInfo!.completedHabits.includes(habitId)

await api.patch(`/habits/${habitId}/toggle`)

let completedHabits: string[] = []

if(isHabitAlreadyCompleted) {

completedHabits = habitsInfo!.completedHabits.filter(id => id !== habitId)

} else {

completedHabits = [...habitsInfo!.completedHabits, habitId]

}

setHabitsInfo({

possibleHabits: habitsInfo!.possibleHabits,

completedHabits

})

}

8. Alterar arquivo HabitList.tsx para

<Checkbox.Root

...

onCheckedChange={() => handleToggleHabit(habit.id)}

...

>