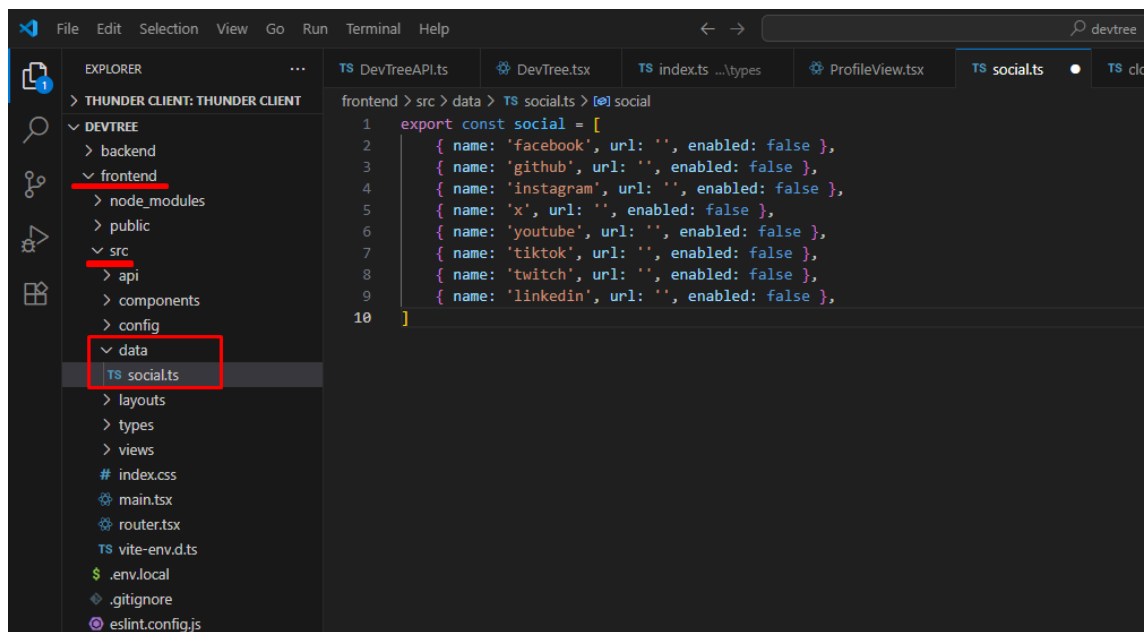
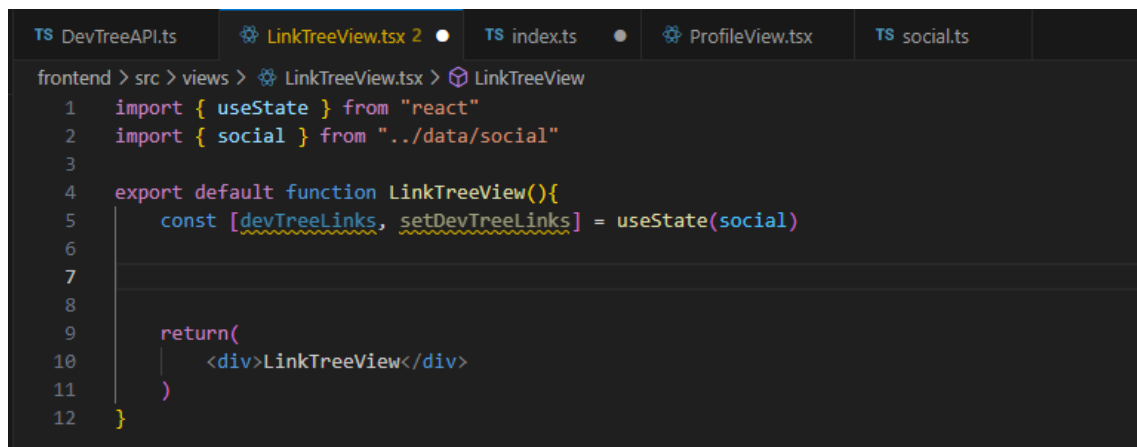


Creando el área de enlaces para DevTree



No viene desde la bd porque es info estática, lo que es dinámico será las redes de cada persona.



```

10 //Pick permite seleccionar otros atributos de otro type
11 export type RegisterForm = Pick<User, 'handle' | 'email' | 'name'> & {
12     password: string
13     password_confirmation: string
14 }
15
16 export type LoginForm = Pick<User, 'email'> & {
17     password: string
18 }
19
20 export type ProfileForm = Pick<User, 'handle' | 'description'>
21
22 export type SocialNetwork = {
23     id: number
24     name: string
25     url: string
26     enabled: boolean
27 }
28 export type DevTreeLink = Pick<SocialNetwork, 'name' | 'url' | 'enabled'>
  
```

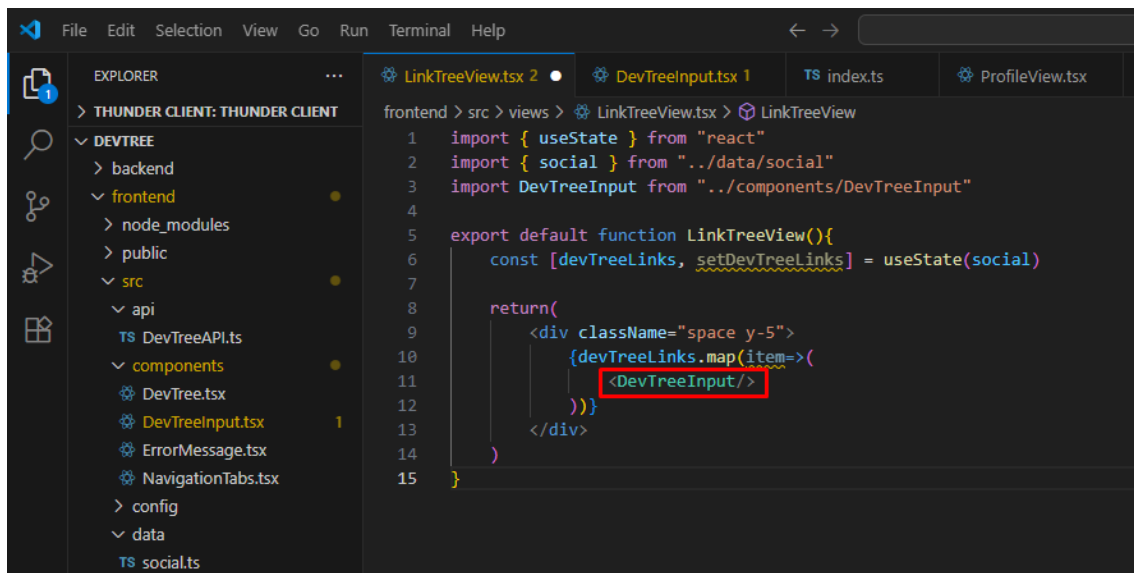
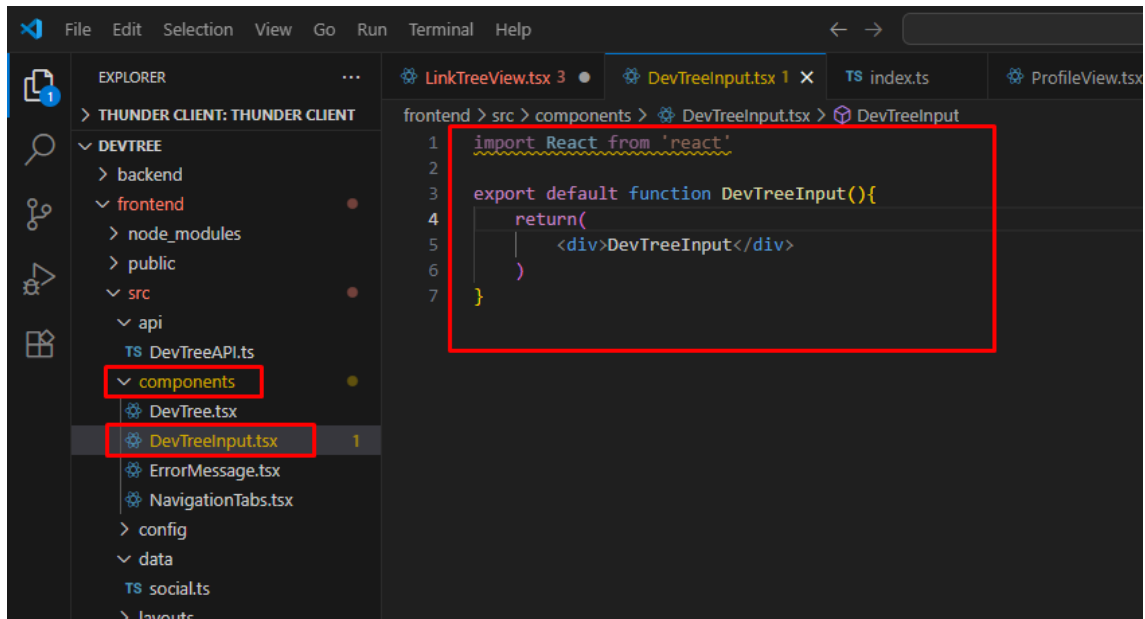
```

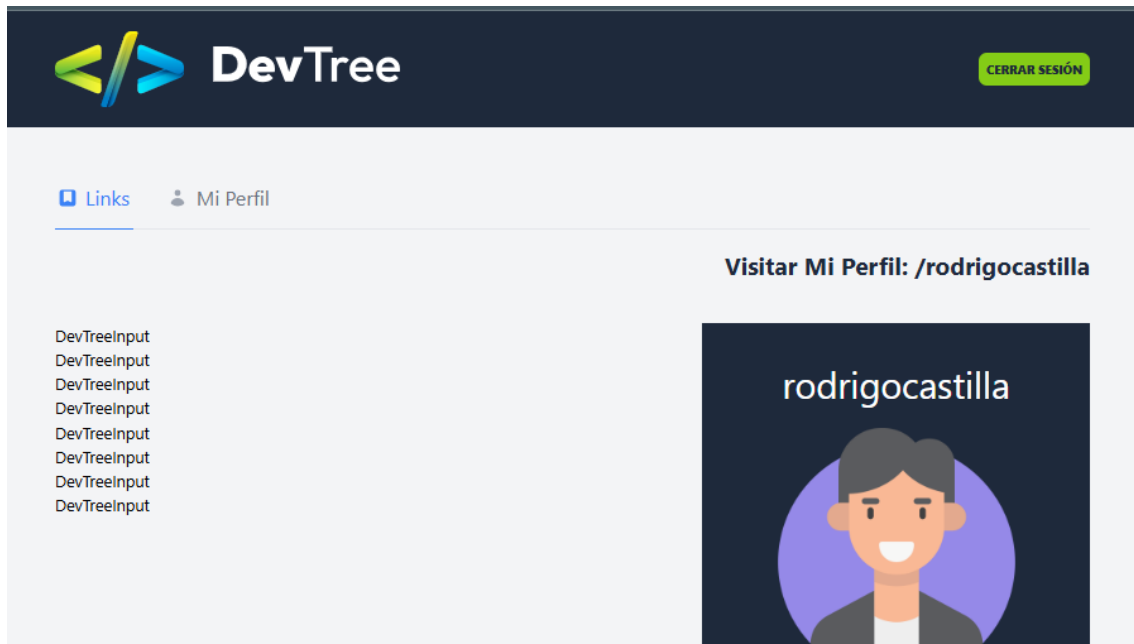
1 import type { DevTreeLink } from "../types";
2
3 export const social : DevTreeLink[] = [
4     { name: 'facebook', url: '', enabled: false },
5     { name: 'github', url: '', enabled: false },
6     { name: 'instagram', url: '', enabled: false },
7     { name: 'x', url: '', enabled: false },
8     { name: 'youtube', url: '', enabled: false },
9     { name: 'tiktok', url: '', enabled: false },
10    { name: 'twitch', url: '', enabled: false },
11    { name: 'linkedin', url: '', enabled: false },
12 ]
  
```

Mostrando el formulario de Redes Sociales

```

1 import { useState } from "react"
2 import { social } from "../data/social"
3
4 export default function LinkTreeView(){
5     const [devTreeLinks, setDevTreeLinks] = useState(social)
6
7     return(
8         <div className="space y-5">
9             {devTreeLinks.map(item=>{
10
11             })}
12         </div>
13     )
14 }
  
```





```

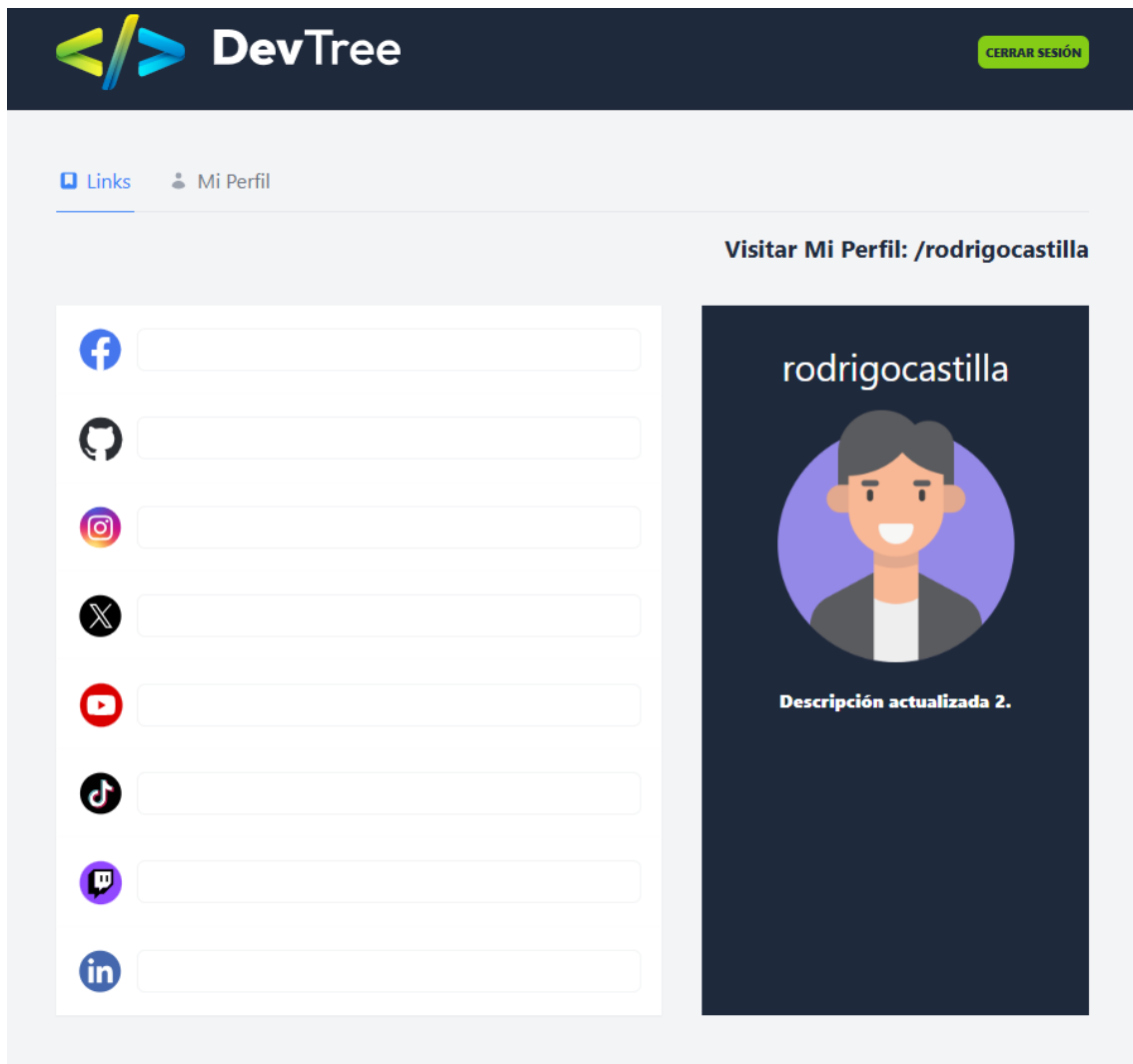
un Terminal Help
LinkTreeView.tsx 2 • DevTreeInput.tsx 1 TS index.ts ProfileView.tsx TS
frontend > src > views > LinkTreeView.tsx > LinkTreeView > devTreeLinks.map() callback
1 import { useState } from "react"
2 import { social } from "../data/social"
3 import DevTreeInput from "../components/DevTreeInput"
4
5 export default function LinkTreeView(){
6   const [devTreeLinks, setDevTreeLinks] = useState(social)
7
8   return(
9     <div className="space y-5">
10       {devTreeLinks.map(item=>{
11         <DevTreeInput
12           key={item.name}
13           //pasamos el item hacia el componente
14           item={item}
15         />
16       )}}
17     </div>
18   )
19 }

```

Sale rojo porque no sabe que este componente espera el prop de item.

import type { DevTreeLink } from "../types"

```
LinkTreeView.tsx | DevTreeInput.tsx | TS index.ts | ProfileView.tsx | TS social.ts
frontend > src > components > DevTreeInput.tsx > DevTreeInput
1  import { DevTreeLink } from "../types"
2
3  type DevTreeInputProps = {
4    item: DevTreeLink
5  }
6
7  export default function DevTreeInput({item}: DevTreeInputProps){
8
9    return(
10     <div className="bg-white shadow-sm p-5 flex items-center gap-3">
11       <div
12         className="w-12 h-12 bg-cover"
13         style={{backgroundImage: `url('/social/icon_${item.name}.svg')`}}
14       ></div>
15       <input
16         type="text"
17         className="flex-1 border border-gray-100 rounded-lg"
18       />
19     </div>
20   )
21 }
```



Añadiendo un Switch para habilitar y deshabilitar redes sociales

<https://headlessui.com/> Serie de componentes dinámicos hecho por los creadores de tailwindcss

```

● PS C:\Users\rodri\devtree\frontend> npm install @headlessui/react

added 20 packages, and audited 316 packages in 17s

87 packages are looking for funding
  run `npm fund` for details

found 0 vulnerabilities
● PS C:\Users\rodri\devtree\frontend> 

```

```

export default function DevTreeInput({ item }: DevTreeInputProps) {
  style={({ backgroundImage: `url('/social/icon_${item.name}.svg')` })}
  </div>
  <input
    type="text"
    className="flex-1 border border-gray-100 rounded-lg"
  />
  <Switch
    checked={item.enabled}
    onChange={() => { }}
    className={classNames(
      item.enabled ? 'bg-blue-500' : 'bg-gray-200',
      'relative inline-flex h-6 w-11 flex-shrink-0 cursor-pointer rounded-full border-2 border-transparent transition-colors duration-200 ease-in-out focus:outline-none'
    )}
  >
    <span
      aria-hidden="true"
      className={classNames(
        item.enabled ? 'translate-x-5' : 'translate-x-0',
        'pointer-events-none inline-block h-5 w-5 transform rounded-full bg-white shadow ring-0 transition duration-200 ease-in-out'
      )}
    />
  </Switch>
</div>
}

```

Creamos nueva carpeta y archivo index.ts

```

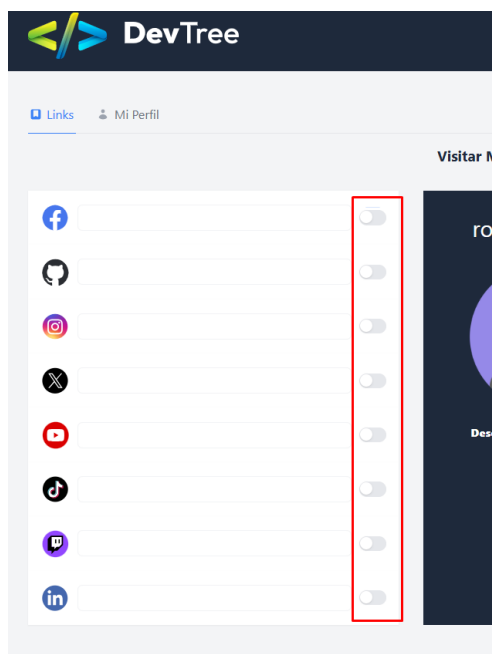
export function classNames(...classes : string[]) {
  return classes.filter(Boolean).join(' ')
}

```

```

frontend > src > components > DevTreeInput.tsx > DevTreeInputProps
1  import { Switch } from "@headlessui/react"
2  import type { DevTreeLink } from "../types"
3  import { classNames } from "../utils"
4
5  type DevTreeInputProps = {
6    item: DevTreeLink
7  }
8
9  export default function DevTreeInput({ item }: DevTreeInputProps) {
10
11    return (
12      <div className="bg-white shadow-sm p-5 flex items-center gap-

```



No lo podemos habilitar porque su value es el checked.

```

frontend > src > data > TS social.ts > social
1  import type { DevTreeLink } from "../types";
2
3  export const social : DevTreeLink[] = [
4    { name: 'facebook', url: '', enabled: false },
5    { name: 'github', url: '', enabled: false },
6    { name: 'instagram', url: '', enabled: true },
7    { name: 'x', url: '', enabled: false },
8    { name: 'youtube', url: '', enabled: false },
9    { name: 'tiktok', url: '', enabled: false },
10   { name: 'twitch', url: '', enabled: false },
11   { name: 'linkedin', url: '', enabled: false },
12 ]

```



Lo devolvemos a false.

Identificando la red social en la que estamos escribiendo

```
LinkTreeView.tsx 1 x DevTreeInput.tsx TS index.ts ProfileView.tsx TS social.ts
frontend > src > views > LinkTreeView.tsx > LinkTreeView
1  import { useState } from "react"
2  import { social } from "../data/social"
3  import DevTreeInput from "../components/DevTreeInput"
4
5  export default function LinkTreeView(){
6      const [devTreeLinks, setDevTreeLinks] = useState(social)
7
8      const handleUrlChange = () => {
9          console.log('escribiendo...')
10     }
11
12     return(
13         <div className="space y-5">
14             {devTreeLinks.map(item=>{
15                 <DevTreeInput
16                     key={item.name}
17                     //pasamos el item hacia el componente
18                     item={item}
19                     handleUrlChange={handleUrlChange}
20                 />
21             )}}
22         </div>
23     )
24 }
```



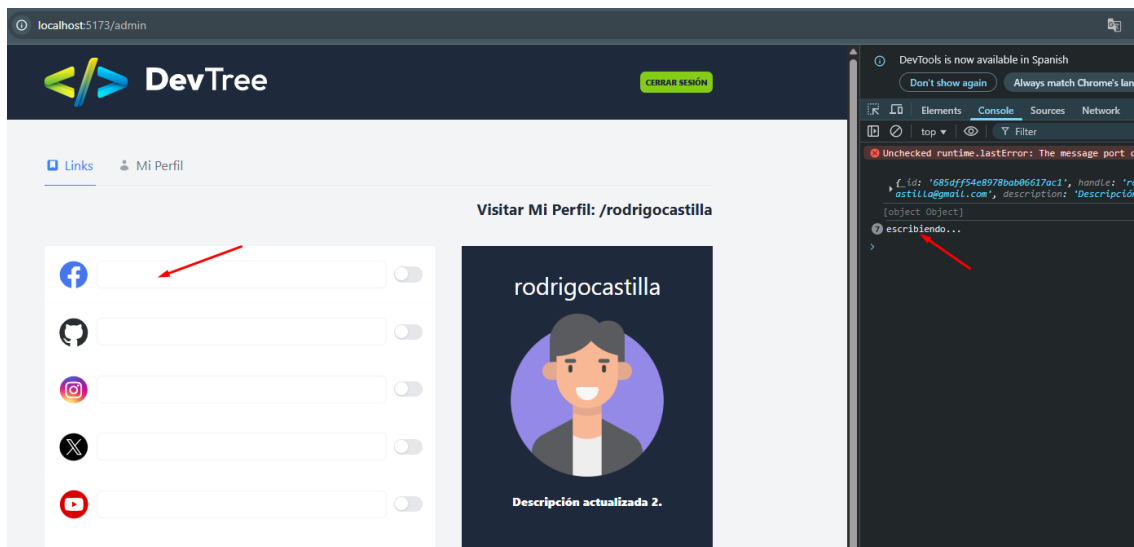
```

Terminal  Help  devtree
LinkTreeView.tsx 1  DevTreeInput.tsx  TS index.ts  ProfileView.tsx  TS social.ts

frontend > src > components > DevTreeInput.tsx > DevTreeInput
1  import { Switch } from "@headlessui/react"
2  import type { DevTreeLink } from "../types"
3  import { classNames } from "../utils"
4
5  type DevTreeInputProps = {
6    item: DevTreeLink
7    handleChange : () => void
8  }
9
10 export default function DevTreeInput({ item, handleChange : DevTreeInputProps } {
11
12   return (
13     <div className="bg-white shadow-sm p-5 flex items-center gap-3">
14       <div
15         className="w-12 h-12 bg-cover"
16         style={{ backgroundImage: `url('/social/icon_${item.name}.svg')` }}
17       ></div>
18       <input
19         type="text"
20         className="flex-1 border border-gray-100 rounded-lg"
21         value={item.url}
22         onChange={handleChange}
23       />
24
25       <Switch
26         checked={item.enabled}

```

Escribo algo...



```

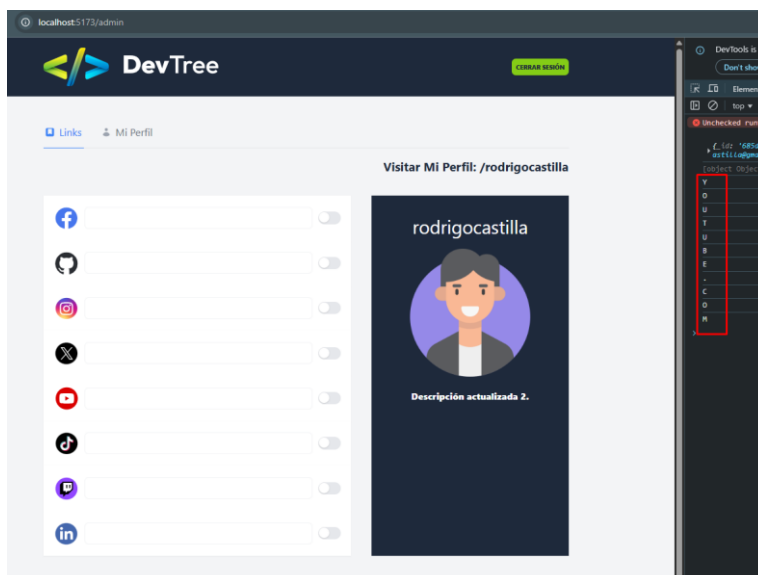
LinkTreeView.tsx 1 x DevTreeInput.tsx TS index.ts ProfileView.tsx TS social.ts
frontend > src > views > LinkTreeView.tsx > LinkTreeView > devTreeLinks.map() callback
1  import { useState } from "react"
2  import { social } from "../data/social"
3  import DevTreeInput from "../components/DevTreeInput"
4
5  export default function LinkTreeView(){
6      const [devTreeLinks, setDevTreeLinks] = useState(social)
7
8      const handleUrlChange = (e : React.ChangeEvent<HTMLInputElement>) => {
9          console.log(e.target.value)
10     }
11
12     return(
13         <div className="space y-5">
14             {devTreeLinks.map(item=>{
15                 <DevTreeInput
16                     key={item.name}
17                     //pasamos el item hacia el componente
18                     item={item}
19                     handleUrlChange={handleUrlChange}

```

```

Terminal Help
LinkTreeView.tsx 1 DevTreeInput.tsx x TS index.ts ProfileView.tsx TS social.ts
frontend > src > components > DevTreeInput.tsx > DevTreeInput
1  import { Switch } from "@headlessui/react"
2  import type { DevTreeLink } from "../types"
3  import { classNames } from "../utils"
4
5  type DevTreeInputProps = {
6      item: DevTreeLink
7      handleUrlChange : (e : React.ChangeEvent<HTMLInputElement>) => void
8  }
9
10 export default function DevTreeInput({ item, handleUrlChange }: DevTreeInputProps) {
11
12     return (
13         <div className="bg-white shadow-sm p-5 flex items-center gap-3">

```



```

Run Terminal Help
devtree

LinkTreeView.tsx | DevTreeInput.tsx | TS index.ts | ProfileView.tsx | TS social.ts

frontend > src > components > DevTreeInput.tsx > DevTreeInput
1 import { Switch } from "@headlessui/react"
2 import type { DevTreeLink } from "../types"
3 import { classNames } from "../utils"
4
5 type DevTreeInputProps = {
6   item: DevTreeLink
7   handleUrlChange : (e : React.ChangeEvent<HTMLInputElement>) => void
8 }
9
10 export default function DevTreeInput({ item, handleUrlChange } : DevTreeInputProps) {
11
12   return (
13     <div className="bg-white shadow-sm p-5 flex items-center gap-3">
14       <div
15         className="w-12 h-12 bg-cover"
16         style={{ backgroundImage: `url('/social/icon_${item.name}.svg')` }}
17       >>/div>
18       <input
19         type="text"
20         className="flex-1 border border-gray-100 rounded-lg"
21         value={item.url}
22         onChange={handleUrlChange}
23         name={item.name} //en cuál red social estoy escribiendo para actualizar la URL de la red social
24       />
25
26       <Switch
27         checked={item.enabled}
28         onChange={() => { }}
29       />
30     </div>
31   )
32 }

```

Almacenando los enlaces en las redes sociales

```

Terminal Help
devtree

LinkTreeView.tsx | DevTreeInput.tsx | TS index.ts | ProfileView.tsx | TS social.ts

frontend > src > views > LinkTreeView.tsx > LinkTreeView > handleUrlChange
1 import { useState } from "react"
2 import { social } from "../data/social"
3 import DevTreeInput from "../components/DevTreeInput"
4
5 export default function LinkTreeView(){
6   const [devTreeLinks, setDevTreeLinks] = useState(social)
7
8   const handleUrlChange = (e : React.ChangeEvent<HTMLInputElement>) => {
9
10     // Mapea cada elemento del estado actual
11     // Si el nombre del enlace coincide con el del input que generó el evento,
12     // actualiza su propiedad 'url' con el nuevo valor ingresado
13     const updatedLinks = devTreeLinks.map(link => link.name === e.target.name ? {...link, url: e.target.value} : link)
14
15     console.log(updatedLinks)
16
17     // Actualiza el estado con la nueva lista de enlaces
18     setDevTreeLinks(updatedLinks)
19   }
20 }

```

Don't show again Always match Chrome's language Switch

Elements Console Sources Network Performance

top Filter

(0) {name: 'facebook', url: 'http://facebook.com', enabled: false}
 (1) {name: 'github', url: '', enabled: false}
 (2) {name: 'instagram', url: '', enabled: false}
 (3) {name: 'twitter', url: '', enabled: false}
 (4) {name: 'youtube', url: '', enabled: false}
 (5) {name: 'tiktok', url: '', enabled: false}
 (6) {name: 'twitch', url: '', enabled: false}
 (7) {name: 'linkedin', url: '', enabled: false}
 (8) {name: 'facebook', url: 'http://facebook.com', enabled: false}

Habilitando/deshabilitando enlaces de nuestro tree

```

LinkTreeView.tsx | DevTreeInput.tsx 4 | TS index.ts | ProfileView.tsx | TS social.ts
frontend > src > views > LinkTreeView.tsx > LinkTreeView > devTreeLinks.map() callback
5  export default function LinkTreeView(){
8    const handleUrlChange = (e : React.ChangeEvent<HTMLInputElement>) => {
14
15      console.log(updatedLinks)
16
17      // Actualiza el estado con la nueva lista de enlaces
18      setDevTreeLinks(updatedLinks)
19    }
20
21    const handleEnableLink = () => {
22      console.log('habilitando o deshabilitando')
23    }
24
25    return(
26      <div className="space y-5">
27        {devTreeLinks.map(item=>{
28          <DevTreeInput
29            key={item.name}
30            //pasamos el item hacia el componente
31            item={item}
32            handleUrlChange={handleUrlChange}
33            handleEnableLink={handleEnableLink}
34          </>
35        })}
36      </div>
37    )

```

```

Terminal Help | devtree
LinkTreeView.tsx | DevTreeInput.tsx 1 | TS index.ts | ProfileView.tsx | TS social.ts
frontend > src > components > DevTreeInput.tsx > DevTreeInput
1  import { Switch } from "@headlessui/react"
2  import type { DevTreeLink } from "../types"
3  import { classNames } from "../utils"
4
5  type DevTreeInputProps = {
6    item: DevTreeLink
7    handleUrlChange : (e : React.ChangeEvent<HTMLInputElement>) => void
8    handleEnableLink : () => void
9  }
10
11  export default function DevTreeInput({ item, handleUrlChange, handleEnableLink }: DevTreeInputProps) {
12
13    return (

```

```

Terminal Help
LinkTreeView.tsx | DevTreeInput.tsx | TS index.ts | ProfileView.tsx | TS social.ts

frontend > src > components > DevTreeInput.tsx > DevTreeInput
1 import { Switch } from "@headlessui/react"
2 import type { DevTreeLink } from "../types"
3 import { classNames } from "../utils"
4
5 type DevTreeInputProps = {
6   item: DevTreeLink
7   handleUrlChange : (e : React.ChangeEvent<HTMLInputElement>) => void
8   handleEnableLink : () => void
9 }
10
11 export default function DevTreeInput({ item, handleUrlChange, handleEnableLink } : DevTreeInputProps) {
12
13   return (
14     <div className="bg-white shadow-sm p-5 flex items-center gap-3">
15       <div
16         className="w-12 h-12 bg-cover"
17         style={{ backgroundImage: `url('/social/icon_${item.name}.svg')` }}
18       >>/div>
19       <input
20         type="text"
21         className="flex-1 border border-gray-100 rounded-lg"
22         value={item.url}
23         onChange={handleUrlChange}
24         name={item.name} //en cuál red social estoy escribiendo para actualizar la URL de la red social
25       />
26
27       <Switch
28         checked={item.enabled}
29         onChange={()=>handleEnableLink(item.name)} //le pasamos el nombre de la red social donde estamos haciendo el cambio
30         className={classNames(
31           item.enabled ? 'bg-blue-500' : 'bg-gray-200',
32           'relative inline-flex h-6 w-11 flex-shrink-0 cursor-pointer rounded-full border-2 border-transparent transition-colors duration-

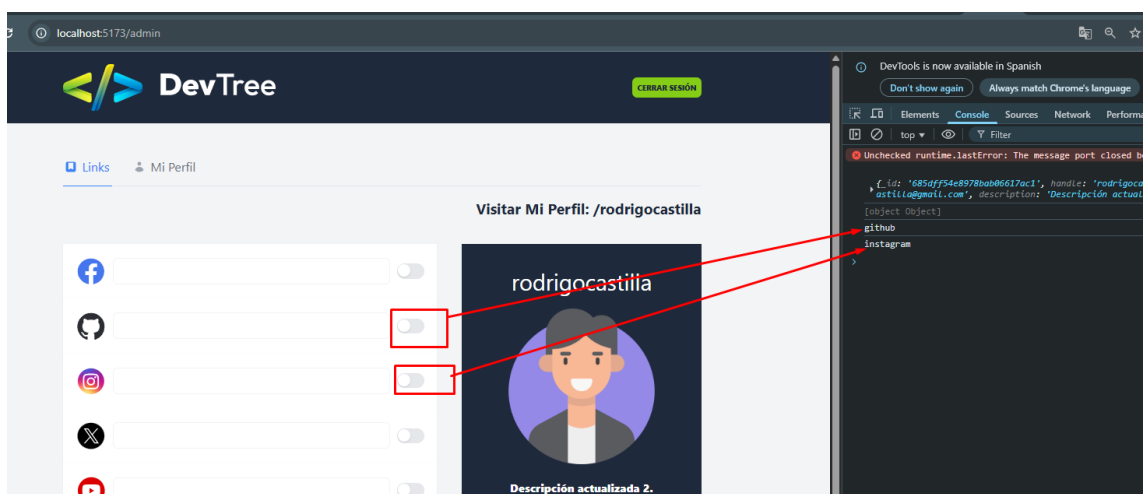
```

```

Terminal Help
LinkTreeView.tsx | DevTreeInput.tsx | TS index.ts | ProfileView.tsx | TS social.ts

frontend > src > views > LinkTreeView.tsx > LinkTreeView > handleEnableLink
4 import { useState } from "react"
5 import { DevTreeLink } from "../types"
6
7 export default function LinkTreeView() {
8   const [devTreeLinks, setDevTreeLinks] = useState<DevTreeLink[]>()
9
10   const handleUrlChange = (e : React.ChangeEvent<HTMLInputElement>) => {
11     // Mapea cada elemento del estado actual
12     // Si el nombre del enlace coincide con el del input que generó el evento,
13     // actualiza su propiedad 'url' con el nuevo valor ingresado
14     const updatedLinks = devTreeLinks.map(link => link.name === e.target.name ? {...link, url: e.target.value} : link)
15
16     console.log(updatedLinks)
17
18     // Actualiza el estado con la nueva lista de enlaces
19     setDevTreeLinks(updatedLinks)
20   }
21
22   const handleEnableLink = (socialNetwork: string) => {
23     console.log(socialNetwork)
24   }
25
26   return(
27     <div className="bg-white p-5">

```



Para ver si está habilitado o no:

```

4
5 export default function LinkTreeView(){
6   const [devTreeLinks, setDevTreeLinks] = useState(social)
7
8   const handleUrlChange = (e : React.ChangeEvent<HTMLInputElement>) => {
9
10    // Mapea cada elemento del estado actual
11    // Si el nombre del enlace coincide con el del input que generó el evento,
12    // actualiza su propiedad 'url' con el nuevo valor ingresado
13    const updatedLinks = devTreeLinks.map(link => link.name === e.target.name ? {...link, url: e.target.value} : link)
14
15    console.log(updatedLinks)
16
17    // Actualiza el estado con la nueva lista de enlaces
18    setDevTreeLinks(updatedLinks)
19  }
20
21  const handleEnableLink = (socialNetwork: string) => {
22    //console.log(socialNetwork)
23    const updatedLinks = devTreeLinks.map(link => link.name === socialNetwork ? {...link, enabled: !link.enabled} : link)
24    console.log(updatedLinks)
25    setDevTreeLinks(updatedLinks)
26  }
27
28  return(

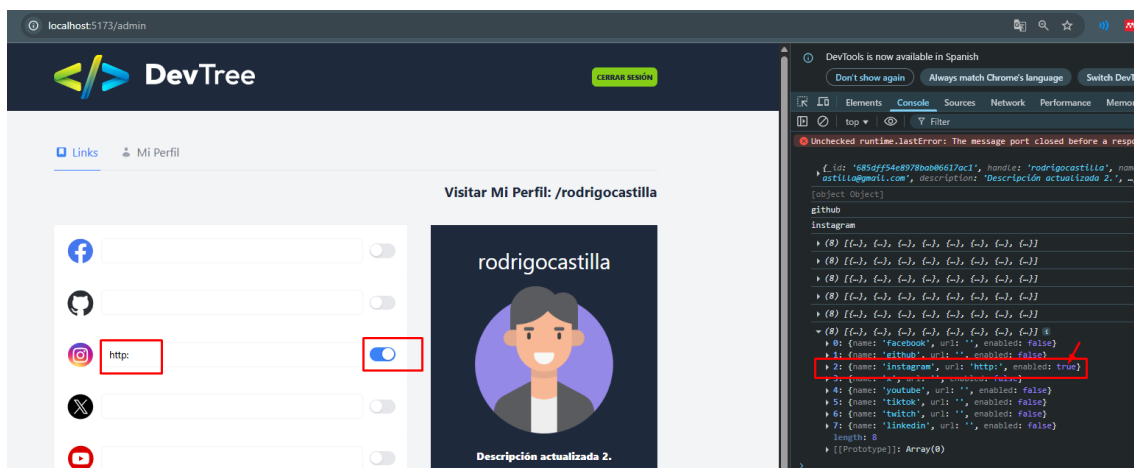
```

Veamos los errores:

```

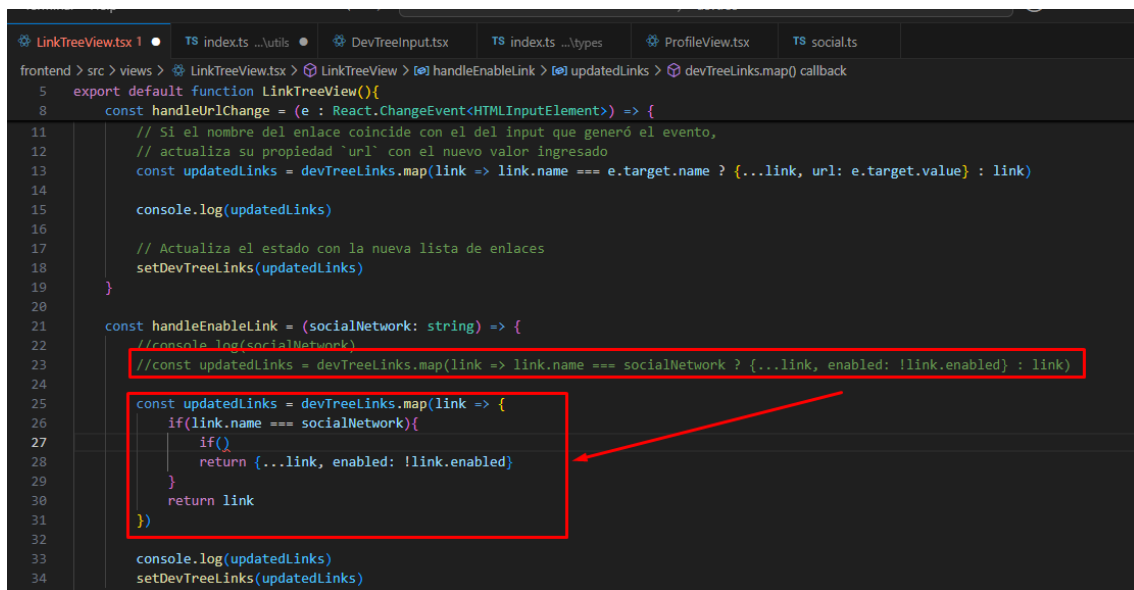
1 import { Switch } from "@headlessui/react"
2 import type { DevTreeLink } from "../types"
3 import { classNames } from "../utils"
4
5 type DevTreeInputProps = {
6   item: DevTreeLink
7   handleUrlChange : (e : React.ChangeEvent<HTMLInputElement>) => void
8   handleEnableLink : (socialNetwork: string) => void
9 }
10
11 export default function DevTreeInput({ item, handleUrlChange, handleEnableLink } : DevTreeInputProps) {
12
13   return (

```



Pero que pasa si ingreso un enlace inválido, veamos.

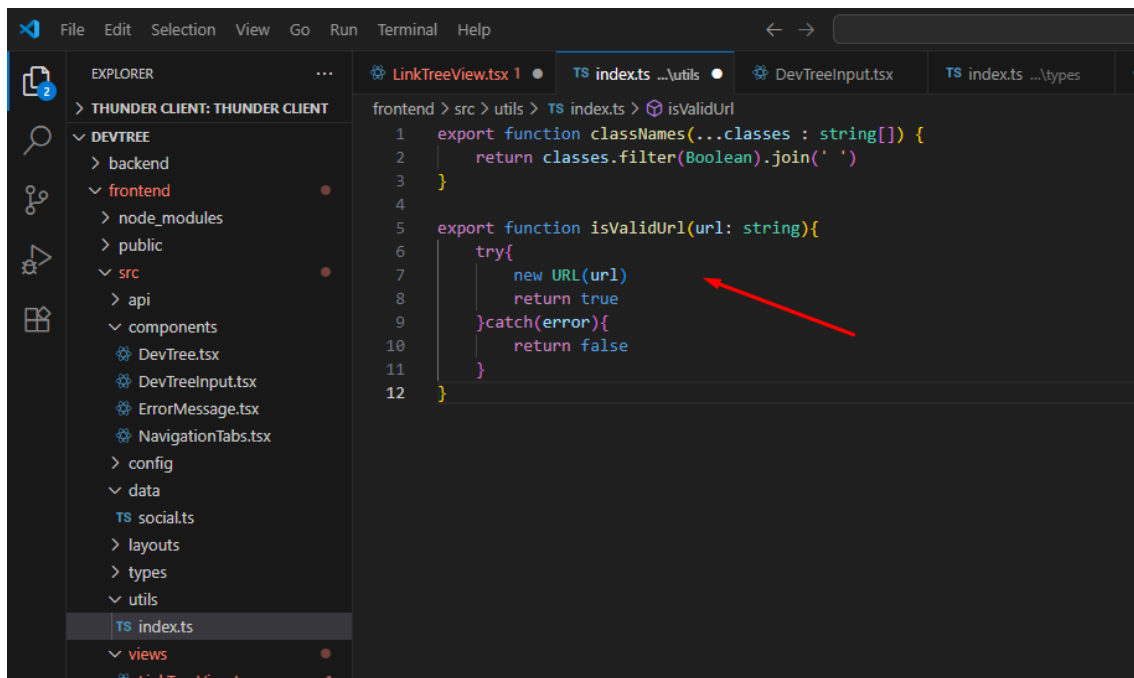
Detectando si una URL es válida antes de habilitarla.



```

5 export default function LinkTreeView() {
8   const handleUrlChange = (e: React.ChangeEvent<HTMLInputElement>) => {
11     // Si el nombre del enlace coincide con el del input que generó el evento,
12     // actualiza su propiedad 'url' con el nuevo valor ingresado
13     const updatedLinks = devTreeLinks.map(link => link.name === e.target.name ? {...link, url: e.target.value} : link)
14
15     console.log(updatedLinks)
16
17     // Actualiza el estado con la nueva lista de enlaces
18     setDevTreeLinks(updatedLinks)
19   }
20
21   const handleEnableLink = (socialNetwork: string) => {
22     //console.log(socialNetwork)
23     //const updatedLinks = devTreeLinks.map(link => link.name === socialNetwork ? {...link, enabled: !link.enabled} : link)
24
25     const updatedLinks = devTreeLinks.map(link => {
26       if(link.name === socialNetwork){
27         if(){
28           return {...link, enabled: !link.enabled}
29         }
30         return link
31       })
32
33     console.log(updatedLinks)
34     setDevTreeLinks(updatedLinks)
  }

```



```

1 export function classNames(...classes: string[]) {
2   return classes.filter(Boolean).join(' ')
3 }
4
5 export function isValidUrl(url: string){
6   try{
7     new URL(url)
8     return true
9   }catch(error){
10    return false
11  }
12 }

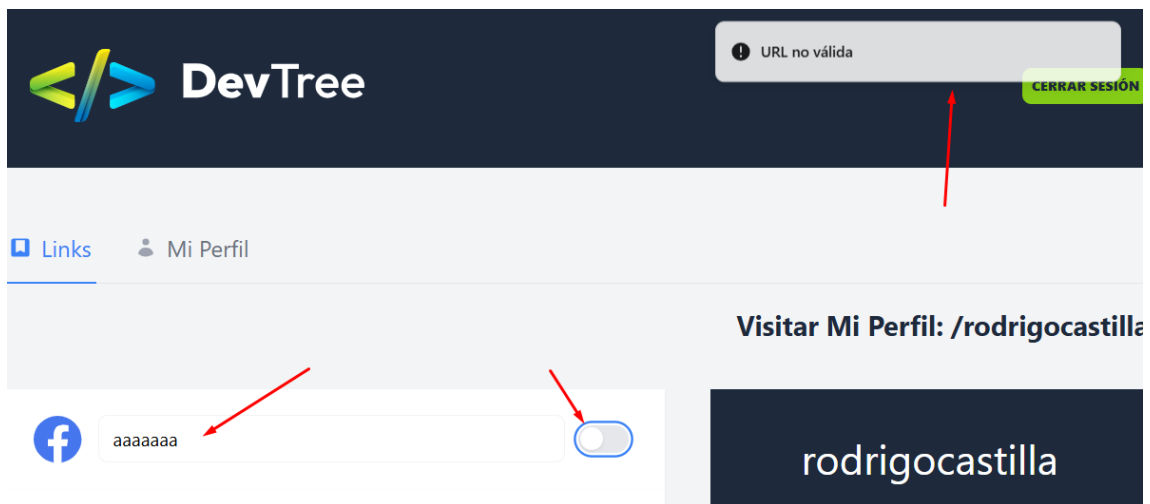
```

```

Run Terminal Help
devtree

LinkTreeView.tsx • TS index.ts ...utils • DevTreeInput.tsx TS index.ts ...types ProfileView.tsx TS social.ts
frontend > src > views > LinkTreeView.tsx > LinkTreeView > handleEnableLink > updatedLinks > devTreeLinks.map() callback
7 export default function LinkTreeView(){
10   const handleUrlChange = (e : React.ChangeEvent<HTMLInputElement>) => {
11
12     // Mapea cada elemento del estado actual
13     // Si el nombre del enlace coincide con el del input que generó el evento,
14     // actualiza su propiedad `url` con el nuevo valor ingresado
15     const updatedLinks = devTreeLinks.map(link => link.name === e.target.name ? {...link, url: e.target.value} : link)
16
17     console.log(updatedLinks)
18
19     // Actualiza el estado con la nueva lista de enlaces
20     setDevTreeLinks(updatedLinks)
21   }
22
23   const handleEnableLink = (socialNetwork: string) => {
24     //console.log(socialNetwork)
25     //const updatedLinks = devTreeLinks.map(link => link.name === socialNetwork ? {...link, enabled: !link.enabled} : link)
26
27     const updatedLinks = devTreeLinks.map(link => {
28       if(link.name === socialNetwork){
29         if(!isValidUrl(link.url)){
30           return {...link, enabled: !link.enabled}
31         }else{
32           toast.error('URL no válida')
33         }
34       }
35       return link
36     })
37

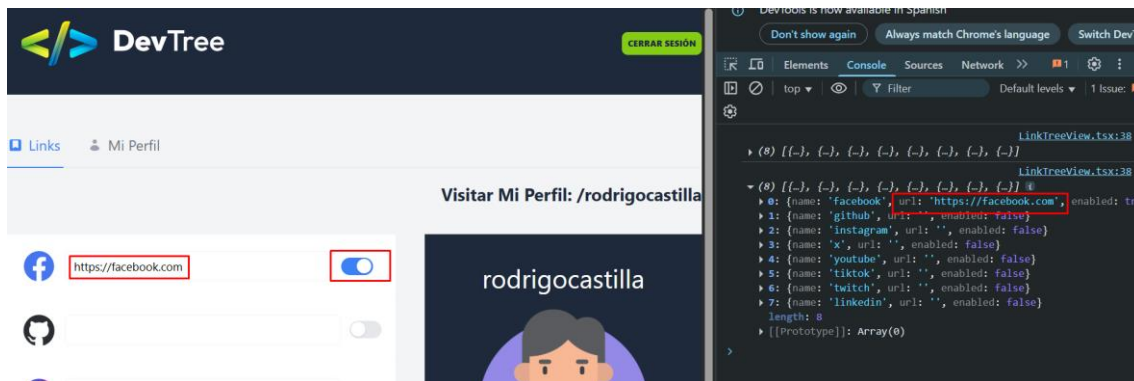
```



```

LinkTreeView.tsx • TS index.ts ...utils • DevTreeInput.tsx TS index.ts ...types ProfileView.tsx
frontend > src > views > LinkTreeView.tsx > LinkTreeView > handleUrlChange
4 import { isValidUrl } from "../utils"
5 import { toast } from "sonner"
6
7 export default function LinkTreeView(){
8   const [devTreeLinks, setDevTreeLinks] = useState(social)
9
10   const handleUrlChange = (e : React.ChangeEvent<HTMLInputElement>) => {
11
12     // Mapea cada elemento del estado actual
13     // Si el nombre del enlace coincide con el del input que generó el evento,
14     // actualiza su propiedad `url` con el nuevo valor ingresado
15     const updatedLinks = devTreeLinks.map(link => link.name === e.target.name ? {
16
17       //console.log(updatedLinks)
18
19     // Actualiza el estado con la nueva lista de enlaces
20     setDevTreeLinks(updatedLinks)
21   }

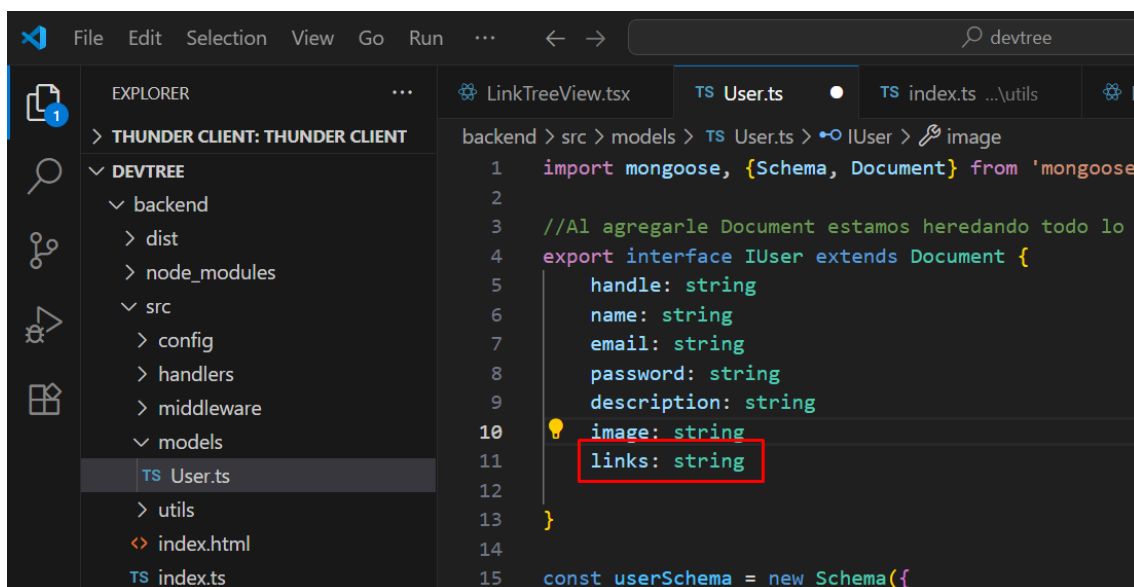
```

Añadiendo links a nuestro modelo de usuarios

Necesitamos ir almacenando los enlaces a nuestra base de datos.

Vamos a hacer primero la mutación.



Si tienes experiencia en mongoDB, sabrás que también te permite guardar arreglos como subdocumentos, el problema es que le da un ID que es muy similar a los ID de MongoDB y en el caso de la librería que estaremos utilizando para el drag and drop requiere que el ID sea consecutivo, es decir, 12345... Entonces lo vamos a guardar como un string y básicamente lo que vamos a hacer es tomar todo ese arreglo y convertirlo a string. Y una vez que obtengamos los datos desde la BD lo estaremos convirtiendo de nuevo a arreglo.

The screenshot shows the VS Code interface with the Explorer on the left displaying the project structure. The file `TS User.ts` is selected. The main editor shows the definition of `userSchema` in `models > TS User.ts`. A red box highlights the `links` property definition:

```

15  const userSchema = new Schema({
36    password: {
37      type: String,
38      required: true,
39      trim: true
40    },
41    description: {
42      type: String,
43      default: ''
44    },
45    image: {
46      type: String,
47      default: ''
48    },
49    links: {
50      type: String,
51      default: '[]'
52    }
53  })
54
55

```

The screenshot shows the VS Code interface with the Explorer on the left displaying the project structure. The file `TS index.ts` is selected. The main editor shows the `updateProfile` function in `handlers > TS index.ts`. A red box highlights the `links` property being assigned to `req.user.links`:

```

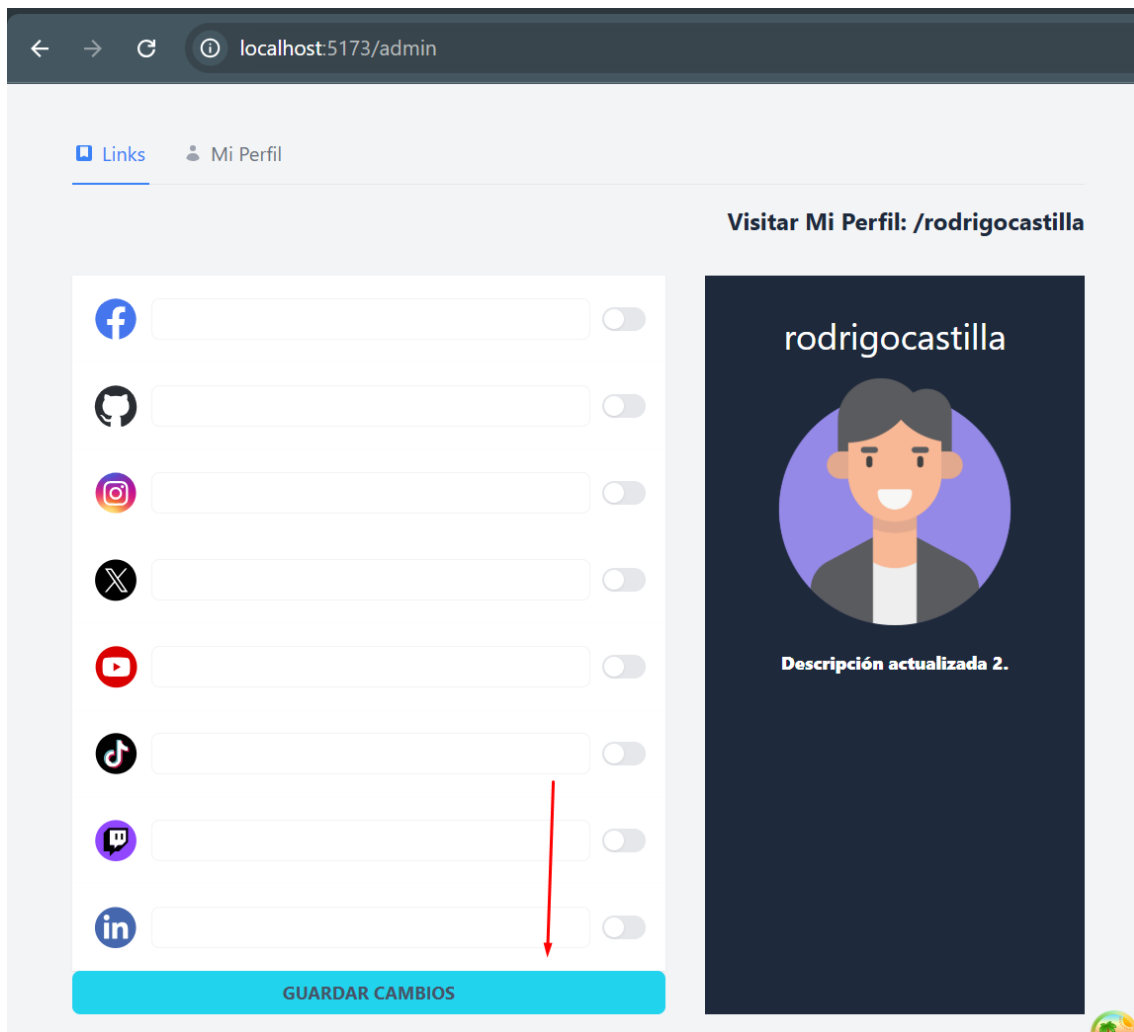
78  export const getUser = async(req: Request, res: Response) => {
79
80  }
81
82  export const updateProfile = async(req: Request, res: Response) => {
83    try{
84      //console.log(req.body)
85      const {description, links} = req.body
86
87      //copiamos de arriba: esto porque el usuario no puede utilizar un usuario que está ut
88      const handle = slug(req.body.handle, '')
89      const handleExists = await User.findOne({handle})
90      if(handleExists && handleExists.email !== req.user.email){
91        const error = new Error('Nombre de usuario no disponible')
92        return res.status(409).json({error: error.message})
93      }
94
95      //Actualizar el usuario
96      req.user.description = description
97      req.user.handle = handle
98      req.user.links = links
99
100     await req.user.save()
101     res.send('Perfil actualizado correctamente')

```

Crear la mutación y mandarla a llamar:

```
LinkTreeView.tsx X TS User.ts TS index.ts TS DevTreeAPI.ts
frontend > src > views > LinkTreeView.tsx > LinkTreeView
4 import { isValidUrl } from "../utils"
5 import { toast } from "sonner"
6 import { useMutation } from "@tanstack/react-query"
7 import { updateProfile } from "../api/DevTreeAPI"
8
9 export default function LinkTreeView(){
10   const [devTreeLinks, setDevTreeLinks] = useState(social)
11
12   const {} = useMutation({
13     mutationFn: updateProfile,
14     onError: (error) => {
15       toast.error(error.message)
16     },
17     onSuccess: () => {
18       toast.success('Actualizado Correctamente')
19     }
20   })
21
```

```
LinkTreeView.tsx 1 TS User.ts TS index.ts TS DevTreeAPI.ts
frontend > src > views > LinkTreeView.tsx > LinkTreeView
4 import { isValidUrl } from "../utils"
5 import { toast } from "sonner"
6 import { useMutation } from "@tanstack/react-query"
7 import { updateProfile } from "../api/DevTreeAPI"
8
9 export default function LinkTreeView(){
10   const [devTreeLinks, setDevTreeLinks] = useState(social)
11
12   const { mutate } = useMutation({
13     mutationFn: updateProfile,
14     onError: (error) => {
15       toast.error(error.message)
16     },
17     onSuccess: () => {
18       toast.success('Actualizado Correctamente')
19     }
20   })
21
```



Almacenando los enlaces en la base de datos.

Tenemos que actualizar links o perfiles y tenemos que utilizar el mismo método de nuestro backend, hay que ajustar el código.

```

frontend > src > views > ProfileView.tsx > ProfileView
8  export default function ProfileView() {
54  const handleChange = (e: React.ChangeEvent<HTMLInputElement>) => {
58  }
59  }
60
61  const handleUserProfileForm = (formData: ProfileForm) => {
62    const user : User = queryClient.getQueryData(['user'])!
63    user.description = formData.description //va a tener el último valor del formulario en caso cambie
64    user.handle = formData.handle
65    updateProfileMutation.mutate(user)
66  }
67
68  return (
69    <form
70      className="bg-white p-10 rounded-lg space-y-5"
71      onSubmit={handleSubmit(handleUserProfileForm)}
72    >
73    <legend className="text-2xl text-slate-800 text-center">Editar Información</legend>

```

```

LinkTreeView.tsx 1 • ProfileView.tsx TS DevTreeAPI.ts TS Users.ts TS index.ts
frontend > src > views > LinkTreeView.tsx > LinkTreeView
9  export default function LinkTreeView(){
35  const handleEnableLink = (socialNetwork: string) => {
48  }
49
50  console.log(updatedLinks)
51  setDevTreeLinks(updatedLinks)
52  }
53
54  return(
55    <div className="space y-5">
56      {devTreeLinks.map(item=>{
57        <DevTreeInput
58          key={item.name}
59          //pasamos el item hacia el componente
60          item={item}
61          handleUrlChange={handleUrlChange}
62          handleEnableLink={handleEnableLink}
63        />
64      })}
65    <button
66      className="bg-cyan-400 p-2 text-lg w-full uppercase text-slate-600 rounded-lg font-bold"
67      onClick={()=>mutate()} //colocamos un callback para que espere por ese evento
68    >Guardar cambios</button>
69  </div>
70  )
71  }

```

```

LinkTreeView.tsx 2 • ProfileView.tsx TS DevTreeAPI.ts TS Users.ts TS index.ts
frontend > src > views > LinkTreeView.tsx > LinkTreeView
1  import { useState } from "react"
2  import { social } from "../data/social"
3  import DevTreeInput from "../components/DevTreeInput"
4  import { isValidUrl } from "../utils"
5  import { toast } from "sonner"
6  import { useMutation, useQueryClient } from "@tanstack/react-query"
7  import { updateProfile } from "../api/DevTreeAPI"
8  import type { User } from "../types"
9
10 export default function LinkTreeView(){
11   const [devTreeLinks, setDevTreeLinks] = useState(social)
12
13   const queryClient = useQueryClient()
14   const user : User = queryClient.getQueryData(['user'])!
15
16   const { mutate } = useMutation({
17     mutationFn: updateProfile,

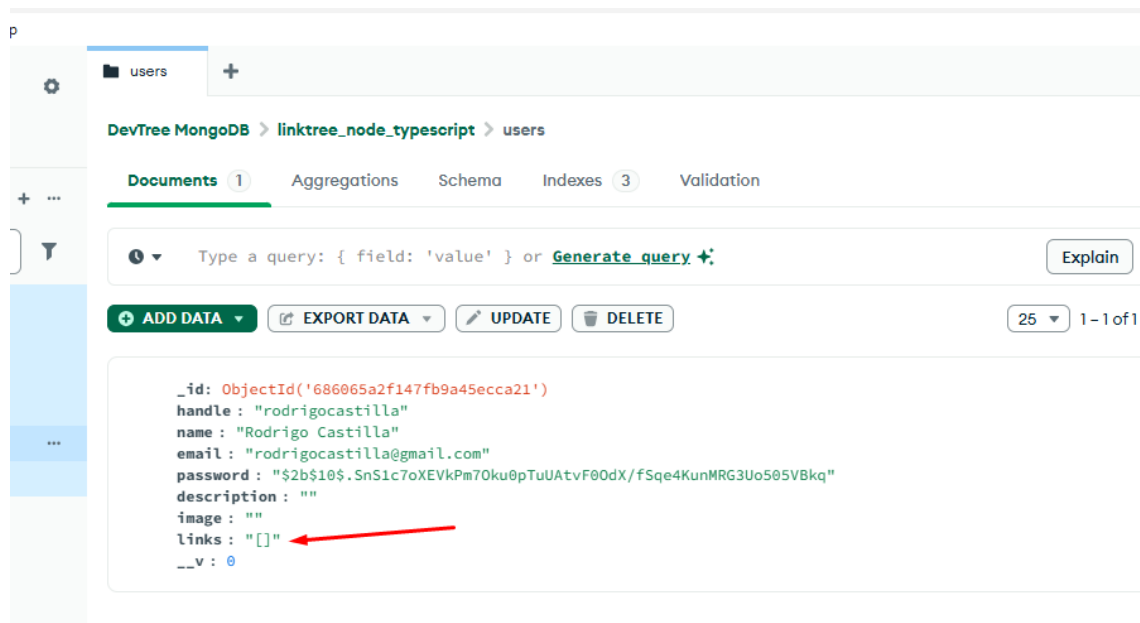
```

```

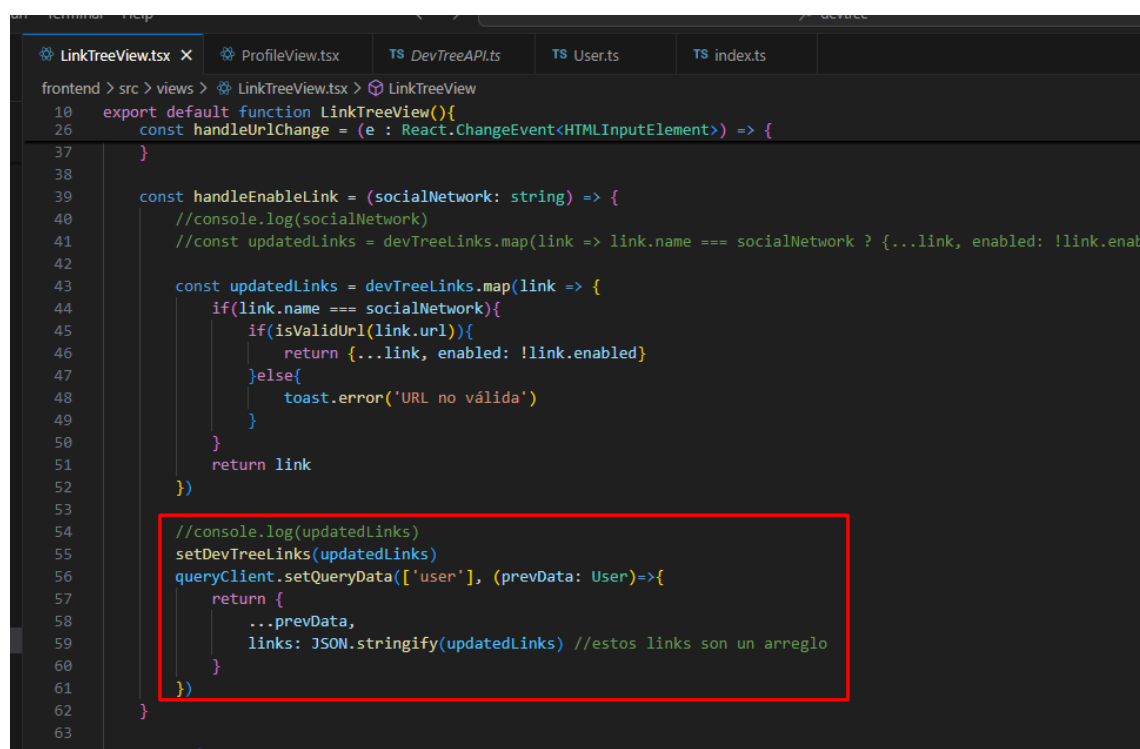
Terminal Help
LinkTreeView.tsx x ProfileView.tsx TS DevTreeAPI.ts TS Users.ts TS index.ts
frontend > src > views > LinkTreeView.tsx > LinkTreeView > devTreeLinks.map() callback
10  export default function LinkTreeView(){
60  {devTreeLinks.map(item=>{
61    <DevTreeInput
62      key={item.name}
63      //pasamos el item hacia el componente
64      item={item}
65      handleUrlChange={handleUrlChange}
66      handleEnableLink={handleEnableLink}
67    />
68  })}
69  <button
70    className="bg-cyan-400 p-2 text-lg w-full uppercase text-slate-600 rounded-lg font-bold"
71    onClick={()=>mutate(user)} //colocamos un callback para que espere por ese evento
72  >Guardar cambios</button>
73  </div>
74  )
75  }

```

Ahora, tenemos un problema y ese problema es que, si pegamos un enlace en Facebook, y guardo cambios, actualiza correctamente, pero en compass, todo está con un arreglo vacío.



Y esto pasa porque en ningún momento estamos pasándole los enlaces a la versión cacheada. Los enlaces están en el state. ¿Cómo lo podemos escribir?



Entonces, una vez que pasa las validaciones lo escribimos en el objeto del usuario que está autenticado. Y si guardo los cambios, vamos a mutar esos cambios que están actualizados.

The screenshot shows the DevTree application interface. At the top, there's a dark header with the DevTree logo and a 'CERRAR SESIÓN' button. Below the header, there's a navigation bar with 'Links' and 'Mi Perfil' tabs. The main content area is titled 'Visitar Mi Perfil: /rodrigocastilla'. On the left, there's a form with social media links. The first link, for Facebook, is highlighted with a red box and has its toggle switch turned on. Below it are input fields for GitHub and Instagram, both with their toggle switches turned off. On the right, there's a profile card for 'rodrigocastilla' with a placeholder image. At the bottom, there's a console window showing a JSON response from a query. The response includes fields like _id, handle, name, email, password, description, image, and links. A red arrow points to the 'links' field in the JSON, which contains an array of objects representing social media links.

DevTree

CERRAR SESIÓN

Links Mi Perfil

Visitar Mi Perfil: /rodrigocastilla

https://facebook.com

rodrigocastilla

type a query: { field: 'value' } or [generate query](#) Explain

ADD DATA EXPORT DATA UPDATE DELETE 25 1 - 1 of 1

```

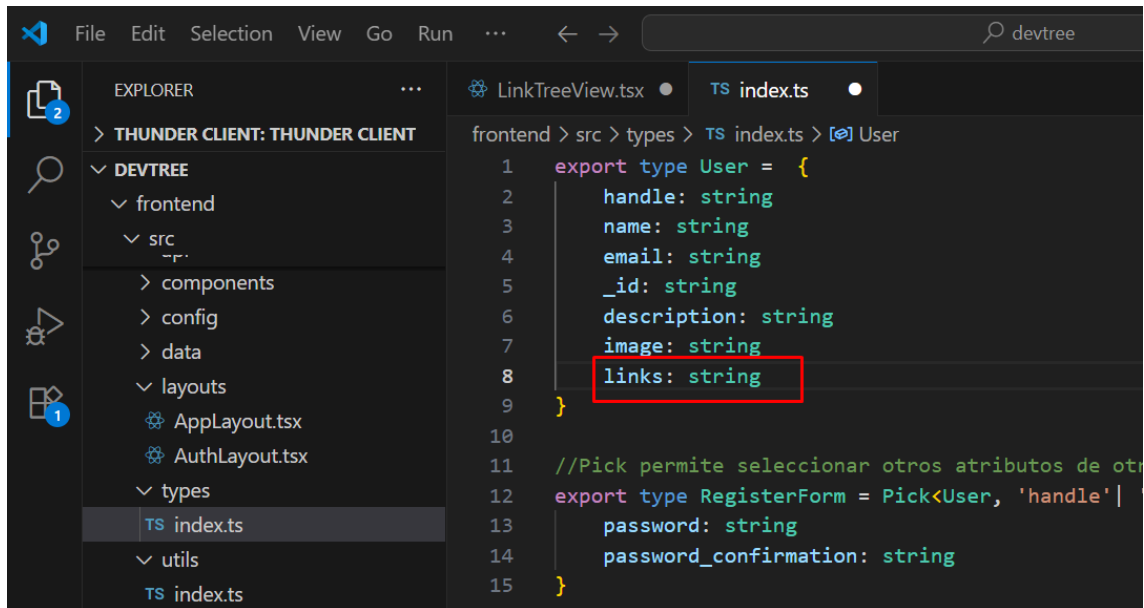
_id: ObjectId('686065a2f147fb9a45ecca21')
handle: "rodrigocastilla"
name: "Rodrigo Castilla"
email: "rodrigocastilla@gmail.com"
password: "$2b$10$.SnS1c7oXEKvPm70ku0pTuUAtvF00dX/fSq4KunMRG3Uo505VBkq"
description: "Rodrigo Castilla es Ingeniero de Sistemas"
image: "https://res.cloudinary.com/dxvaf2cz4/image/upload/v1751148138/a4399675..."
links: [{"name": "facebook", "url": "https://facebook.com", "enabled": true}, {"name": "github", "url": "https://github.com", "enabled": false}, {"name": "instagram", "url": "https://instagram.com", "enabled": false}]
__v: 0

```

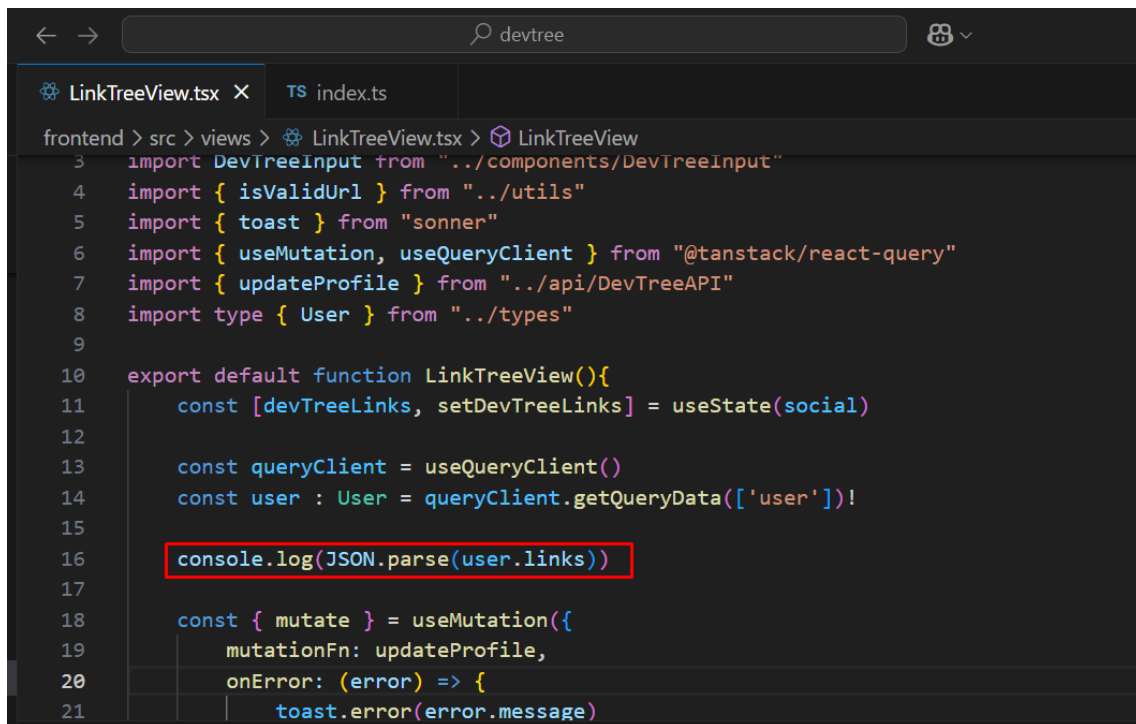
¿Que hemos hecho bien? Bueno, en consola vemos que se ha guardado la url y que el enabled está bien, y en devtreeapi reutilizamos la misma función para profileview y linkstreeview. Pero hay dos detalles, si recargamos no aparece el link que el usuario almacenó y tampoco se agregan debajo de la foto de perfil.

Recuperando los enlaces de la base de datos y llenando el formulario

En esta parte estaremos viendo cómo seleccionar una vez que yo visito esta página y llenar los inputs con lo que tenemos en la base de datos. Habíamos visto que los links vienen como string y tenemos que convertirlos a un arreglo.

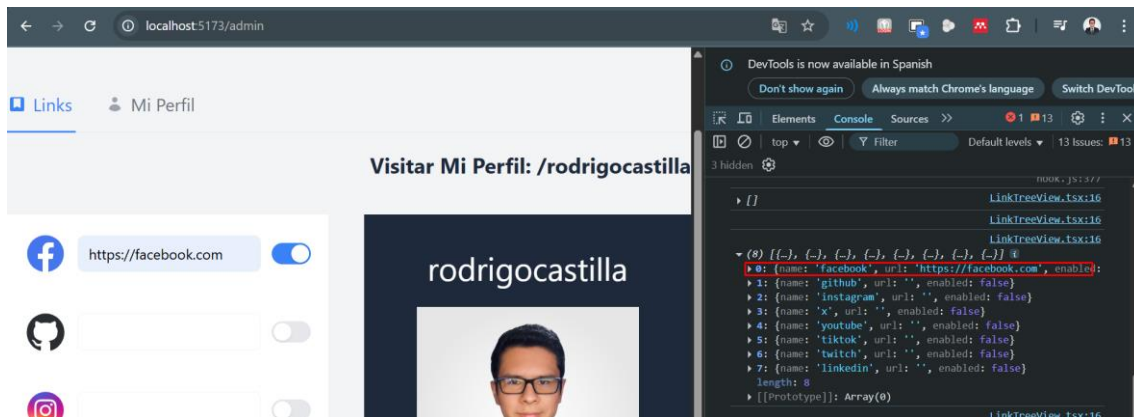


Con eso agregamos el type de links.

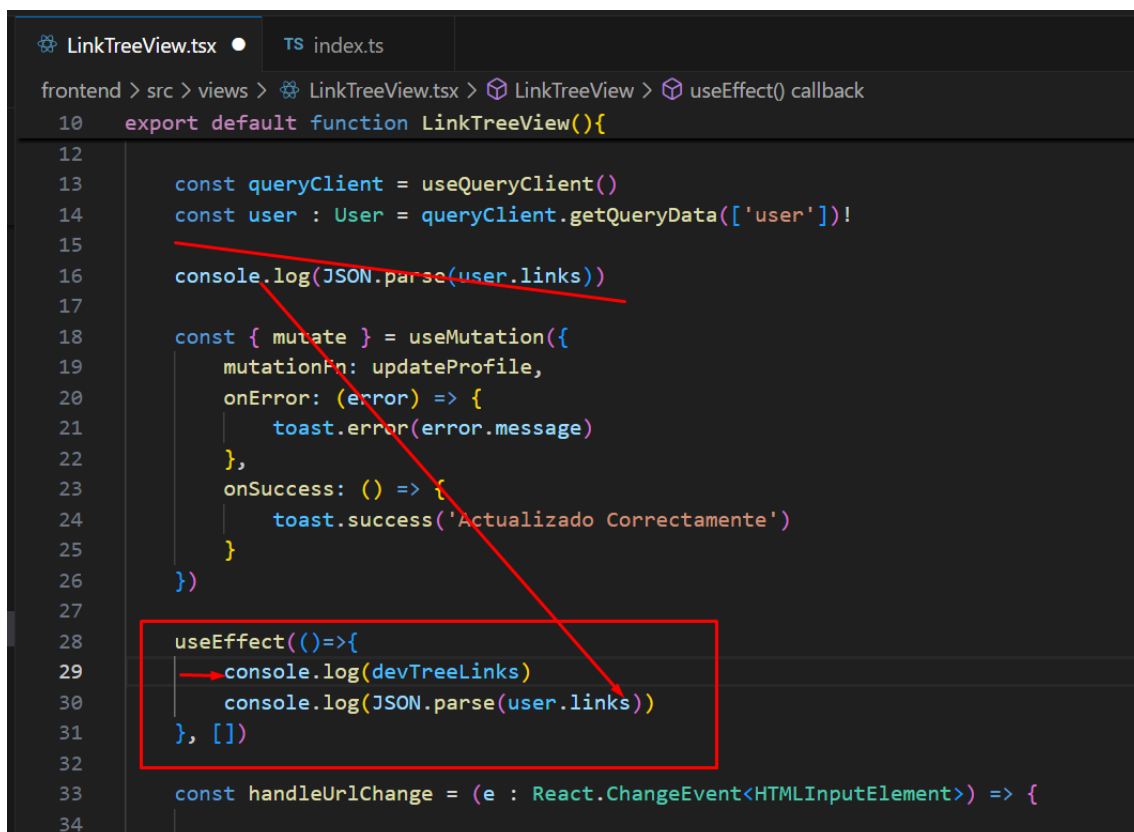


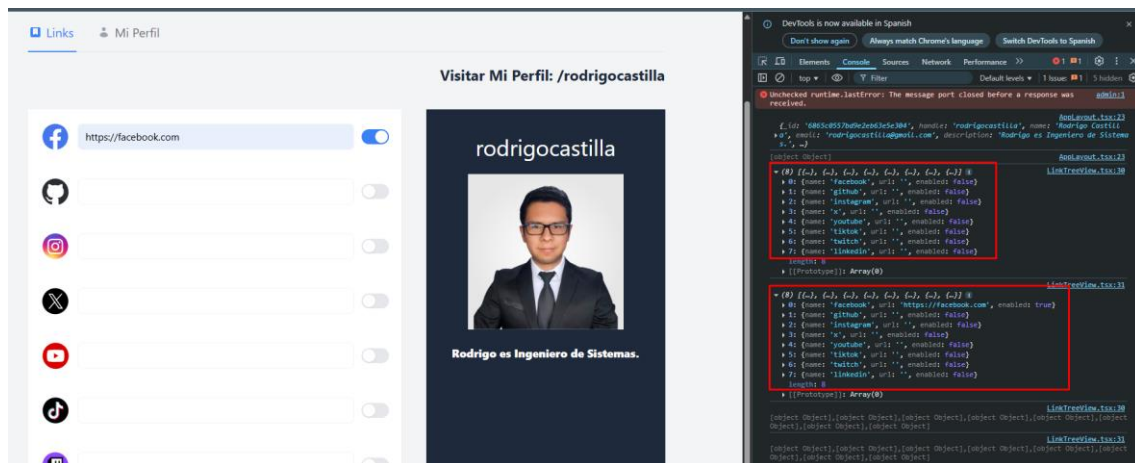
Ya tenemos la URL

Si recargamos, vemos que tenemos el acceso a todas las redes sociales.

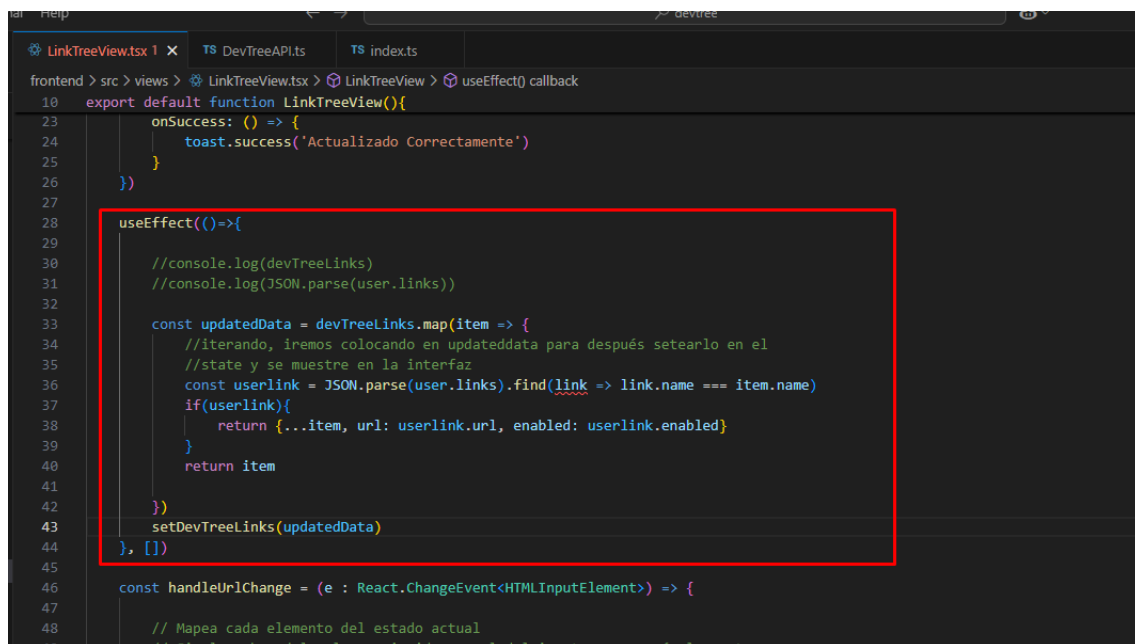


El siguiente paso es detectar cuáles de estas redes sociales ya tienen la información para llenar en automático los inputs del formulario. Entonces, lo que vamos a hacer es, después del `UseMutation` vamos a colocar un `UseEffect` y vamos a pasarle un arreglo de dependencias que esté vacío para que se ejecute una vez cuando el componente esté listo. Es decir cuando cambio de pestaña de MiPerfil a Links aquí queremos que se ejecute.





Entonces con eso tenemos los campos de las redes sociales, pero también tenemos los enlaces agregados del usuario. Y lo que tenemos que hacer es iterar en ambas e ir buscando cuáles ya tienen información, cuáles ya el usuario agregó algo.



Interface for managing social media links. It includes a list of social media icons (Facebook, GitHub, Instagram, X, YouTube, TikTok, Twitch, LinkedIn) with input fields for their URLs and toggle switches to enable or disable them. The Facebook and Instagram entries are highlighted with red boxes, showing their URLs and active toggle switches. A blue button labeled 'GUARDAR CAMBIOS' is at the bottom.

Ahora sí está sincronizado, hacemos la prueba agregando redes sociales, habilitando, deshabilitando, guardando cambios.

Sin embargo, si agregamos un enlace a una red social y no la habilitamos, no se va a guardar. Esto pasa porque el caché no se está actualizando.

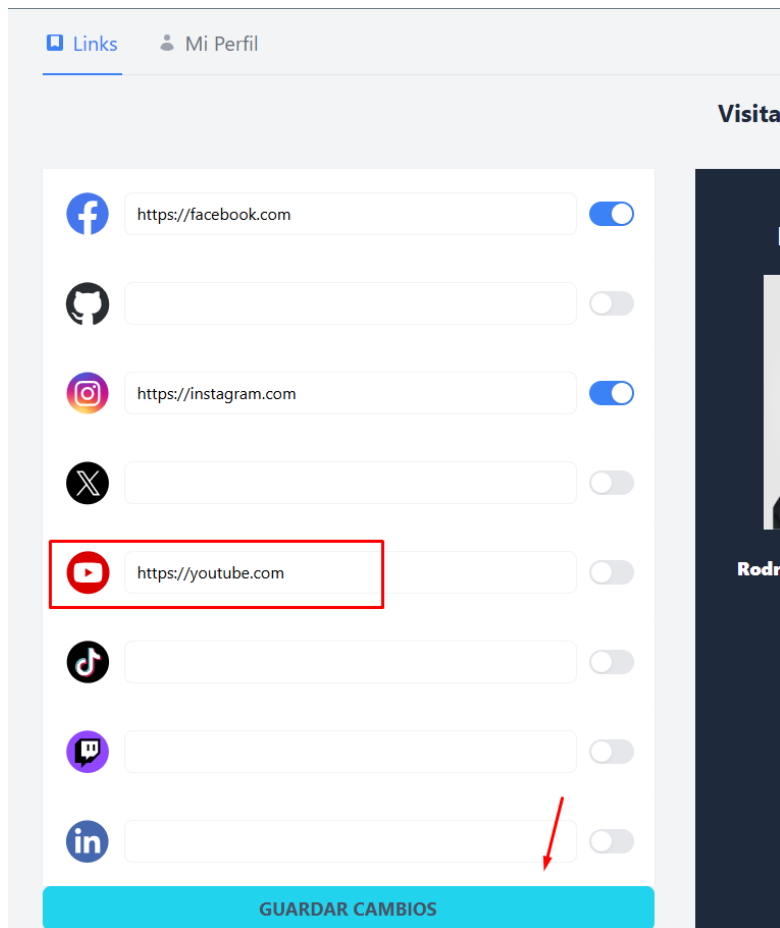
```

10 export default function LinkTreeView() {
28   useEffect(() => {
33     const updatedData = devTreeLinks.map(item => {
42     })
43     setDevTreeLinks(updatedData)
44   }, [])
45
46   const handleUrlChange = (e : React.ChangeEvent<HTMLInputElement>) => {
47
48     // Mapea cada elemento del estado actual
49     // Si el nombre del enlace coincide con el del input que generó el evento,
50     // actualiza su propiedad 'url' con el nuevo valor ingresado
51     const updatedLinks = devTreeLinks.map(link => link.name === e.target.name ? {...link, url: e.target.value} : link)
52
53     //console.log(updatedLinks)
54
55     // Actualiza el estado con la nueva lista de enlaces
56     setDevTreeLinks(updatedLinks)
57
58     queryClient.setQueryData(['user'], (prevData: User) => {
59       return {
60         ...prevData,
61         links: JSON.stringify(updatedLinks) //estos links son un arreglo
62       }
63     })
64   }
65 }

```

De esta forma cuando haya un cambio en la URL se debe ir escribiendo en el caché del usuario.

Ahora sí se mantiene:



Pero, no se puede habilitar una URL si no es válida. Hagan la prueba.

Mostrando los enlaces habilitados en nuestro perfil

En este video estaremos viendo cómo mostrar de este lado los enlaces para tener un preview de cómo sería nuestro perfil con enlaces.

Vemos que “links” es de tipo Any. Si es la primera vez que trabajas con TypeScript, debes saber que, se limita únicamente al código pero no puede inferir un string que tiene un arreglo, tampoco algo de localStorage ni el resultado de una API. Esto se puede solucionar con otras herramientas.

En este caso, le decimos que link va a ser SocialNetwork.

```

useEffect(()=>{

  //console.log(devTreeLinks)
  //console.log(JSON.parse(user.links))

  const updatedData = devTreeLinks.map(item => {
    //iterando, iremos colocando en updateddata para después setearlo en el
    //state y se muestre en la interfaz
    const userlink = JSON.parse(user.links) find((link: SocialNetwork) => link.name === item.name)
    if(userlink){
      return {...item, url: userlink.url, enabled: userlink.enabled}
    }
    return item
  })
  setDevTreeLinks(updatedData)
}, [])

```

Mostraremos únicamente los enlaces habilitados en la sección del perfil.

```

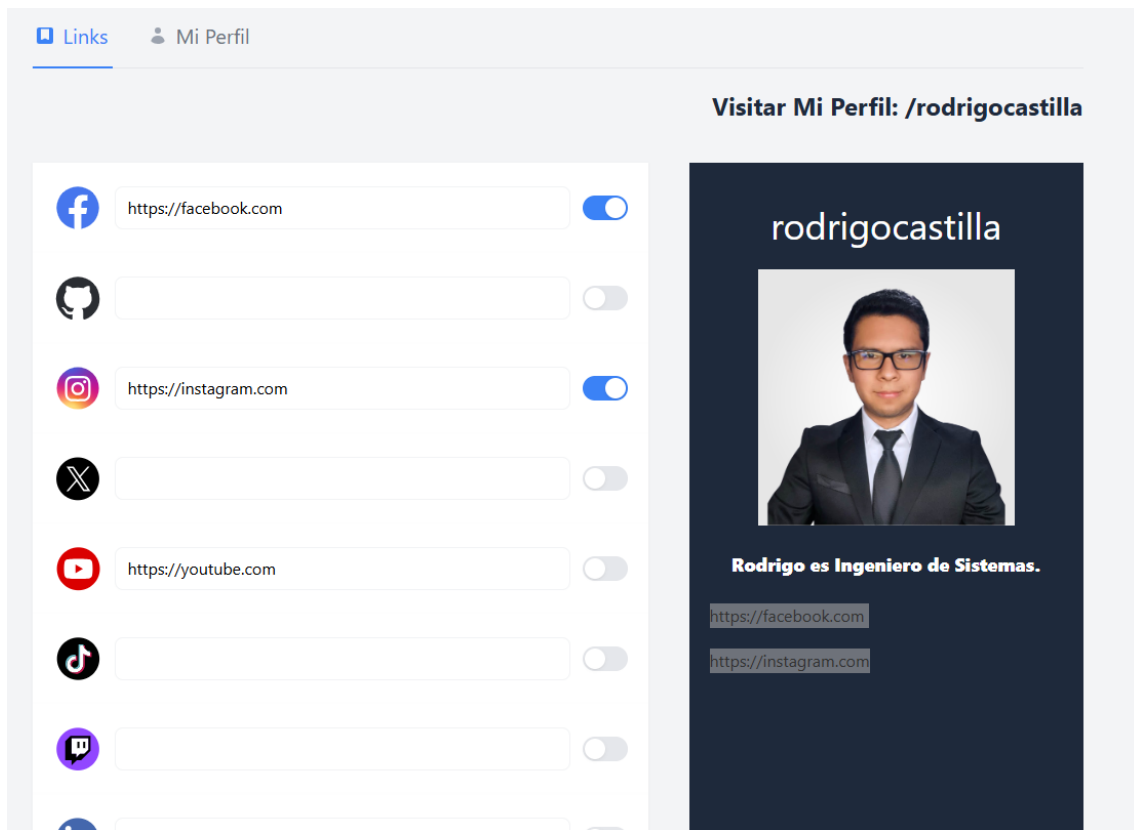
1 import NavigationTabs from "../components/NavigationTabs";
2 import { Link, Outlet } from "react-router-dom";
3 import { Toaster } from "sonner";
4 import type { SocialNetwork, User } from "../types";
5 import { useState } from "react";
6
7 type DevTreeProps = {
8   data: User
9 }
10
11 export default function DevTree({data}: DevTreeProps){
12
13   const [enabledLinks, setEnabledLinks] = useState<SocialNetwork[]>(JSON.parse(data.links).filter((item: SocialNetwork) => item.enabled))
14
15   //console.log(enabledLinks)
16   //Para mostrar los links activos: console.log(JSON.parse(data.links).filter(item => item.enabled))
17
18   return(

```

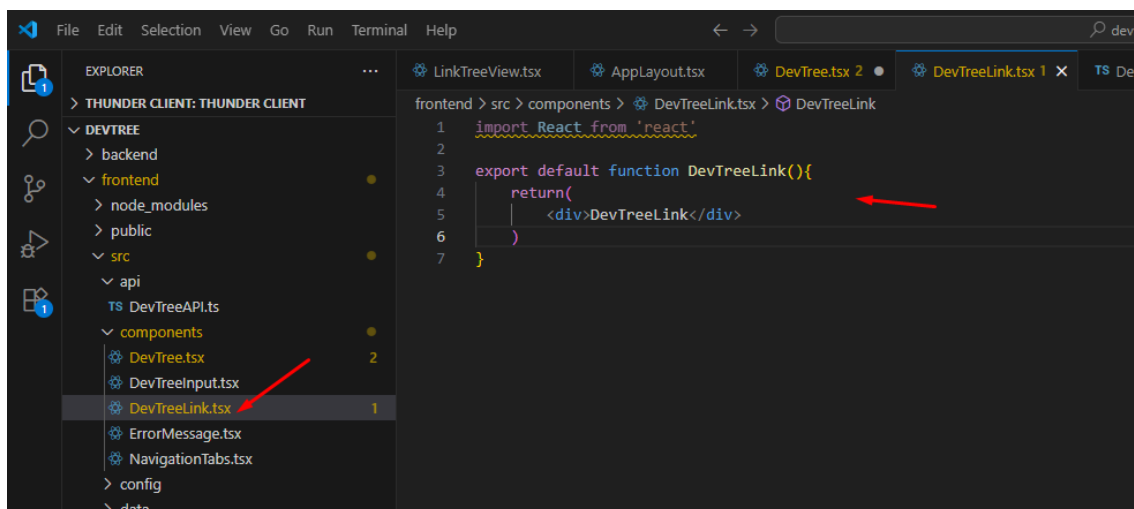
```

11 export default function DevTree({data}: DevTreeProps){
44   target="_blank"
45   rel="noopener noreferrer"
46   >Visitar Mi Perfil: {data.handle}</Link>
47 </div>
48
49 <div className="flex flex-col md:flex-row gap-10 mt-10">
50   <div className="flex-1">
51     <Outlet />
52   </div>
53   <div className="w-full md:w-96 bg-slate-800 px-5 py-10 space-y-6">
54     <p className="text-4xl text-center text-white">{data.handle}</p>
55
56     {data.image && //si tenemos una imagen, la renderizamos, pero si no:
57     <img src={data.image} alt="Imagen Perfil" className="mx-auto max-w-[250px]"></img>
58     }
59
60     <p className="text-center text-lg font-black text-white">{data.description}</p>
61
62     <div className="mt-20 flex flex-col gap-5">
63       {enabledLinks.map(link => (
64         <p>{link.url}</p>
65       ))}
66     </div>
67   </div>
68 </div>
69 </main>
70 </div>
71 <Toaster position="top-right" />
72 </>
73
74 )
75 }

```



Añadiendo CSS a los Enlaces habilitados de nuestro perfil



Tenemos que pasarle el key, le pasamos el name que será único.

```

LinkTreeView.tsx  AppLayout.tsx  DevTree.tsx 1  DevTreeLink.tsx 1  TS DevTreeAPI.ts  TS index.ts
frontend > src > components > DevTree.tsx > DevTree
12 export default function DevTree({data}: DevTreeProps){
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50 <div className="flex flex-col md:flex-row gap-10 mt-10">
51   <div className="flex-1">
52     <Outlet />
53   </div>
54   <div className="w-full md:w-96 bg-slate-800 px-5 py-10 space-y-6">
55     <p className="text-4xl text-center text-white">{data.handle}</p>
56
57     {data.image && //si tenemos una imagen, la renderizamos, pero si no:
58     <img src={data.image} alt="Imagen Perfil" className="mx-auto max-w-[250px]" />
59   }
60
61   <p className="text-center text-lg font-black text-white">{data.description}</p>
62
63   <div className="mt-20 flex flex-col gap-5">
64     {enabledLinks.map(link => (
65       <DevTreeLink key={link.name} link={link} />
66     ))}
67   </div>
68 </div>
69
70 </div>
71 </div>
72 </main>

```

```

LinkTreeView.tsx  AppLayout.tsx  DevTree.tsx 1  DevTreeLink.tsx 1  TS DevTreeAPI.ts  TS index.ts
frontend > src > components > DevTreeLink.tsx > DevTreeLink
1 import type { SocialNetwork } from "../types"
2
3 type DevTreeLinkProps = {
4   link: SocialNetwork
5 }
6
7 export default function DevTreeLink({link}: DevTreeLinkProps){
8   return(
9     <div>DevTreeLink</div>
10  )
11 }

```

```


LinkTreeView.tsx  AppLayout.tsx  DevTree.tsx 1  DevTreeLink.tsx 1  DevTreeInput.tsx  TS DevTreeAPI.ts  TS index.ts
frontend > src > components > DevTreeLink.tsx > DevTreeLink
1 import type { SocialNetwork } from "../types"
2
3 type DevTreeLinkProps = {
4   link: SocialNetwork
5 }
6
7 export default function DevTreeLink({link}: DevTreeLinkProps){
8   return(
9     <li className="bg-white px-5 py-2 flex items-center gap-5 rounded-lg">
10       <div
11         className="w-12 h-12 bg-cover"
12         style={{ backgroundImage: `url('/social/icon_${link.name}.svg')` }}
13       >></div>
14       <p className="capitalize">Visita mi: <span className="font-bold">{link.name}</span></p>
15     </li>
16   )
17 }

```


Links


Mi Perfil


Visitar Mi Perfil: /rodrigocastilla




https://facebook.com













https://instagram.com














https://youtube.com











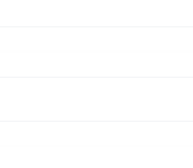







GUARDAR CAMBIOS


rodrigocastilla



Rodrigo es Ingeniero de Sistemas.



Visita Mi: **Facebook**



Visita Mi: **Instagram**

Si guardo cambios actualiza correctamente, pero debo de actualizar la página para que se muestre el cambio. Hay que darle un comportamiento en tiempo real.

Live Preview de redes sociales habilitadas/deshabilitadas

```

1 import NavigationTabs from "../components/NavigationTabs";
2 import { Link, Outlet } from "react-router-dom";
3 import { Toaster } from "sonner";
4 import type { SocialNetwork, User } from "../types";
5 import { useEffect, useState } from "react";
6 import DevTreeLink from "../DevTreeLink";
7
8 type DevTreeProps = {
9   data: User
10 }
11
12 export default function DevTree({data}: DevTreeProps){
13
14   const[enabledLinks, setEnabledLinks] = useState<SocialNetwork[]>(JSON.parse(data.links).filter((item: SocialNetwork) => item.enabled))
15
16   useEffect(()=>{
17     setEnabledLinks(JSON.parse(data.links).filter((item: SocialNetwork) => item.enabled))
18   }, [data])
19
20   //console.log(enabledLinks)
21   //Para mostrar los links activos: console.log(JSON.parse(data.links).filter(item => item.enabled))

```