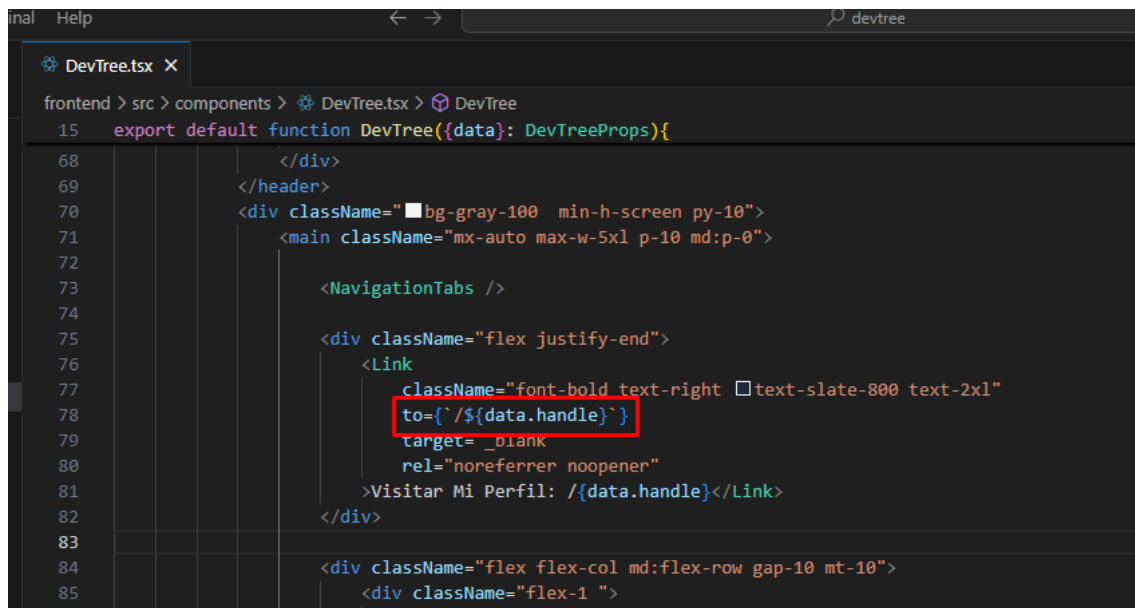


Creando el enlace para compartir tu DevTree



```
15 export default function DevTree({data}: DevTreeProps){
68   </div>
69   </header>
70   <div className="bg-gray-100 min-h-screen py-10">
71     <main className="mx-auto max-w-5xl p-10 md:p-0">
72       <NavigationTabs />
73       <div className="flex justify-end">
74         <Link
75           className="font-bold text-right text-slate-800 text-2xl"
76           to={`/${data.handle}`}
77           target=_blank
78           rel="noreferrer noopener"
79         >Visitar Mi Perfil: {data.handle}</Link>
80       </div>
81       <div className="flex flex-col md:flex-row gap-10 mt-10">
82         <div className="flex-1">
```

Al dar clic en “Visitar Mi Perfil”



Agregar diseño:

```

1 import {BrowserRouter, Routes, Route} from 'react-router-dom' //1. prepara todo para usar react
2 import LoginView from './views/LoginView'
3 import RegisterView from './views/RegisterView'
4 import AuthLayout from './layouts/AuthLayout'
5 import AppLayout from './layouts/AppLayout'
6 import LinkTreeView from './views/LinkTreeView'
7 import ProfileView from './views/ProfileView'
8 import HandleView from './views/HandleView'
9 export default function () {
10   return(
11     <BrowserRouter>
12       <Routes>
13         <Route element={<AuthLayout/>}>
14           <Route path='/auth/login' element={<LoginView/>}>
15           <Route path='/auth/register' element={<RegisterView/>}>
16         </Route>
17
18         <Route path='/admin' element={<AppLayout/>}>
19           <Route index={true} element={<LinkTreeView/>}></Route>
20           <Route path='profile' element={<ProfileView/>}></Route>
21         </Route>
22
23         <Route path='/:handle' element={<AuthLayout/>}>
24           <Route element={<HandleView/>} index={true}/>
25         </Route>
26       </Routes>
27     </BrowserRouter>
28   )
29 }

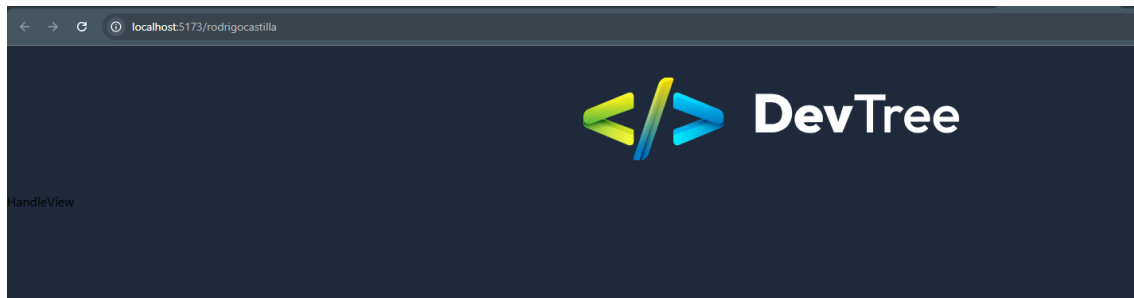
```

Creamos el HandleView.tsx

```

1 import React from 'react'
2
3 export default function HandleView(){
4   return(
5     <div>HandleView</div>
6   )
7 }

```

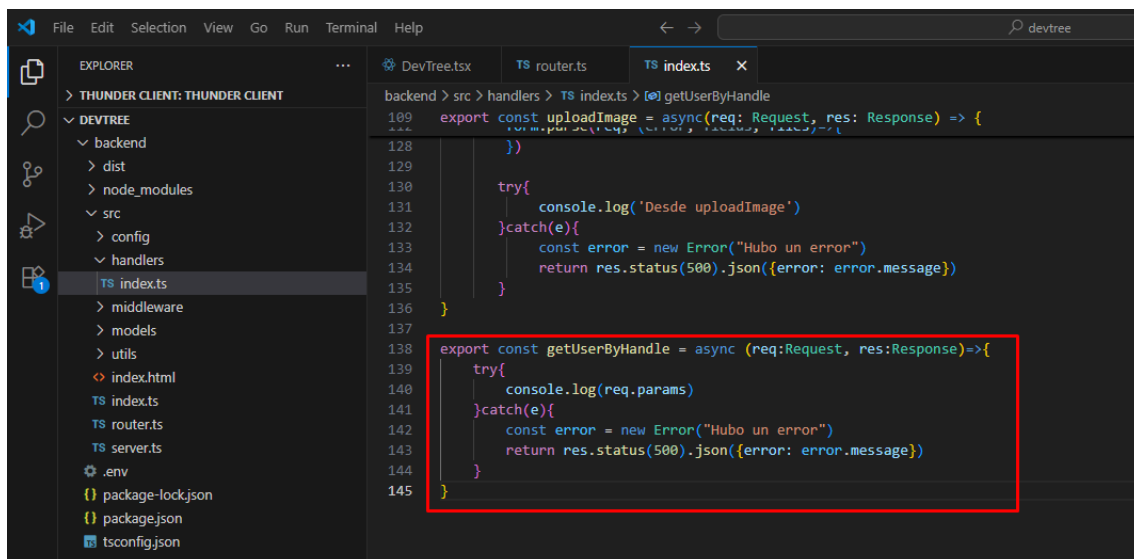


Ahora, tenemos que crear un nuevo endpoint de nuestra Rest API que consulte si ese usuario existe o no y entonces se traiga su información.

Registrando la ruta en el backend

Para visitar: /nombredeusuario

Volvemos al backend.

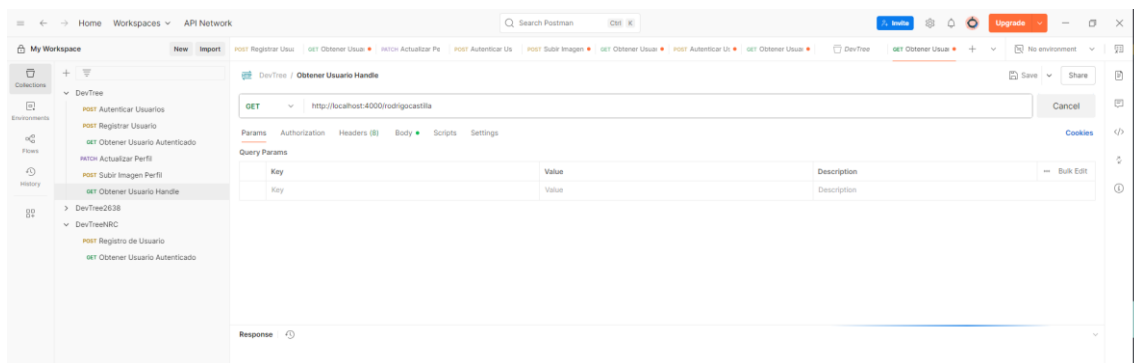


```

25     body('password')
26     .notEmpty()
27     .withMessage("El password es obligatorio"),
28
29     login
30 )
31
32
33 router.get('/user', authenticate, getUser)
34
35 //actualizar un registro en la BD: put y patch
36 //put: crea un nuevo elemento o reemplaza una representación del elemento de destino con los datos
37 //patch: aplica modificaciones parciales a un recurso (sería mejor porque solo vamos a cambiar el t
38
39 //para modificar un usuario necesitamos que esté autenticado por eso el autehtnciate.
40 router.patch('/user',
41   //validación:
42   body('handle').notEmpty().withMessage('El handle no puede ir vacío'),
43   body('description').notEmpty().withMessage('La descripción no puede ir vacía'),
44   //el middleware
45   handleInputErrors,
46   authenticate,
47   updateProfile)
48
49 router.post('/user/image', authenticate, uploadImage)
50
51 //routing dinámico
52 router.get('/:handle', getUserByHandle)
53
54 export default router
55
56

```

Enviamos una petición desde Postman:



En consola:

```

https://res.cloudinary.com/dxvz2ez4/image/upload/e_auto,g_auto,h_300,w_300/
[Object: null prototype] { handle: 'rodrigocastilla' }

```

```

109 export const uploadImage = async (req: Request, res: Response) => {
110   // ...
128 }
129
130 try{
131   console.log('Desde uploadImage')
132 }catch(e){
133   const error = new Error("Hubo un error")
134   return res.status(500).json({error: error.message})
135 }
136
137
138 export const getUserByHandle = async (req:Request, res:Response)=>{
139   try{
140     const {handle} = req.params
141     console.log(handle)
142   }catch(e){
143     const error = new Error("Hubo un error")
144     return res.status(500).json({error: error.message})
145   }
146 }
147

```

Entonces, teniéndolo en esa variable podemos consultar a la BD. Si el usuario existe, entonces mostramos su información, y si no existe, le decimos al usuario que ese perfil no existe.

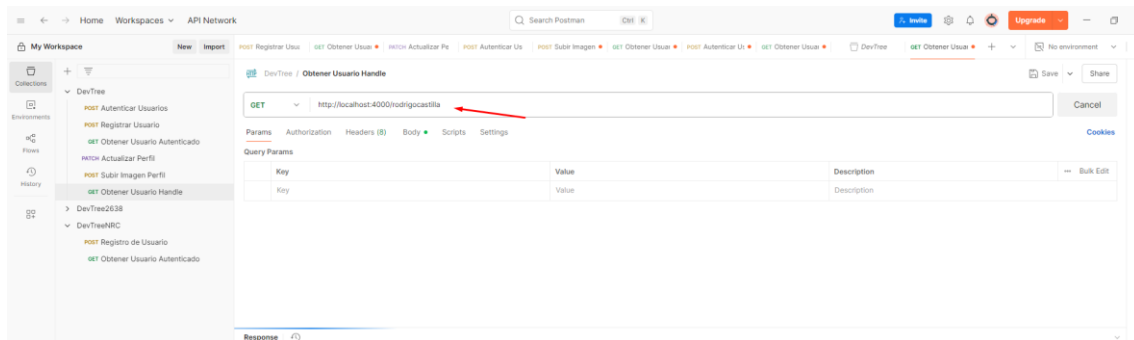
Consultando la base de datos y mostrando la información

```

109 export const uploadImage = async (req: Request, res: Response) => {
110   // ...
128 }
129
130 try{
131   console.log('Desde uploadImage')
132 }catch(e){
133   const error = new Error("Hubo un error")
134   return res.status(500).json({error: error.message})
135 }
136
137
138 export const getUserByHandle = async (req:Request, res:Response)=>{
139   try{
140     const {handle} = req.params
141     const user = await User.findOne({handle})
142     console.log({user})
143   }catch(e){
144     const error = new Error("Hubo un error")
145     return res.status(500).json({error: error.message})
146   }
147 }
148
149

```

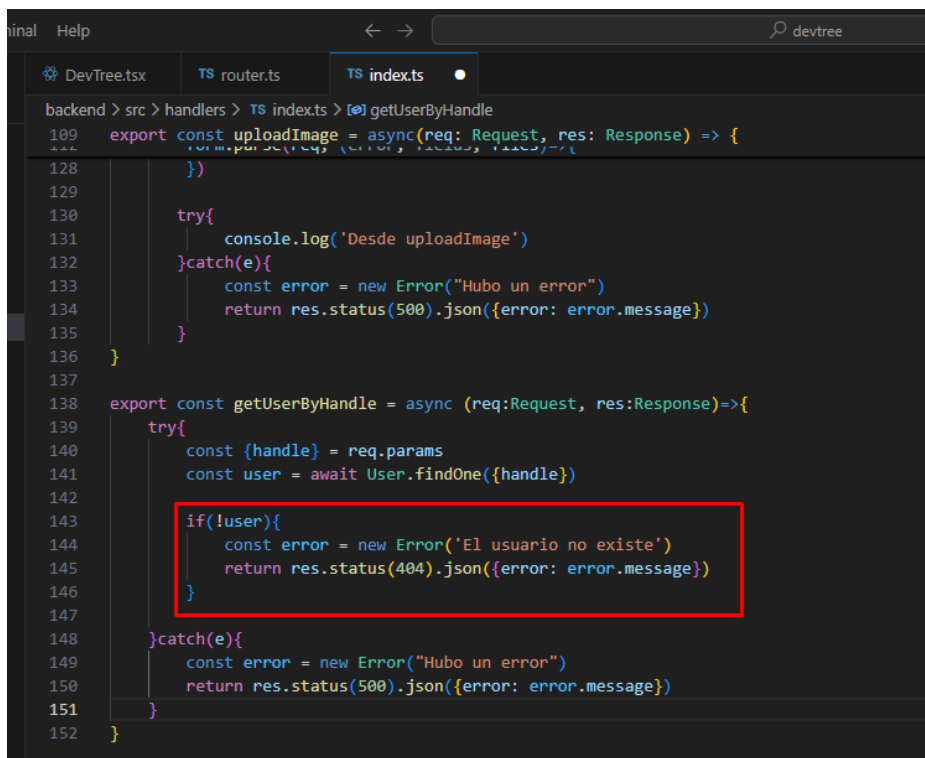
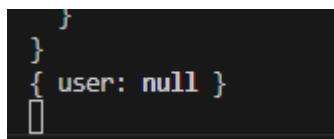
Si buscamos un usuario que existe:

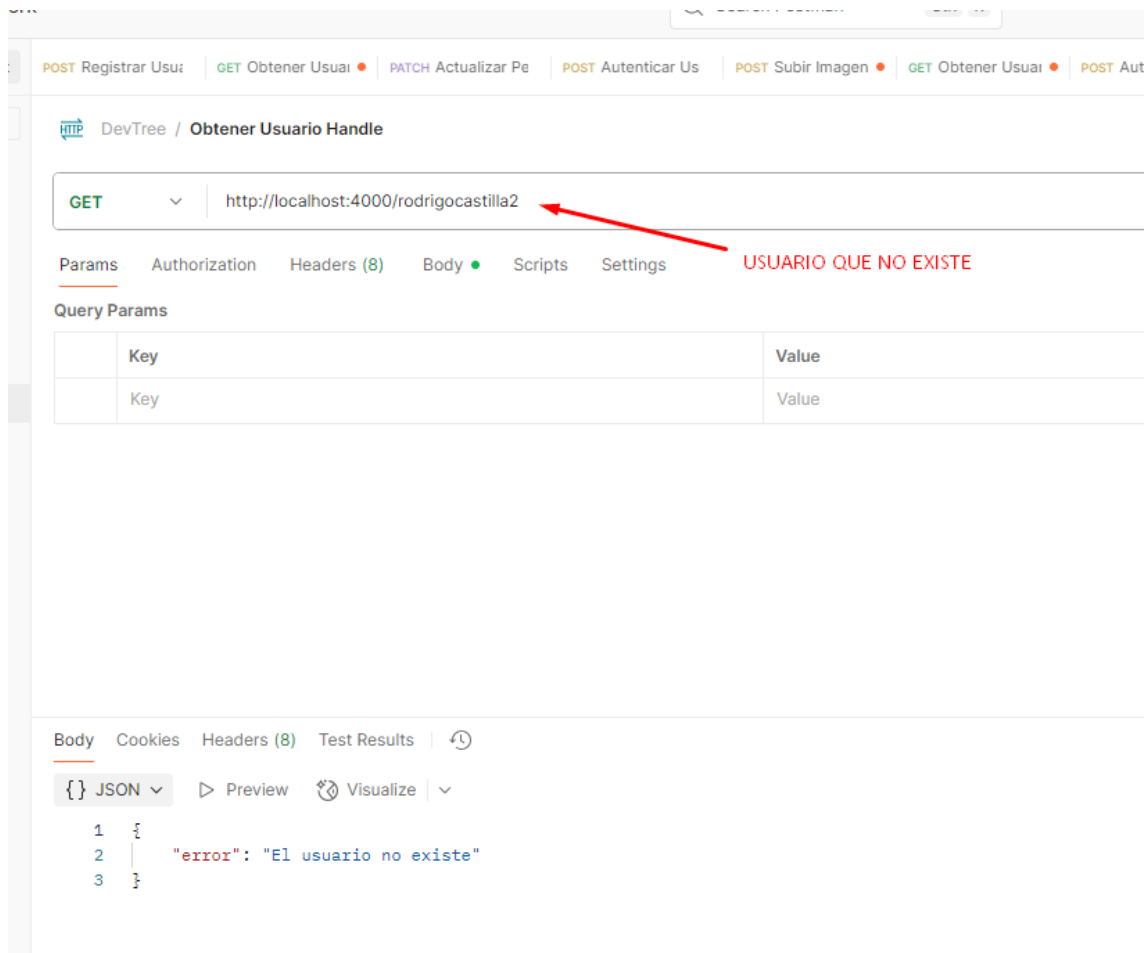


En consola tenemos su información.

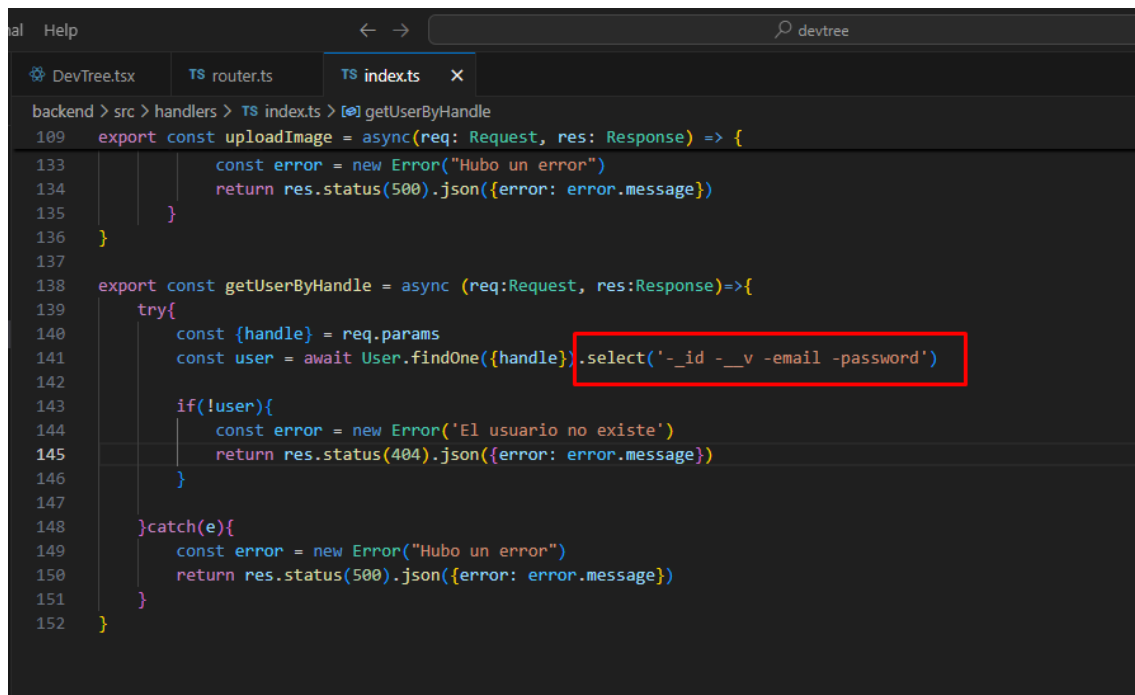


Si buscamos un usuario que no existe:





Traemos toda la información menos el ID, Versión, email y password.



```

Help
devtree
TS index.ts ...handlers X TS index.ts ...types HandleView.tsx router.tsx TS rout
backend > src > handlers > TS index.ts > [x] getUserByHandle
109 export const uploadImage = async(req: Request, res: Response) => {
130   try{
131     console.log('Desde uploadImage')
132   }catch(e){
133     const error = new Error("Hubo un error")
134     return res.status(500).json({error: error.message})
135   }
136 }
137
138 export const getUserByHandle = async (req:Request, res:Response)=>{
139   try{
140     const {handle} = req.params
141     const user = await User.findOne({handle}).select('-_id -__v -email -password')
142
143     if(!user){
144       const error = new Error('El usuario no existe')
145       return res.status(404).json({error: error.message})
146     }
147     return res.json(user)
148   }catch(e){
149     const error = new Error("Hubo un error")
150     return res.status(500).json({error: error.message})
151   }
152 }
153 }

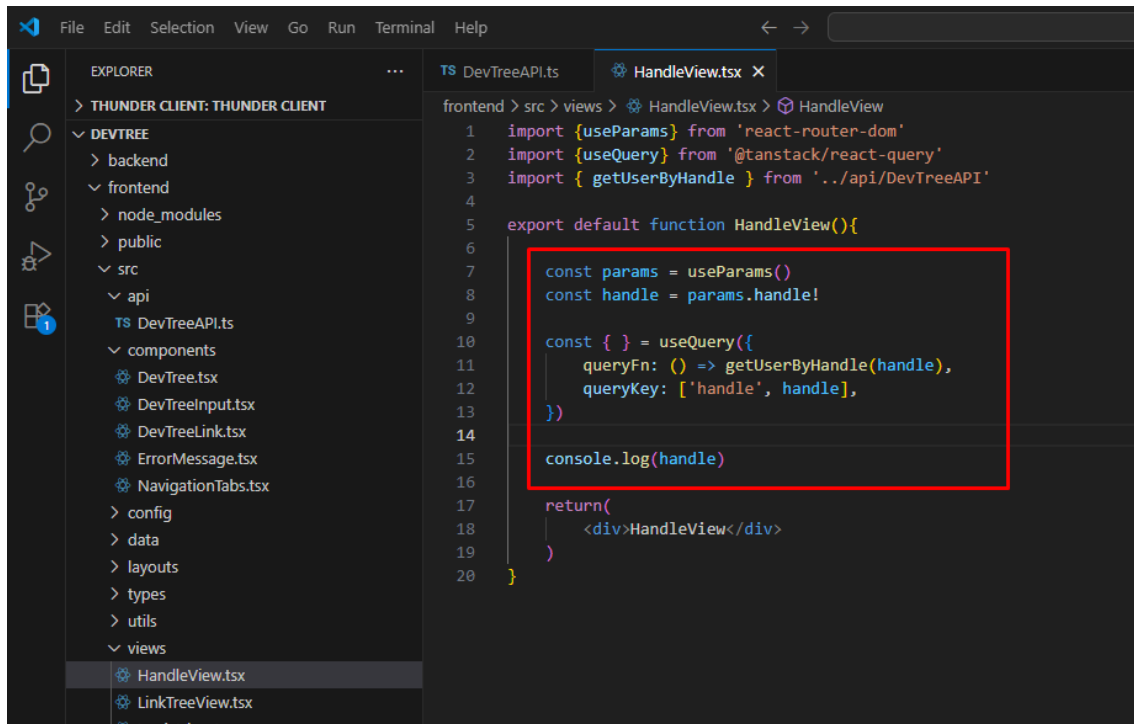
```

Consultando el Tree con React

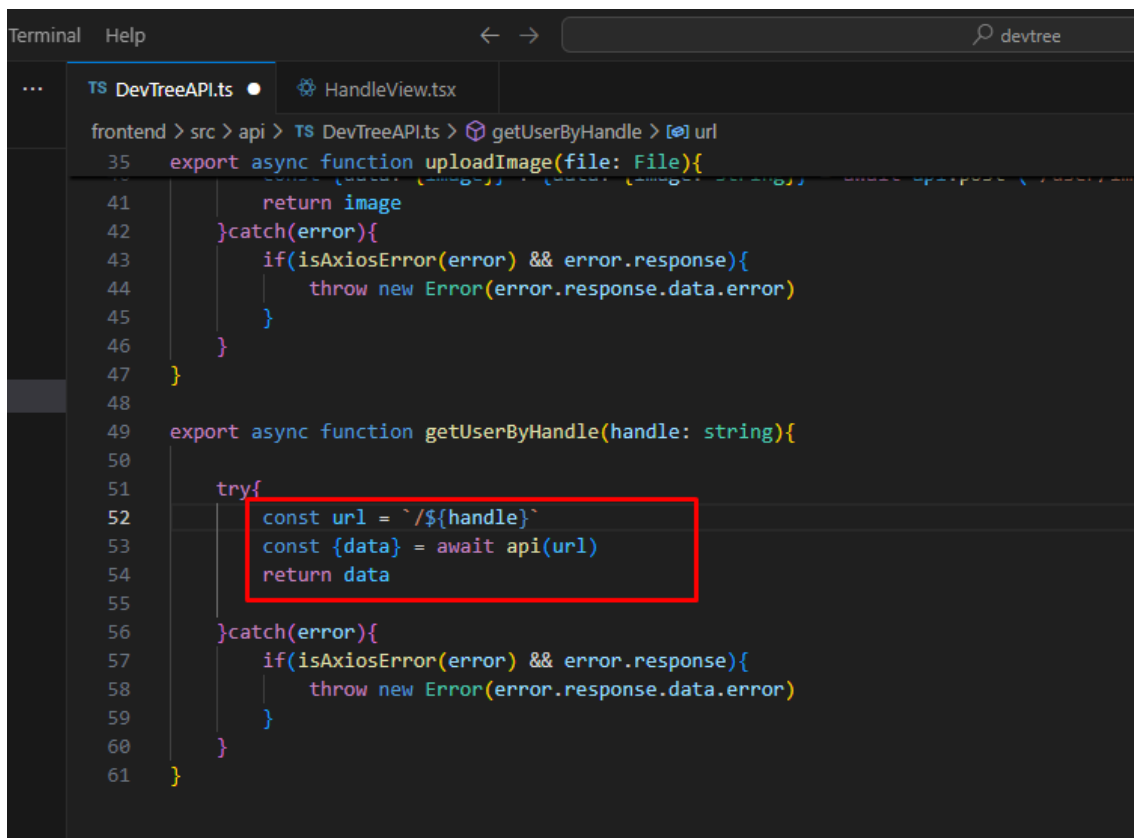
```

File Edit Selection View Go Run Terminal Help
devtree
EXPLORER
THUNDER CLIENT: THUNDER CLIENT
DEV TREE
  > backend
  > frontend
  > node_modules
  > public
  > src
  > api
    TS DevTreeAPI.ts
  > components
    DevTree.tsx
    DevTreeInput.tsx
    DevTreeLink.tsx
    ErrorMessage.tsx
    NavigationTabs.tsx
  > config
  > data
  > layouts
  > types
  > utils
TS DevTreeAPI.ts
HandleView.tsx
frontend > src > api > TS DevTreeAPI.ts > [x] getUserByHandle
35 export async function uploadImage(file: File){
46   }
47 }
48
49 export async function getUserByHandle(handle: string){
50   try{
51     console.log(handle)
52     return
53
54     const {data} = await api.patch<string>(`/user`)
55     return data
56   }catch(error){
57     if(isAxiosError(error) && error.response){
58       throw new Error(error.response.data.error)
59     }
60   }
61 }
62
63
64

```

```
1 import {useParams} from 'react-router-dom'
2 import {useQuery} from '@tanstack/react-query'
3 import {getUserByHandle} from '../api/DevTreeAPI'
4
5 export default function HandleView(){
6
7     const params = useParams()
8     const handle = params.handle!
9
10    const { } = useQuery({
11        queryFn: () => getUserByHandle(handle),
12        queryKey: ['handle', handle],
13    })
14
15    console.log(handle)
16
17    return(
18        <div>HandleView</div>
19    )
20 }
```



```
35 export async function uploadImage(file: File){
36     // ...
37     return image
38 }catch(error){
39     if(isAxiosError(error) && error.response){
40         throw new Error(error.response.data.error)
41     }
42 }
43
44 export async function getUserByHandle(handle: string){
45     try{
46         const url = `/${handle}`
47         const {data} = await api(url)
48         return data
49     }catch(error){
50         if(isAxiosError(error) && error.response){
51             throw new Error(error.response.data.error)
52         }
53     }
54 }
```

```

Help
DevTreeAPI.ts HandleView.tsx
frontend > src > views > HandleView.tsx > HandleView
1 import {useParams} from 'react-router-dom'
2 import {useQuery} from '@tanstack/react-query'
3 import { getUserByHandle } from '../api/DevTreeAPI'
4
5 export default function HandleView(){
6
7     const params = useParams()
8     const handle = params.handle!
9
10    const { data, error, isLoading } = useQuery({
11        queryFn: () => getUserByHandle(handle),
12        queryKey: ['handle', handle],
13        retry: 1
14    })
15
16    console.log(isLoading)
17    console.log(error)
18    console.log(data)
19
20    return(
21        <div>HandleView</div>
22    )
23 }

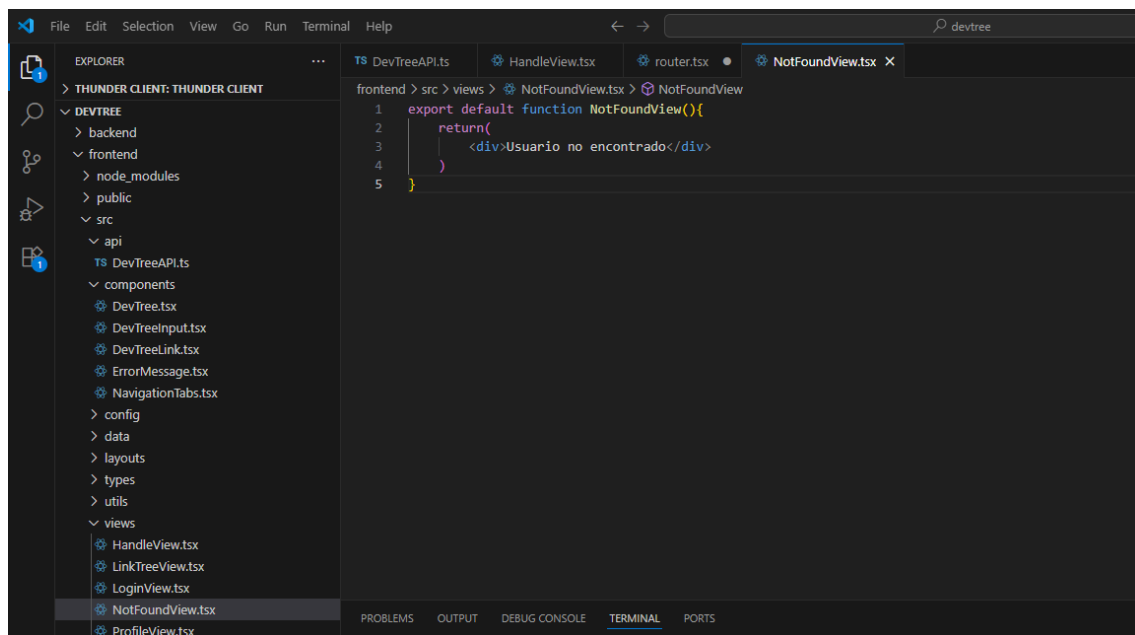
```



Creando una página 404

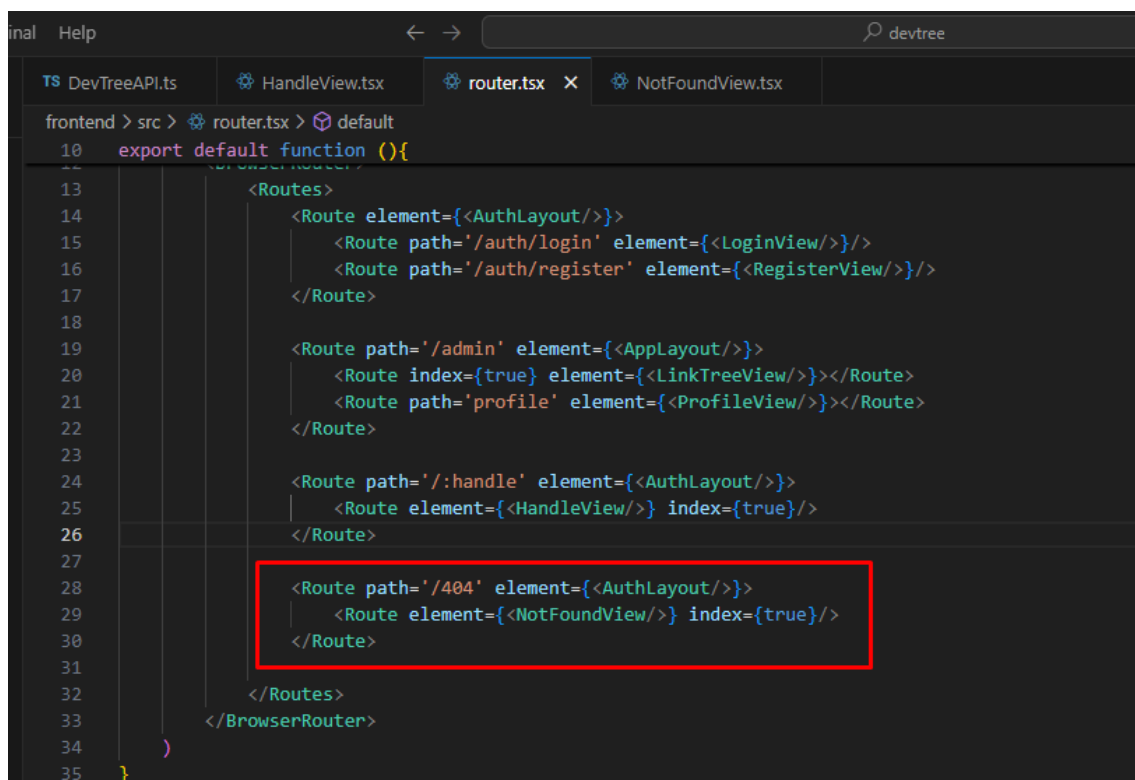
Ya estamos obteniendo los datos del usuario que estamos visitando en su URL, pero digamos que busco un usuario que no existe.

Creamos una nueva vista



This screenshot shows the Visual Studio Code interface. On the left, the Explorer sidebar displays the project structure under 'THUNDER CLIENT: THUNDER CLIENT'. The 'views' directory is expanded, showing files like 'HandleView.tsx', 'LinkTreeView.tsx', 'LoginView.tsx', 'NotFoundView.tsx', and 'ProfileView.tsx'. The 'NotFoundView.tsx' file is selected. The main editor area shows the content of 'NotFoundView.tsx' with the following code:

```
1 export default function NotFoundView(){
2   return(
3     <div>Usuario no encontrado</div>
4   )
5 }
```



This screenshot shows the Visual Studio Code interface with the 'router.tsx' file selected in the Explorer. The main editor area displays the router configuration code. A red box highlights the addition of a new route for the 404 error page:

```
10 export default function (){
11   // ...
12   <Routes>
13     <Route element={AuthLayout}>
14       <Route path='/auth/login' element={LoginView}/>
15       <Route path='/auth/register' element={RegisterView}/>
16     </Route>
17     <Route path='/admin' element={AppLayout}>
18       <Route index={true} element={LinkTreeView}></Route>
19       <Route path='profile' element={ProfileView}></Route>
20     </Route>
21     <Route path='/:handle' element={AuthLayout}>
22       <Route element={HandleView} index={true}/>
23     </Route>
24     <Route path='/404' element={AuthLayout}>
25       <Route element={NotFoundView} index={true}/>
26     </Route>
27   </Routes>
28 </BrowserRouter>
29 )
30 }
```

```

Help
TS DevTreeAPI.ts HandleView.tsx X router.tsx NotFoundView.tsx
frontend > src > views > HandleView.tsx > HandleView
1 import {Navigate, useParams} from 'react-router-dom'
2 import {useQuery} from '@tanstack/react-query'
3 import {getUserByHandle} from '../api/DevTreeAPI'
4
5 export default function HandleView(){
6
7     const params = useParams()
8     const handle = params.handle!
9
10    const { data, error, isLoading } = useQuery({
11        queryFn: () => getUserByHandle(handle),
12        queryKey: ['handle', handle],
13        retry: 1
14    })
15
16    if(isLoading) return 'Cargando...'
17    if(error) return <Navigate to={'/404'}/>
18
19    console.log(isLoading)
20    console.log(error)
21    console.log(data)
22
23    return(
24        <div>HandleView</div>
25    )

```

```

Help
TS DevTreeAPI.ts HandleView.tsx router.tsx NotFoundView.tsx X
frontend > src > views > NotFoundView.tsx > NotFoundView
1 export default function NotFoundView(){
2     return(
3         <p className="font-bold text-2xl text-center text-white">Usuario no encontrado</p>
4     )
5 }

```



Edita los estilos:

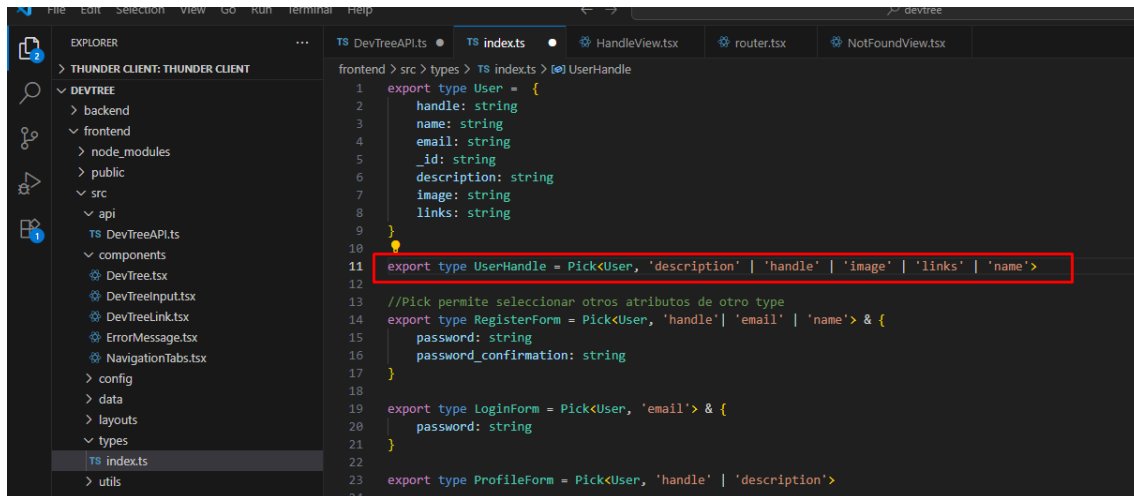
```

if(isLoading) return <p className='text-center text-white'>Cargando...</p>
if(error) return <Navigate to={'/404'}/>

```

Creando los types para la página de Handle

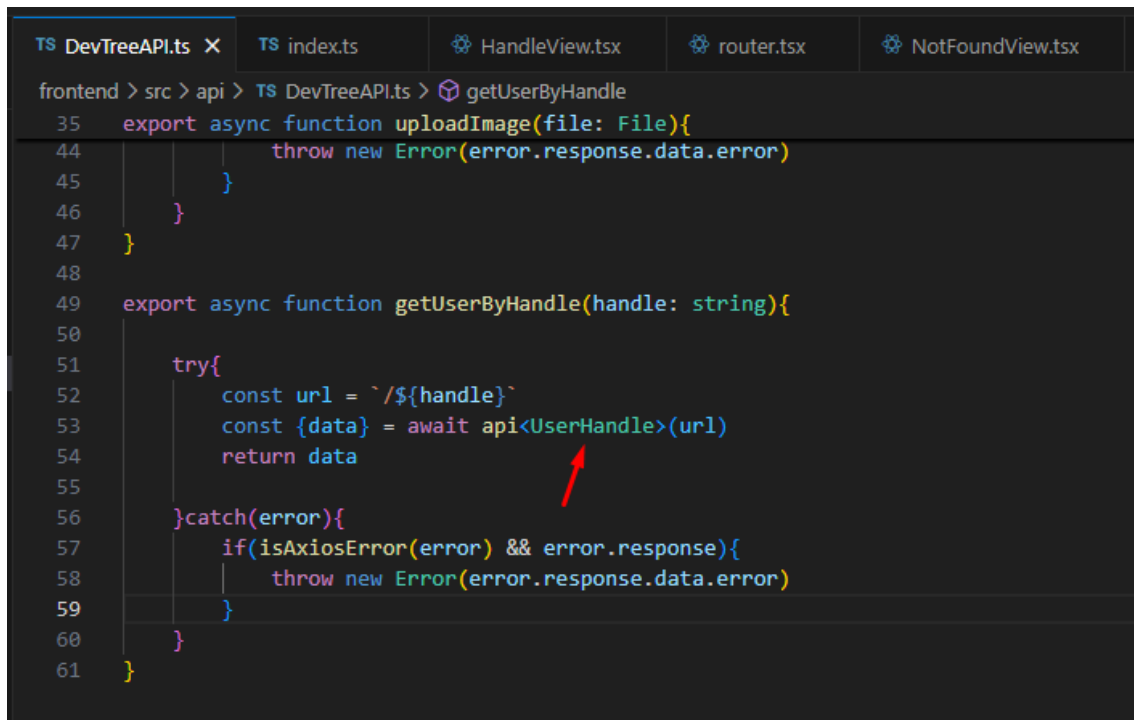
Los usuarios crearon sus cuentas y quieren compartirlo.



```

1  export type User = {
2    handle: string
3    name: string
4    email: string
5    _id: string
6    description: string
7    image: string
8    links: string
9  }
10
11  export type UserHandle = Pick<User, 'description' | 'handle' | 'image' | 'links' | 'name'>
12
13  //Pick permite seleccionar otros atributos de otro type
14  export type RegisterForm = Pick<User, 'handle' | 'email' | 'name'> & {
15    password: string
16    password_confirmation: string
17  }
18
19  export type LoginForm = Pick<User, 'email'> & {
20    password: string
21  }
22
23  export type ProfileForm = Pick<User, 'handle' | 'description'>
24

```



```

35  export async function uploadImage(file: File){
44    throw new Error(error.response.data.error)
45  }
46  }
47  }
48
49  export async function getUserByHandle(handle: string){
50
51    try{
52      const url = `/${handle}`
53      const {data} = await api<UserHandle>(url)
54      return data
55    }catch(error){
56      if(isAxiosError(error) && error.response){
57        throw new Error(error.response.data.error)
58      }
59    }
60  }
61  }

```

Creamos otro componente

EXPLORER

THUNDER CLIENT: THUNDER CLIENT

- DEV TREE
 - backend
 - frontend
 - node_modules
 - public
 - src
 - api
 - DevTreeAPI.ts
 - components
 - DevTree.tsx
 - DevTreeInput.tsx
 - DevTreeLink.tsx
 - ErrorMessage.tsx
 - HandleData.tsx
 - NavigationTabs.tsx
 - config
 - data

frontend > src > components > HandleData.tsx > HandleData

```

1
2 export default function HandleData(){
3   return (
4     <div>HandleData</div>
5   )
6 }

```

frontend > src > views > HandleView.tsx > HandleView

```

1 import {Navigate, useParams} from 'react-router-dom'
2 import {useQuery} from '@tanstack/react-query'
3 import {getUserByHandle} from '../api/DevTreeAPI'
4 import HandleData from '../components/HandleData'
5
6 export default function HandleView(){
7
8   const params = useParams()
9   const handle = params.handle!
10
11   const { data, error, isLoading } = useQuery({
12     queryFn: () => getUserByHandle(handle),
13     queryKey: ['handle', handle],
14     retry: 1
15   })
16
17   if(isLoading) return <p className='text-center text-white'>Cargando...</p>
18   if(error) return <Navigate to={'/404'}/>
19   if(data) return <HandleData data={data}/>
20 }

```

frontend > src > components > HandleData.tsx > HandleData

```

1 import type { UserHandle } from "../types"
2
3 type HandleDataProps = {
4   data: UserHandle
5 }
6
7 export default function HandleData({data}:HandleDataProps){
8   return (
9     <div>HandleData</div>
10  )
11 }

```

```

1 import type { UserHandle } from "../types"
2
3 type HandleDataProps = {
4   data: UserHandle
5 }
6
7 export default function HandleData({data}:HandleDataProps){
8   return (
9     <div className="space-y-6 text-white">
10       <p className="text-5xl text-center font-black">{data.handle}</p>
11       {data.image && <img src={data.image} className="max-w-[250px] mx-auto"/>}
12       <p className="text-lg text-center font-bold">{data.description}</p>
13     </div>
14   )
15 }

```



Mostrando los enlaces activos del Usuario

Vamos a mostrar las redes sociales que los usuarios han compartido. Solo vamos a mostrar los que están habilitadas.

```
...  ← →  devtree  [Icons]  [Icons]  [Icons]  [Icons]  [Icons]
TS DevTreeAPI.ts  TS index.ts  HandleView.tsx  router.tsx  TS router.ts  HandleData.tsx X
frontend > src > components > HandleData.tsx > HandleData
7  export default function HandleData({data}:HandleDataProps){
8
9      const links: SocialNetwork[] = JSON.parse(data.links).filter((link: SocialNetwork)=>link.enabled)
10
11      return (
12          <div className="space-y-6 text-white">
13              <p className="text-5xl text-center font-black">{data.handle}</p>
14
15              {data.image && <img src={data.image} className="max-w-[250px] mx-auto"/>}
16
17              <p className="text-lg text-center font-bold">{data.description}</p>
18              <div className="mt-20 flex flex-col gap-6">
19                  {links.length ?
20                      <p className="text-center">Sí hay enlaces</p>
21                      : <p className="text-center">No hay enlaces en este perfil</p>}
22              </div>
23          </div>
24      )
25  }
26  }
```




```

...  ← →  devtree
TS DevTreeAPI.ts  TS index.ts  HandleView.tsx  router.tsx  TS router.ts  HandleData.tsx X
frontend > src > components > HandleData.tsx > HandleData
7  fault function HandleData({data}:HandleDataProps){
11  n (
12  div className="space-y-6 text-white">
13    <p className="text-5xl text-center font-black">{data.handle}</p>
14
15    {data.image && <img src={data.image} className="max-w-[250px] mx-auto"/>}
16
17    <p className="text-lg text-center font-bold">{data.description}</p>
18    <div className="mt-20 flex flex-col gap-6">
19      {links.length ?
20        links.map(link=>{
21          <a
22            key={link.name}
23            className="bg-white px-5 py-2 flex items-center gap-5 rounded-lg w-full max-w-md mx-auto"
24            href={link.url}
25            target="_blank"
26            rel="noopener noreferrer"
27          >
28            <img src={`/social/icon_${link.name}.svg`} alt="icono red social" className="w-12"/>
29            <p className="text-black capitalize font-bold text-lg">Visita mi: {link.name}</p>
30          </a>
31        })
32        : <p className="text-center">No hay enlaces en este perfil</p>}
33    </div>
34  </div>
35  /div>
36

```

Vista final:

