MIT em Desenvolvimento Full Stack

Front-end Jamstack com Gatsby

Agenda

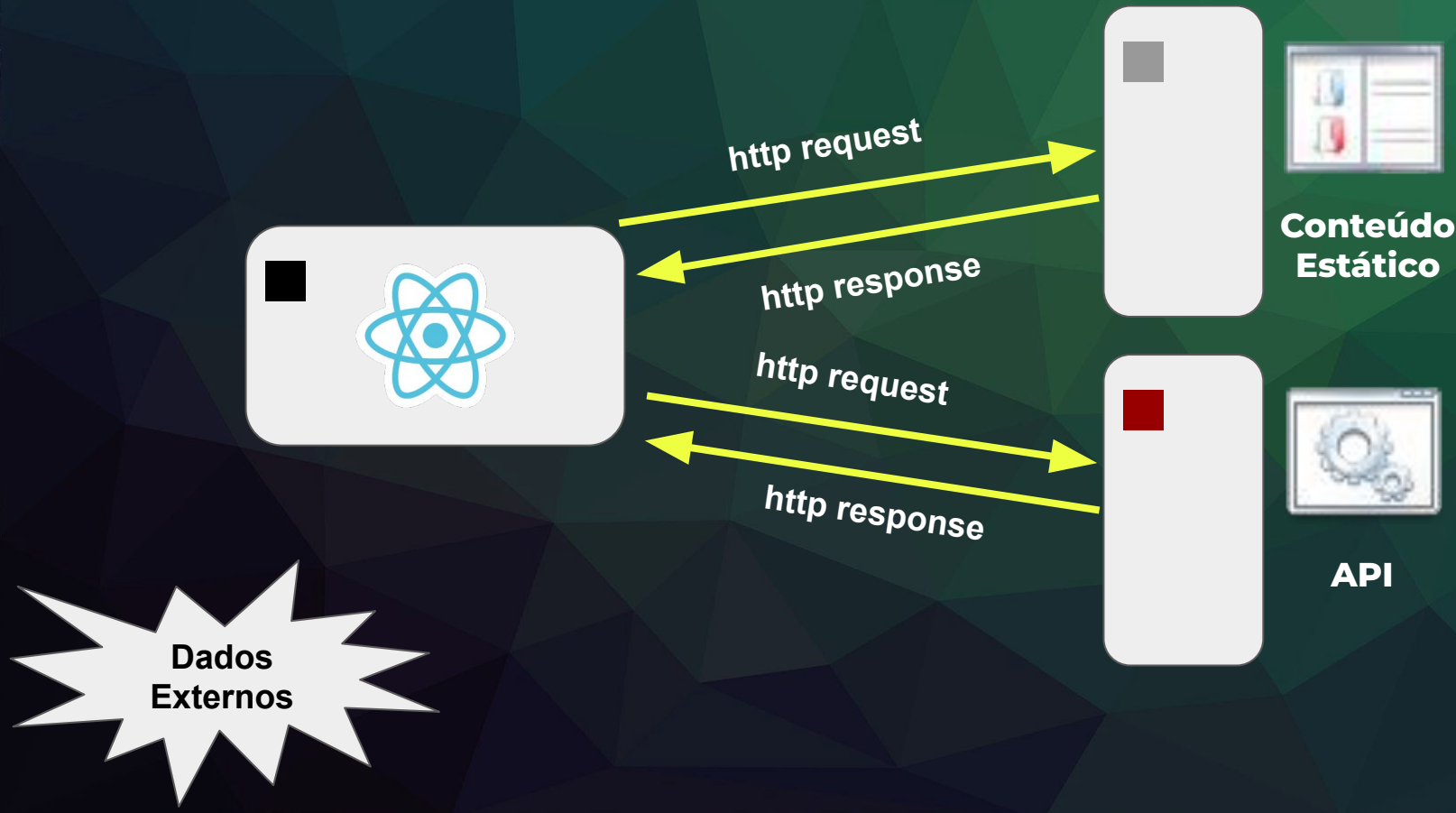
Aula 4: Desenvolvimento da Estrutura da Aplicação com Gatsby:

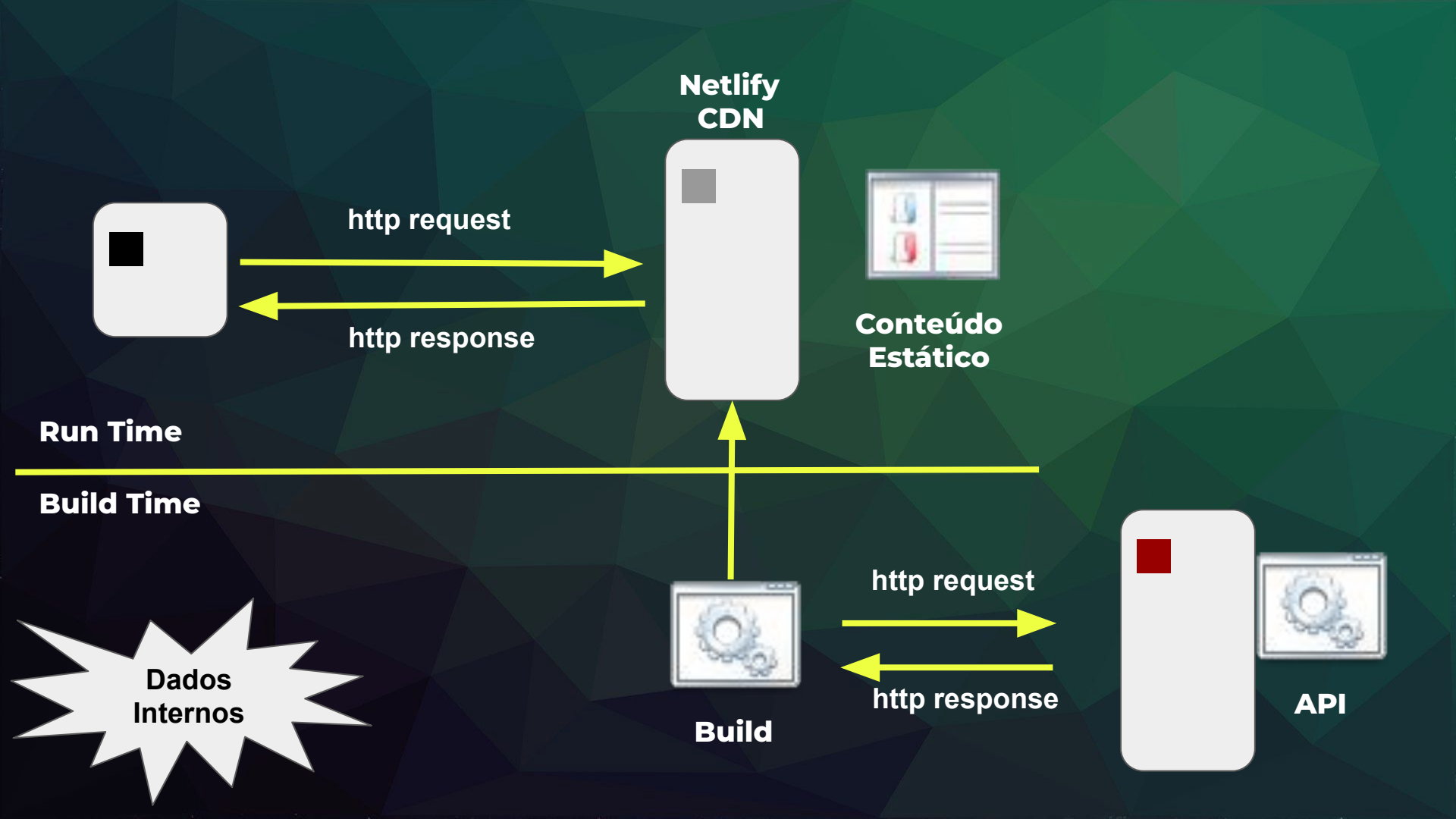
- Recapitulando.
- Dados Internos.

Desenvolvimento da Aplicação com Formulários Estáticos em Gatsby:

- Formulários Estáticos.
- Formulário em React.

Recapitulando

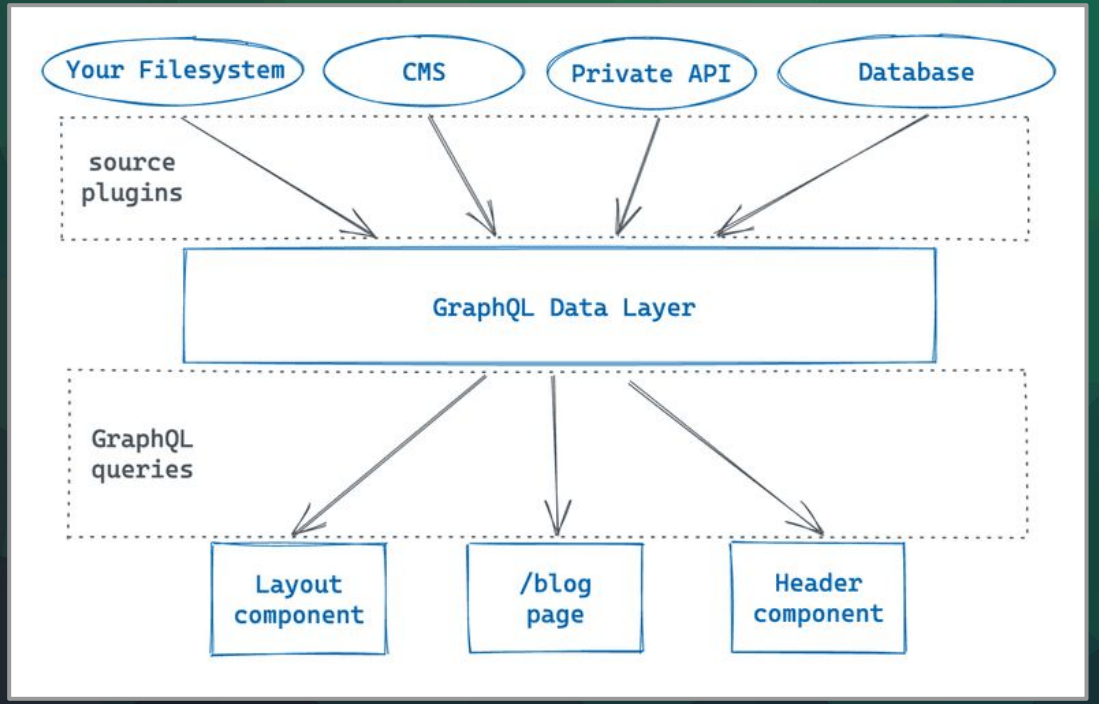

http request
http response
http request
http response
Conteúdo Estático
API
Dados Externos


Netlify
CDN
http request
http response
Conteúdo
Estático
Run Time
Build Time
Dados
Internos
Build
http request
http response
API

Dados Internos

GraphQL Data Layer é uma camada intermediária que reúne e normaliza dados vindos de múltiplas fontes (arquivos locais, APIs, CMS, bancos, etc.) em uma única API acessível no momento do build.

Essa camada coleta dados de diferentes origens, converte tudo em nós (nodes) dentro de um grafo interno, disponibiliza os dados via **GraphQL** de maneira uniforme.



GraphQL é uma linguagem de consulta de dados (query language) e um runtime desenvolvido pelo Facebook (2015).

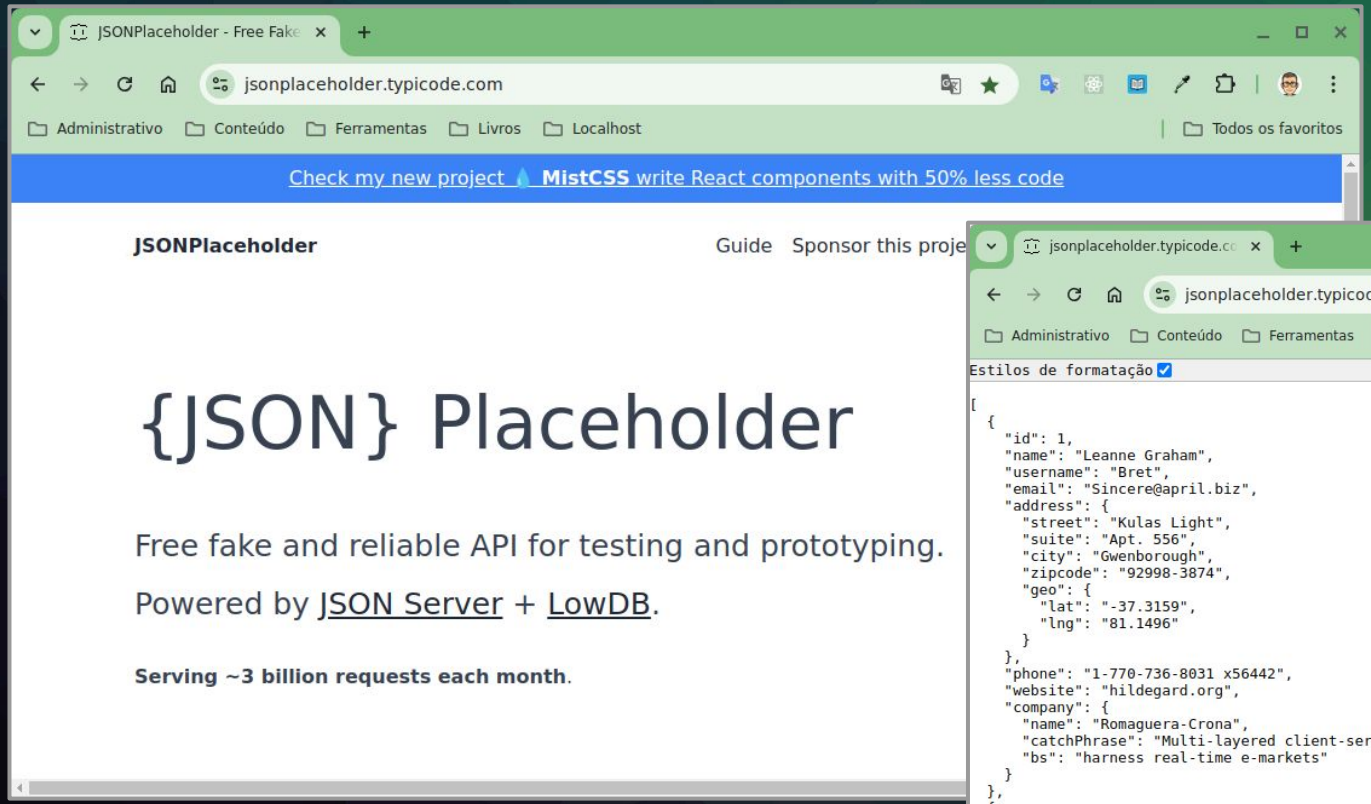
Ele permite que o cliente (por exemplo, o Gatsby ou um app React) peça exatamente os dados que precisa, nem mais, nem menos.

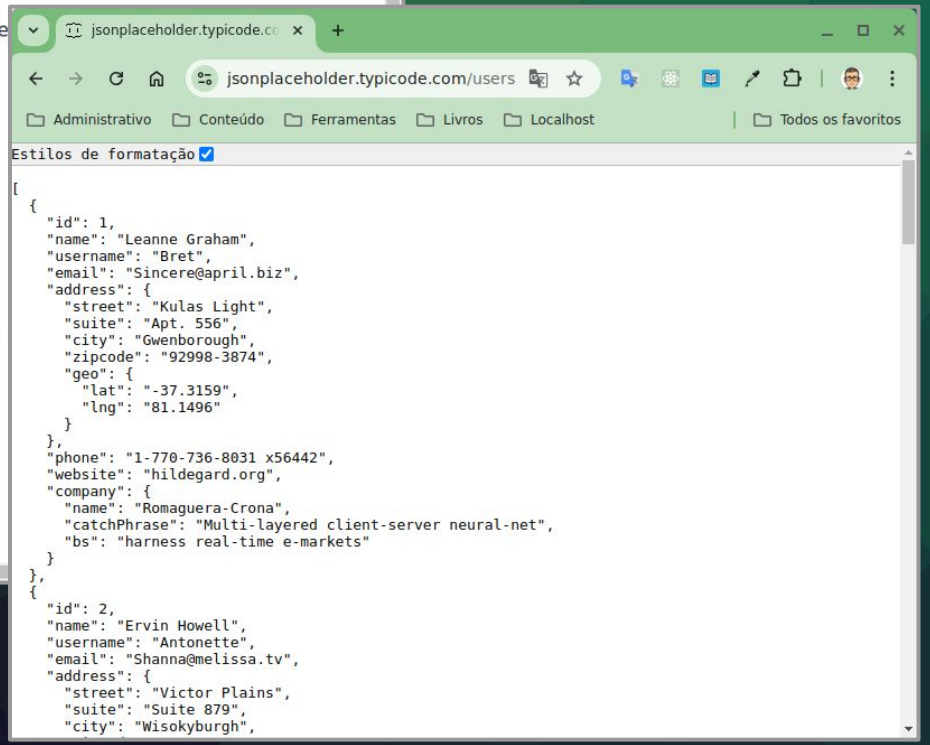
É uma alternativa moderna ao REST: em vez de múltiplos endpoints (ex.: /users, /posts, /comments), você faz uma só requisição com a estrutura de dados desejada.

```
GET /users/1
GET /users/1/posts
```

```
{
  user(id: 1) {
    name
    email
    posts {
      title
    }
  }
}
```

Data layer unificado	Combina várias fontes (APIs, CMS, Markdown, Firestore, etc.) em uma só camada.
Consultas declarativas	O componente pede *apenas* o que precisa — menos dados, mais performance.
Autocompletar e tipagem forte	O Gatsby gera um schema GraphQL tipado, ajudando com autocomplete no editor e segurança em runtime.
Build estático otimizado	Dados são resolvidos no build; o usuário final recebe HTML puro e rápido.
Menos dependência de APIs no frontend	Evita chamadas dinâmicas (fetch), reduzindo tempo de carregamento e risco de falhas.


JSONPlaceholder - Free Fake
jsonplaceholder.typicode.com
Administrativo
Conteúdo
Ferramentas
Livros
Localhost
Todos os favoritos
Check my new project MistCSS write React components with 50% less code
JSONPlaceholder
Guide
Sponsor this proje
{JSON} Placeholder
Free fake and reliable API for testing and prototyping.
Powered by JSON Server + LowDB.
Serving ~3 billion requests each month.


jsonplaceholder.typicode.cc
jsonplaceholder.typicode.com/users
Administrativo
Conteúdo
Ferramentas
Livros
Localhost
Todos os favoritos
Estilos de formatação
[
{
"id": 1,
"name": "Leanne Graham",
"username": "Bret",
"email": "Sincere@april.biz",
"address": {
"street": "Kulas Light",
"suite": "Apt. 556",
"city": "Gwenborough",
"zipcode": "92998-3874",
"geo": {
"lat": "-37.3159",
"lng": "81.1496"
}
},
"phone": "1-770-736-8031 x56442",
"website": "hildegard.org",
"company": {
"name": "Romaguera-Crona",
"catchPhrase": "Multi-layered client-server neural-net",
"bs": "harness real-time e-markets"
}
},
{
"id": 2,
"name": "Ervin Howell",
"username": "Antonette",
"email": "Shanna@melissa.tv",
"address": {
"street": "Victor Plains",
"suite": "Suite 879",
"city": "Wisokyburgh",

1 npm install node-fetch

Continuando no projeto anterior

gatsby-node.js U

gatsby-node.js > ...

```
1  import fetch from "node-fetch";
2
3  export const sourceNodes = async ({ actions, createNodeId, createContentDigest }) => {
4    const { createNode } = actions;
5
6    //Buscar os dados da API
7    const response = await fetch("https://jsonplaceholder.typicode.com/users");
8    const users = await response.json();
9
10   //Criar nodes Gatsby para cada usuário
11   users.forEach(user => {
12     createNode({
13       ...user,
14       id: createNodeId(`User-${user.id}`),
15       parent: null,
16       children: [],
17       internal: {
18         type: "User",
19         contentDigest: createContentDigest(user),
20       },
21     });
22   });
23 };
```

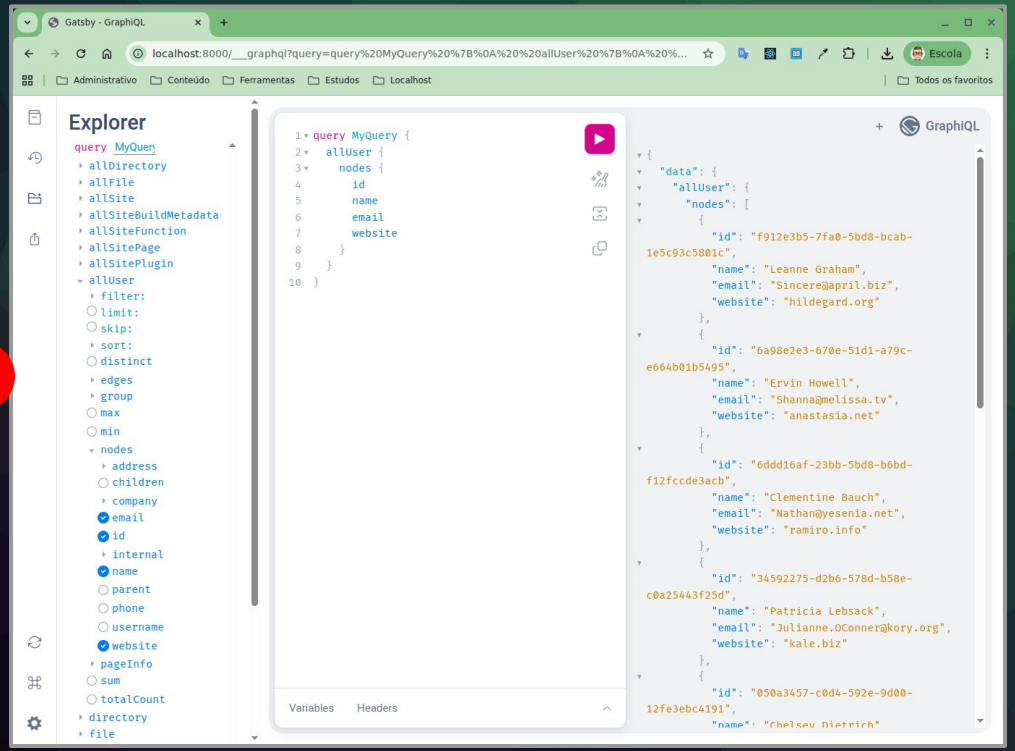
2

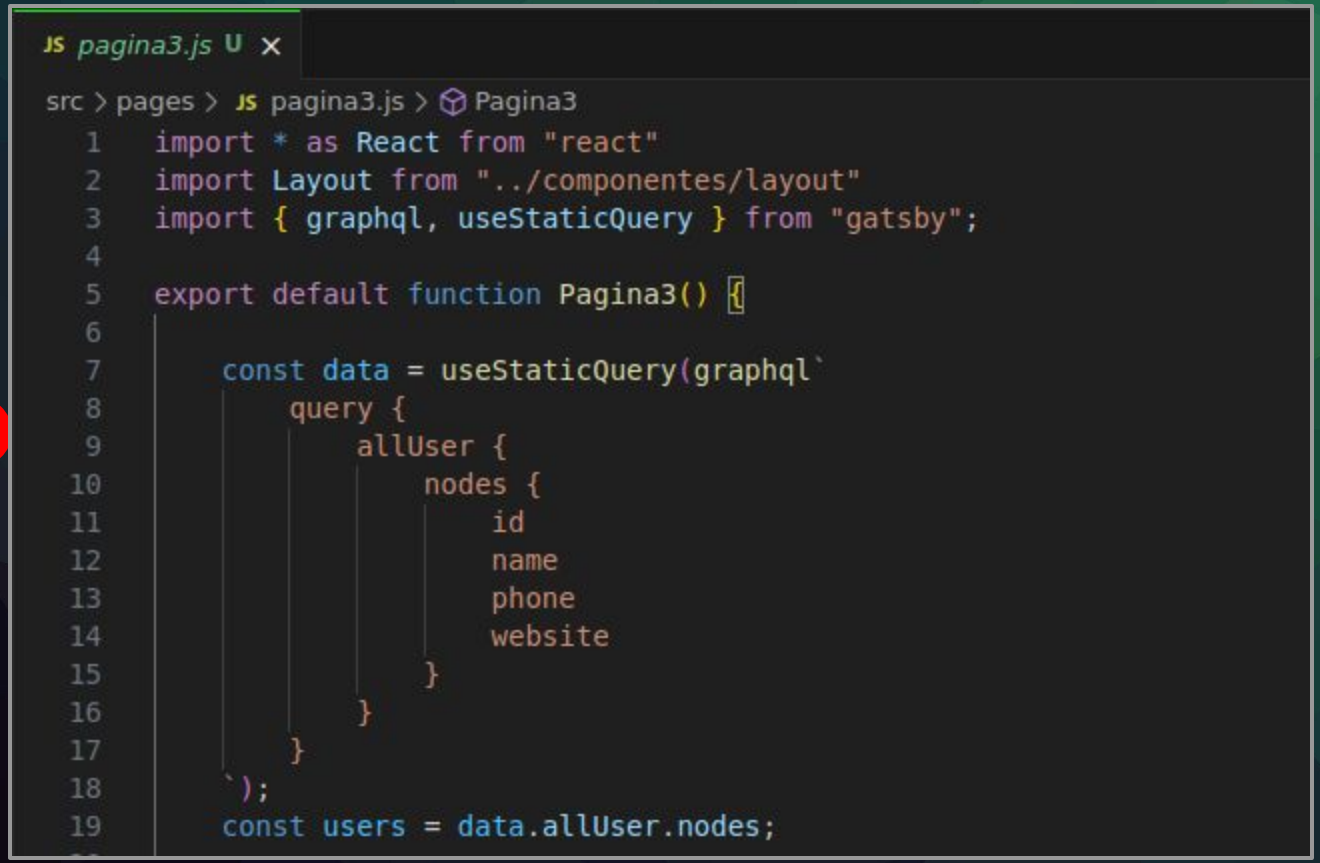
PROBLEMAS SAÍDA CONSOLE DE DEPURAÇÃO TERMINAL PORTAS

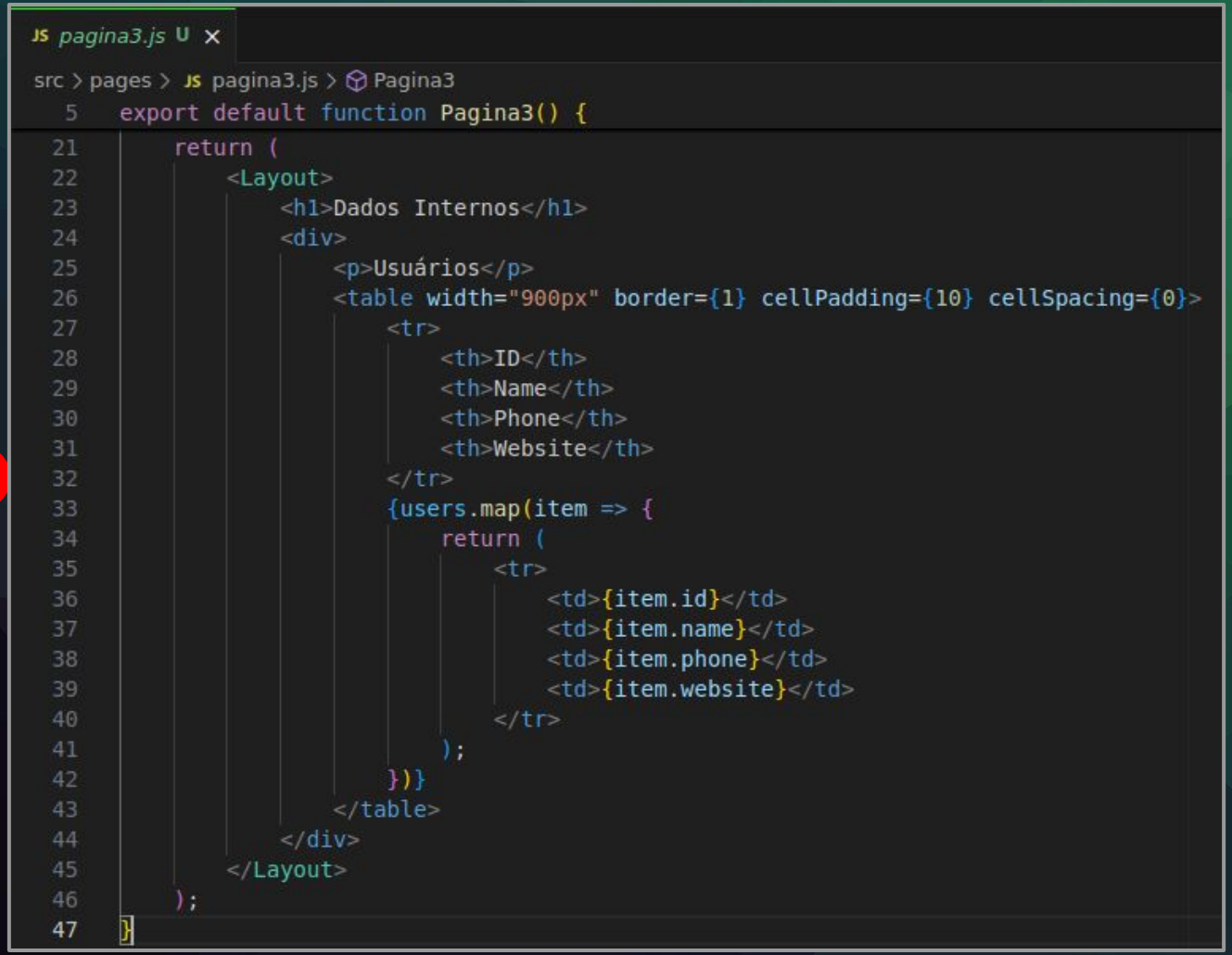
3

```
  http://localhost:8000/___graphql

Note that the development build is not optimized.
To create a production build, use gatsby build

success Building development bundle - 4.389s
success Writing page-data.json and slice-data.json files to public directory - 0.103s - 1/1 9.69/s
```

4

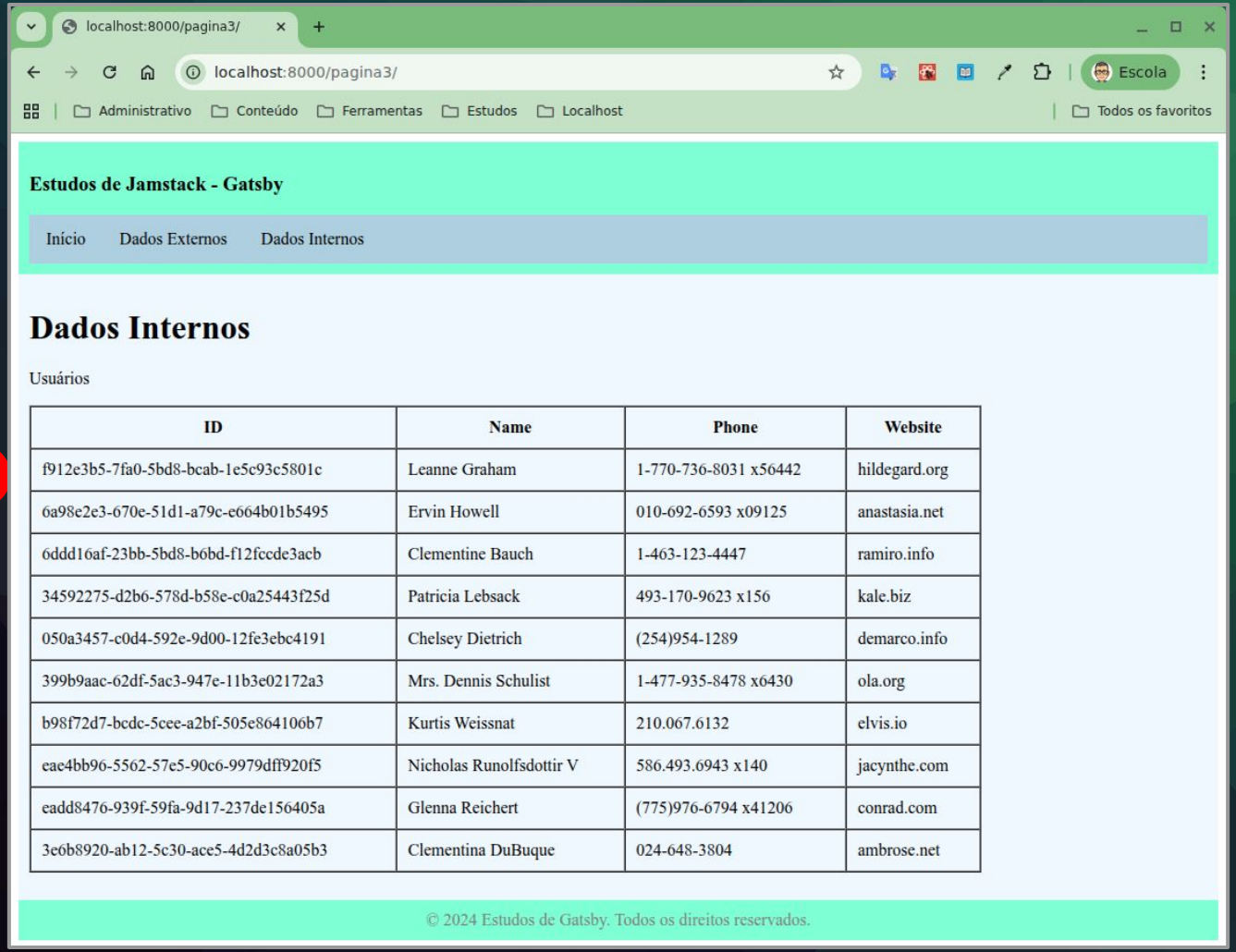



pagina3.js U
src > pages > pagina3.js > Pagina3
5
1 import * as React from "react"
2 import Layout from "../componentes/layout"
3 import { graphql, useStaticQuery } from "gatsby";
4
5 export default function Pagina3() {
6
7 const data = useStaticQuery(graphql`
8 query {
9 allUser {
10 nodes {
11 id
12 name
13 phone
14 website
15 }
16 }
17 }
18 `);
19 const users = data.allUser.nodes;


pagina3.js U
src > pages > pagina3.js > Pagina3
export default function Pagina3() {
return (
<Layout>
<h1>Dados Internos</h1>

Usuários
<table width="900px" border={1} cellPadding={10} cellSpacing={0}>
<tr>
<th>ID</th>
<th>Name</th>
<th>Phone</th>
<th>Website</th>
</tr>
{users.map(item => {
return (
<tr>
<td>{item.id}</td>
<td>{item.name}</td>
<td>{item.phone}</td>
<td>{item.website}</td>
</tr>
);
})}
</table>

</Layout>
);
}
6


localhost:8000/pagina3/
localhost:8000/pagina3/
Escola
Administrativo
Conteúdo
Ferramentas
Estudos
Localhost
Todos os favoritos
Estudos de Jamstack - Gatsby
Início
Dados Externos
Dados Internos
Dados Internos
Usuários
ID
Name
Phone
Website
f912e3b5-7fa0-5bd8-bcab-1e5c93c5801c
Leanne Graham
1-770-736-8031 x56442
hildegard.org
6a98e2e3-670e-51d1-a79c-e664b01b5495
Ervin Howell
010-692-6593 x09125
anastasia.net
6ddd16af-23bb-5bd8-b6bd-f12fccde3acb
Clementine Bauch
1-463-123-4447
ramiro.info
34592275-d2b6-578d-b58e-c0a25443f25d
Patricia Lebsack
493-170-9623 x156
kale.biz
050a3457-c0d4-592e-9d00-12fe3ebc4191
Chelsey Dietrich
(254)954-1289
demarco.info
399b9aac-62df-5ac3-947e-11b3e02172a3
Mrs. Dennis Schulist
1-477-935-8478 x6430
ola.org
b98f72d7-bcdc-5cee-a2bf-505e864106b7
Kurtis Weissnat
210.067.6132
elvis.io
eae4bb96-5562-57e5-90c6-9979dff920f5
Nicholas Runolfsdottir V
586.493.6943 x140
jacynthe.com
eadd8476-939f-59fa-9d17-237de156405a
Glenna Reichert
(775)976-6794 x41206
conrad.com
3e6b8920-ab12-5c30-ace5-4d2d3c8a05b3
Clementina DuBuque
024-648-3804
ambrose.net
© 2024 Estudos de Gatsby. Todos os direitos reservados.

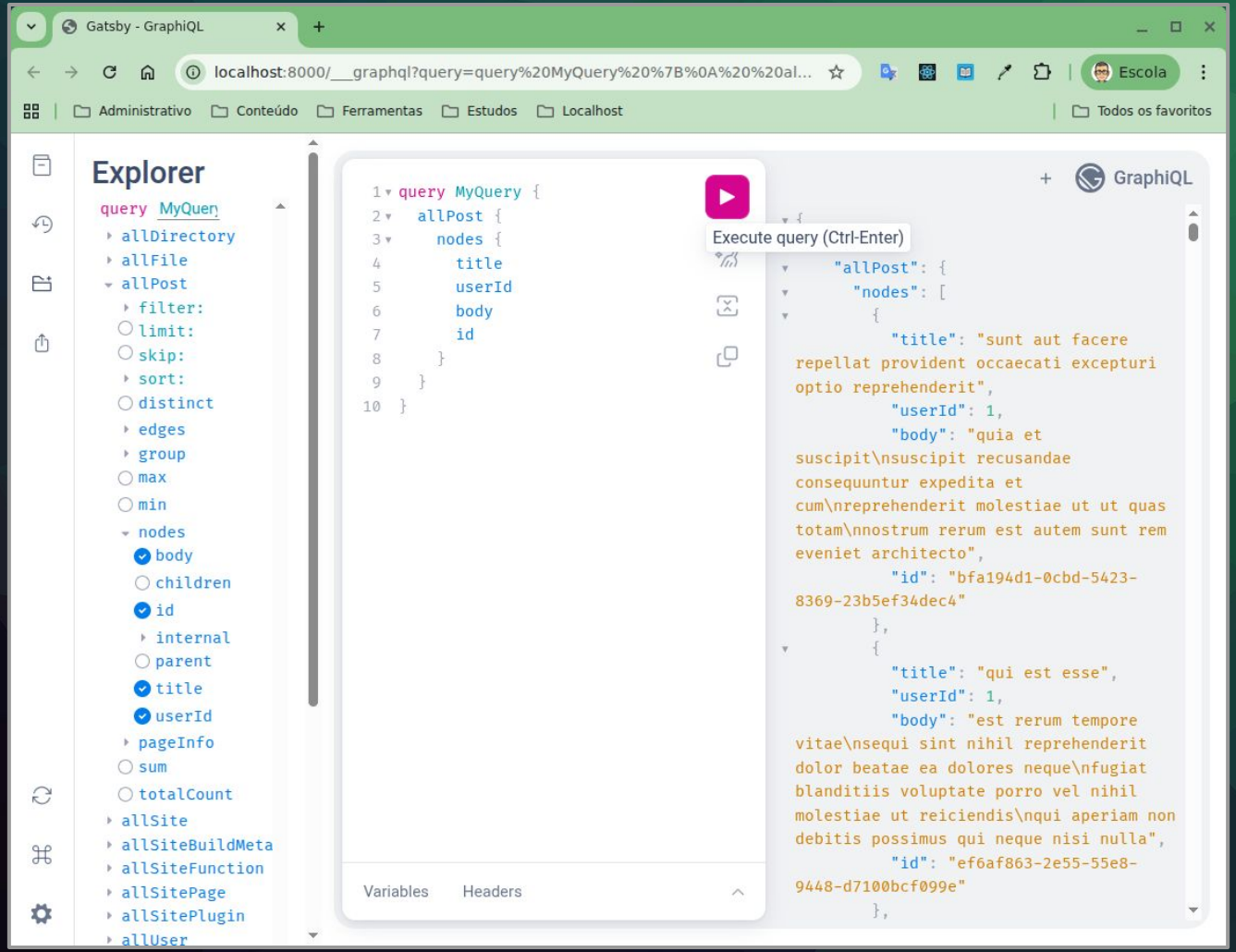
7

8

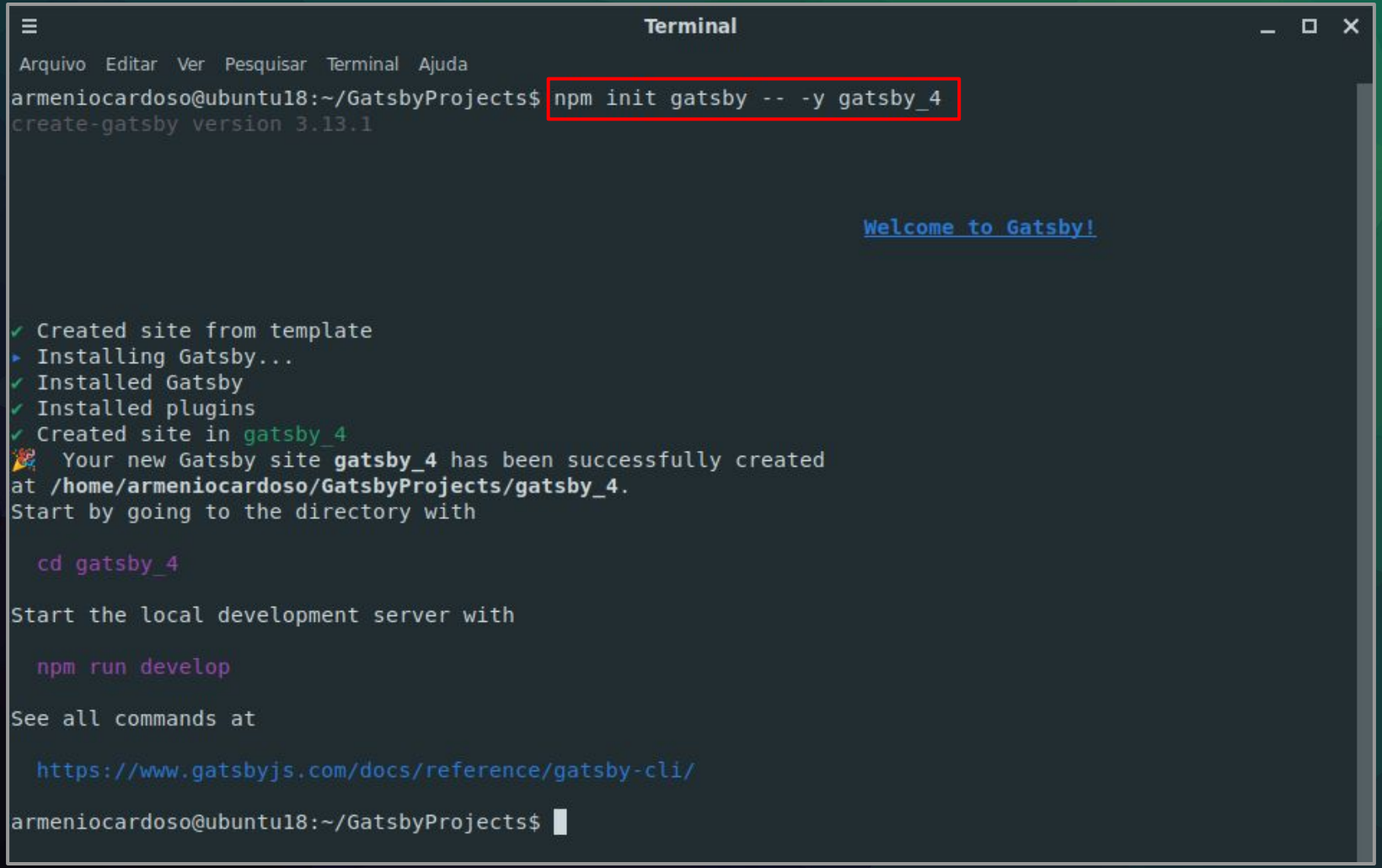
gatsby-node.js U

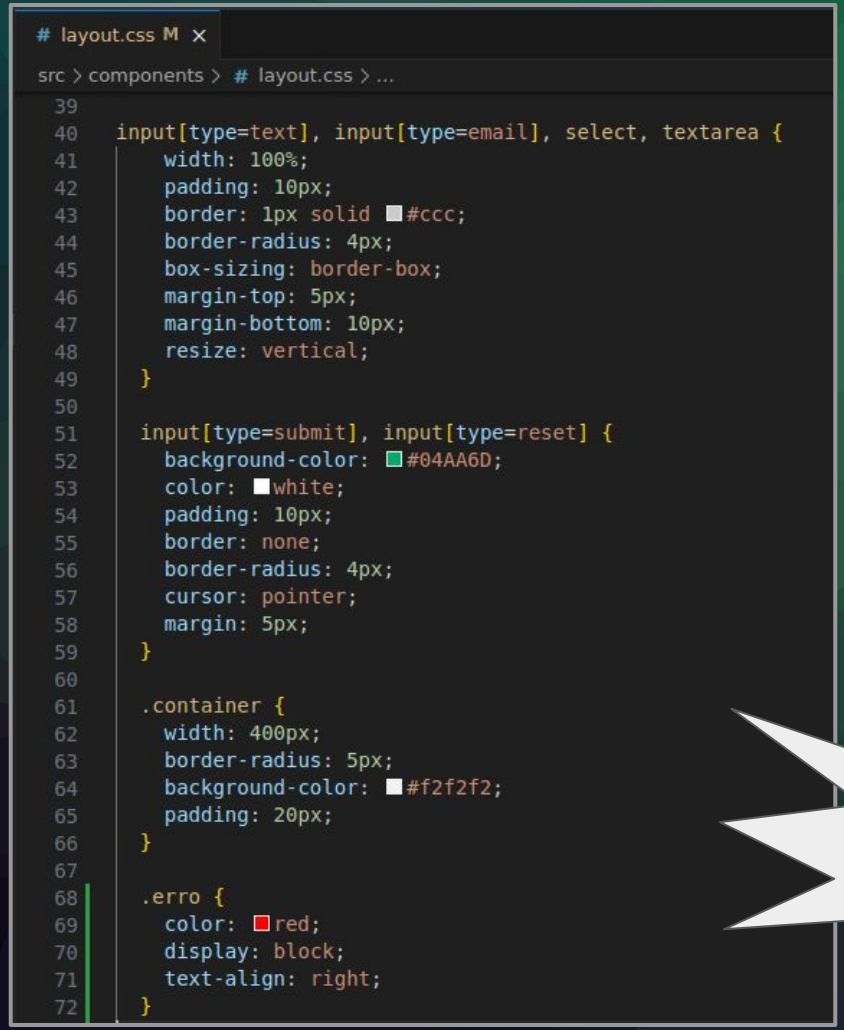
gatsby-node.js > ...

```
3   export const sourceNodes = async ({ actions, createNodeId, createContentDigest }) => {
23    //Buscar dados da API de posts
24    const postsResponse = await fetch("https://jsonplaceholder.typicode.com/posts");
25    const posts = await postsResponse.json();
26
27    posts.forEach(post => {
28      createNode({
29        ...post,
30        id: createNodeId(`Post-${post.id}`),
31        parent: null,
32        children: [],
33        internal: {
34          type: "Post",
35          contentDigest: createContentDigest(post),
36        },
37      });
38    });
39  };
```


Gatsby - GraphiQL
localhost:8000/___graphql?query=query%20MyQuery%20%7B%0A%20%20al...
Escola
Administrativo
Conteúdo
Ferramentas
Estudos
Localhost
Todos os favoritos
Explorer
query MyQuery
allDirectory
allFile
allPost
filter:
limit:
skip:
sort:
distinct
edges
group
max
min
nodes
body
children
id
internal
parent
title
userId
pageInfo
sum
totalCount
allSite
allSiteBuildMeta
allSiteFunction
allSitePage
allSitePlugin
allUser
query MyQuery {
allPost {
nodes {
title
userId
body
id
}
}
}
Execute query (Ctrl-Enter)
GraphiQL
"allPost": {
"nodes": [
"title": "sunt aut facere repellat provident occaecati excepturi optio reprehenderit",
"userId": 1,
"body": "quia et suscipit\nsuscipit recusandae consequuntur expedita et cum\nreprehenderit molestiae ut ut quas totam\nnostrum rerum est autem sunt rem eveniet architecto",
"id": "bfa194d1-0cbd-5423-8369-23b5ef34dec4"
"title": "qui est esse",
"userId": 1,
"body": "est rerum tempore vitae\nsequi sint nihil reprehenderit dolor beatae ea dolores neque\nfugiat blanditiis voluptate porro vel nihil molestiae ut reiciendis\nqui aperiam non debitis possimus qui neque nisi nulla",
"id": "ef6af863-2e55-55e8-9448-d7100bcf099e"
Variables
Headers

Formulário Estático


Terminal
Arquivo Editar Ver Pesquisar Terminal Ajuda
armeniocardoso@ubuntu18:~/GatsbyProjects$ npm init gatsby -- -y gatsby_4
create-gatsby version 3.13.1
Welcome to Gatsby!
✔ Created site from template
▸ Installing Gatsby...
✔ Installed Gatsby
✔ Installed plugins
✔ Created site in gatsby_4
🎉 Your new Gatsby site gatsby_4 has been successfully created
at /home/armeniocardoso/GatsbyProjects/gatsby_4.
Start by going to the directory with
cd gatsby_4
Start the local development server with
npm run develop
See all commands at
https://www.gatsbyjs.com/docs/reference/gatsby-cli/
armeniocardoso@ubuntu18:~/GatsbyProjects$



Estilos dos Elementos do Formulário

pagina1.js M

src > pages > pagina1.js > ...

```
6       <Layout>
7         <h2>Entre em Contato:</h2>
8         <div className="container">
9           <form name="form_estatico" method="post" data-netlify="true" data-netlify-honeypot="bot-field">
10            <input type="hidden" name="form-name" value="form_estatico" />
11            <label>
12              Nome
13              <input type="text" name="nome" />
14            </label>
15            <label>
16              Email
17              <input type="email" name="email" />
18            </label>
19            <label>
20              Assunto
21              <input type="text" name="assunto" />
22            </label>
23            <label>
24              Mensagem
25              <textarea name="mensagem" rows="5" />
26            </label>
27            <input type="submit" value="Enviar" />
28            <input type="reset" value="Limpar" />
29          </form>
30        </div>
31      </Layout>
32    )
33  }
34
35  export default Pagina1
36
37  export const Head = () => <title>Formulário Estático</title>
```

Usando o Netlify para receber os dados

Dataset

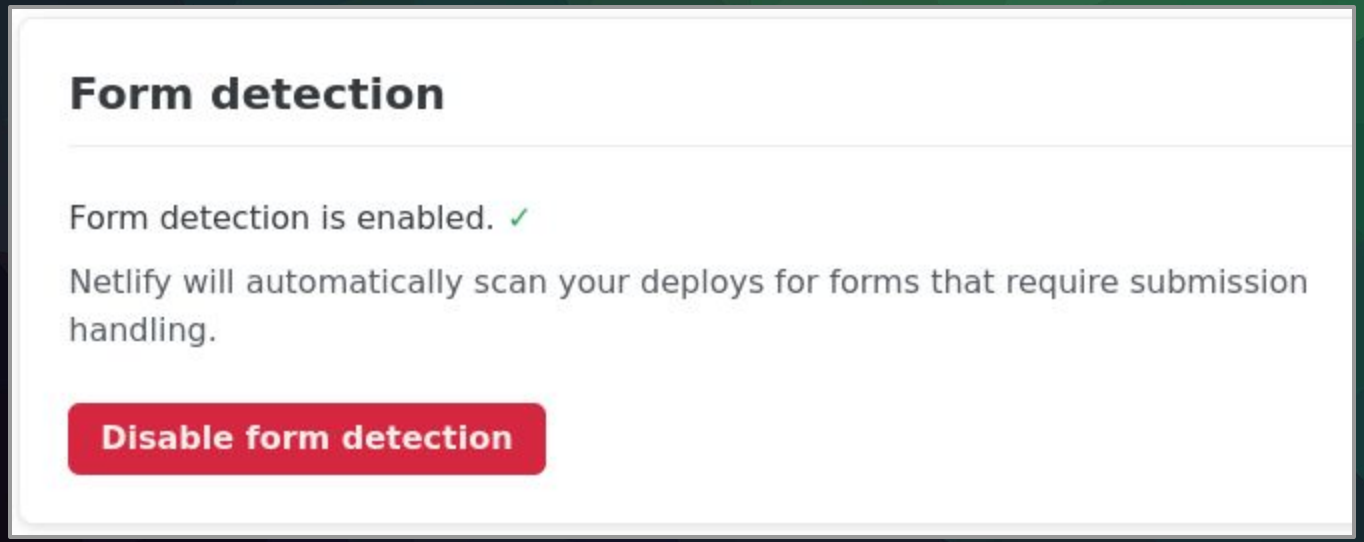
A propriedade somente leitura **dataset** fornece acesso de leitura/gravação a **atributos de dados personalizados** (data-*) em elementos HTML.

Ele expõe um mapa de strings com uma entrada para cada atributo **data-*** .

Em HTML o nome do atributo começa com **data-** . Ele pode conter apenas letras, números, hífens (-), pontos (.), dois pontos (:) e sublinhados (_). Quaisquer letras maiúsculas são convertidas em minúsculas.

Em JavaScript o nome da propriedade de um atributo de dados personalizados é o mesmo que o atributo HTML sem o prefixo **data** e remove os traços simples (-) para colocar em maiúscula o nome "camelCased" da propriedade.

```
----------------------------------------
git add .
git commit -m "Formulario Estatico"
git push -u origin main
----------------------------------------
```



Submissions from form_estatico

Extra spam prevention enabled via honeypot field.

Learn more about form handling in the docs ↗

Download as CSV | Delete form

Verified submissions

Select all on page | Delete submission | Mark as spam | Expand all

Machado de Assis Muito bom esse teste

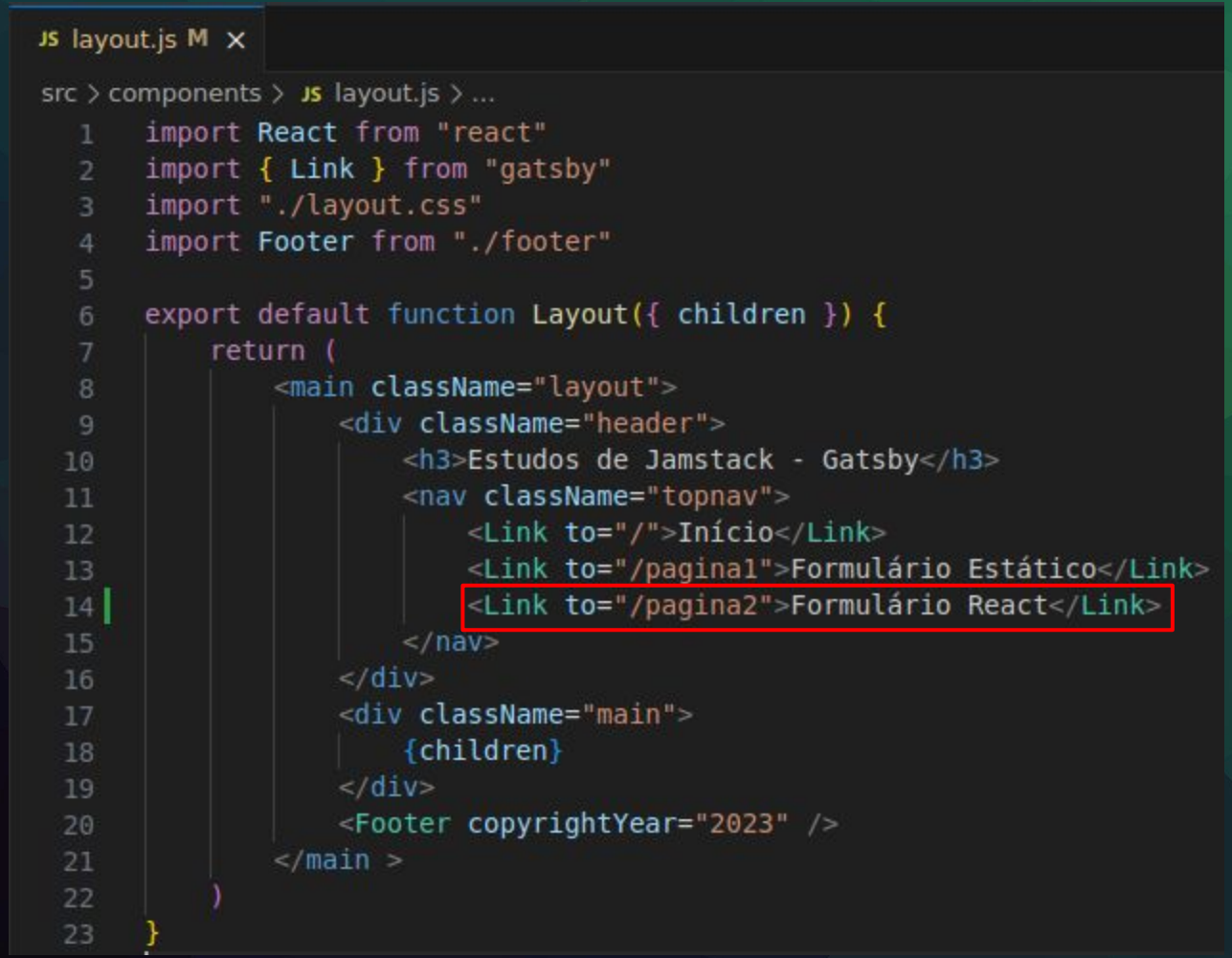
Nome Machado de Assis

Email machado@abl.org.br

Assunto Esse é um teste

Mensagem Muito bom esse teste

Formulário React


JS layout.js M
src > components > JS layout.js > ...
1 import React from "react"
2 import { Link } from "gatsby"
3 import "./layout.css"
4 import Footer from "./footer"
5
6 export default function Layout({ children }) {
7 return (
8 <main className="layout">
9
10 <h3>Estudos de Jamstack - Gatsby</h3>
11 <nav className="topnav">
12 <Link to="/">Início</Link>
13 <Link to="/pagina1">Formulário Estático</Link>
14 <Link to="/pagina2">Formulário React</Link>
15 </nav>
16
17
18 {children}
19
20 <Footer copyrightYear="2023" />
21 </main >
22)
23 }

pagina2.js | pagina1.js

src > pages > pagina2.js > Pagina2 > handleSubmit > then() callback > assunto

```
1  import * as React from "react"
2  import { useState } from "react";
3  import Layout from "../components/layout"
4
5  const Pagina2 = () => {
6
7    const [inputs, setInputs] = useState({ nome: "", email: "", assunto: "", mensagem: "" });
8
9    const handleChange = event => {
10     const name = event.target.name;
11     const value = event.target.value;
12     setInputs(values => ({ ...values, [name]: value }));
13   }
14
15   const encode = (data) => {
16     return Object.keys(data)
17       .map(key => encodeURIComponent(key) + "=" + encodeURIComponent(data[key]))
18       .join("&");
19   }
20
21   const handleSubmit = event => {
22     event.preventDefault();
23     fetch("/", {
24       method: "POST",
25       headers: { "Content-Type": "application/x-www-form-urlencoded" },
26       body: encode({ "form-name": "form_react", ...inputs })
27     }).then(() => {
28       alert("Em breve daremos um retorno do seu contato. Obrigado!");
29       setInputs({ nome: "", email: "", assunto: "", mensagem: "" });
30     }).catch(error => alert(error));
31   };
```

```
const handleSubmit = event => {
  event.preventDefault();
  fetch("/", {
    method: "POST",
    headers: { "Content-Type": "application/x-www-form-urlencoded" },
    body: encode({ "form-name": "form_react", ...inputs })
  }).then(() => {
    alert("Em breve daremos um retorno do seu contato. Obrigado!");
    setInputs({ nome: "", email: "", assunto: "", mensagem: "" });
  }).catch(error => alert(error));
};
```

pagina2.js

src > pages > pagina2.js > Pagina2 > handleSubmit > then() callback > assunto

```
33    return (
34      <Layout>
35        <h2>Entre em Contato:</h2>
36        <div className="container">
37          <form name="form_react" method="post" onSubmit={handleSubmit}
38            data-netlify="true" data-netlify-honeypot="bot-field">
39
40            <input type="hidden" name="form-name" value="form_estatico" />
41            <label>
42              Nome
43              <input type="text" name="nome" value={inputs.nome} onChange={handleChange} />
44            </label>
45            <label>
46              Email
47              <input type="email" name="email" value={inputs.email} onChange={handleChange} />
48            </label>
49            <label>
50              Assunto
51              <input type="text" name="assunto" value={inputs.assunto} onChange={handleChange} />
52            </label>
53            <label>
54              Mensagem
55              <textarea name="mensagem" rows="5" value={inputs.mensagem} onChange={handleChange} />
56            </label>
57            <input type="submit" value="Enviar" />
58            <input type="reset" value="Limpar" />
59          </form>
60        </div>
61      </Layout>
62    )
63  }
```

```
-------------------------------------
git add .
git commit -m "Formulario React"
git push -u origin main
-------------------------------------
```

Forms

Forms Level 0

2 forms collecting data.

Learn more about form handling in the docs ↗

Usage and configuration ↓ | Form notifications

Active forms

form_react
Last submission at 4:21 PM (38 minutes ago)
No submissions yet ›

form_estatico
Last submission at 2:55 PM (2 hours ago)
No submissions yet ›