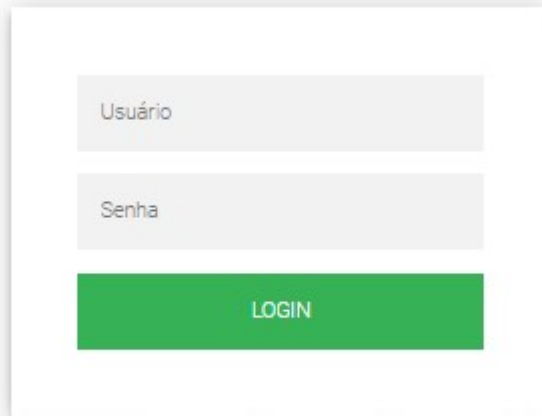


## Explicação do sistema

link → <https://hugocfranciscon.github.io/gerenciador-mineradora/>

1 → Ao acessar o link acima, será apresentada a tela de login abaixo, como exemplo de teste, o login e senha será (admin, admin).



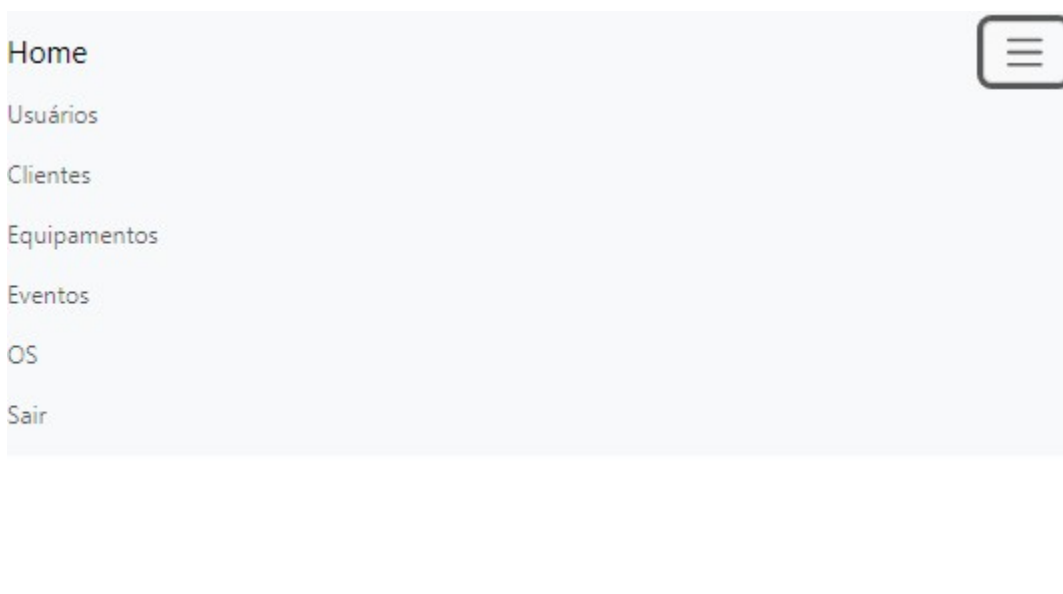
A login form with a white background and a subtle shadow. It contains two light gray input fields stacked vertically. The top field is labeled 'Usuário' and the bottom field is labeled 'Senha'. Below these fields is a solid green button with the word 'LOGIN' in white capital letters.

2 → Após o login realizado, será apresentada a tela de menu abaixo.



3 →

Conforme a imagem abaixo, esses são os menus do sistema.



4 → Conforme o exemplo do cadastro (Cadastro de clientes) abaixo, as funcionalidades são apresentadas pelos botões: “Novo”, “Editar” e “Excluir”.

4.1 → Ao clicar no botão “Novo”, o usuário será direcionado para a tela de cadastro com os campos vazios, esperando preenchimento;

4.2 → Ao clicar no botão “Editar”, o usuário será direcionado para a tela de cadastro com os campos preenchidos, esperando a correção de algum campo;

4.3 → Ao clicar no botão “Excluir”, o sistema fará uma pergunta, para confirmação ou não da exclusão do registro.

Home

Clientes

Novo

Pesquisar...

Pesquisar...

Pesquisar...

Editar	Excluir	hugo	hugo@gmail.com	43988087502
Editar	Excluir	cliente	a@a.a	43999999999

«

«

1

»

»»

5 →  
Tela de  
cadastro  
de  
clientes

Ao ser

direcionado para a tela de cadastro pelos botões “Novo” e “Editar”, os campos abaixo serão apresentados, além dos botões “Confirmar” e “Cancelar”.

Para confirmar o cadastro, todos os campos devem ser preenchidos, e visando evitar problemas, o botão confirmar só será habilitado quando o formulário estiver valido.

Para retornar à tela anterior sem efetuar mudanças, basta clicar no botão “Cancelar”.

6 →  
Dessa  
forma,  
os  
demais

Home

### Cientes

Nome	Fantasia	CNPJ/CPF
<input type="text" value="hugo"/>	<input type="text" value="hugo"/>	<input type="text" value="070.470.609-10"/>
Telefone	Email	
<input type="text" value="(43) 98808-7502"/>	<input type="text" value="hugo@gmail.com"/>	
Cidade	Estado	Logradouro
<input type="text" value="cambira"/>	<input type="text" value="pr"/>	<input type="text" value="teste"/>
Status		
<input type="text" value="Ativo"/>		
<div><div>Confirmar</div><div>Cancelar</div></div>		

cadastros (Usuário, Equipamentos e Eventos) seguem a mesma dinâmica da tela acima mencionada.

7 → Para o caso da “OS”, a tela de consulta segue a mesma dinâmica das anteriores, com os botões “Novo”, “Editar” e “Excluir”.

Home

### OS

Novo

Editar

Excluir

Projeto

Data Inicio

teste	13/11/2023
-------	------------

1

8 → Já na tela de cadastro da OS, além dos dados cadastrais, será exibido uma tabela com os dias em que foram feitas atividades, estas serão lançadas clicando no botão “Novo”, serão alteradas usando o botão “Editar” e excluídas através do botão “Excluir”.

Home

### OS

Cliente:

hugo

Nome: teste

Data Inicial: 13/11/2023

Previsão de termino: 30/11/2023

Valor do projeto: 100,00

Observação: teste

**Confirmar** **Cancelar**

#### Eventos por dia

Novo		Date
<b>Editar</b>	<b>Excluir</b>	01/11/2023
<b>Editar</b>	<b>Excluir</b>	03/11/2023
<b>Editar</b>	<b>Excluir</b>	04/11/2023
<b>Editar</b>	<b>Excluir</b>	05/11/2023
<b>Editar</b>	<b>Excluir</b>	06/11/2023

« « 1 2 » »

9 →

Clicando no botão “Novo” ou “Cancelar” será apresentada a tela abaixo, com o campo “Data Inicial”, que deverá ser informado.

A partir dessa informação lançada, os eventos referentes ao dia poderão ser manipulados, para manipulá-los, deve seguir a mesma regra das telas anteriores, através dos botões “Novo”, “Editar” e “Excluir”.

Home

OS

Data Inicial

01/11/2023

Confirmar

Cancelar

Eventos por dia

Novo

Usuário

Evento

Início

Fim

Editar

Excluir

HUGO

transporte de sonda

15:00

18:00

00:00

00

1

20

20:20

10 →

Clicando nos botões “Novo” ou “Editar” da tela anterior, será apresentada a tela abaixo para informação dos eventos.

Home

OS

Usuário

HUGO

Evento

transporte de sonda

Equipamento

pá carregadeira

Hora Inicial

15:00

Hora Final

18:00

Status

Ativo

Observação

teste

Confirmar

Cancelar

**Explicação dos tópicos do checklist:**

- Criar um repositório no GitHub com a estrutura do Gitflow, incluindo pelo menos as branches principais "main" e "develop."

Link do repositório do github → <https://github.com/hugocfranciscon/gerenciador-mineradora>

- Utilizar componentes de um framework CSS, como Bootstrap, Materialize ou outro à sua escolha.

Para o desenvolvimento foi utilizado o framework Bootstrap.

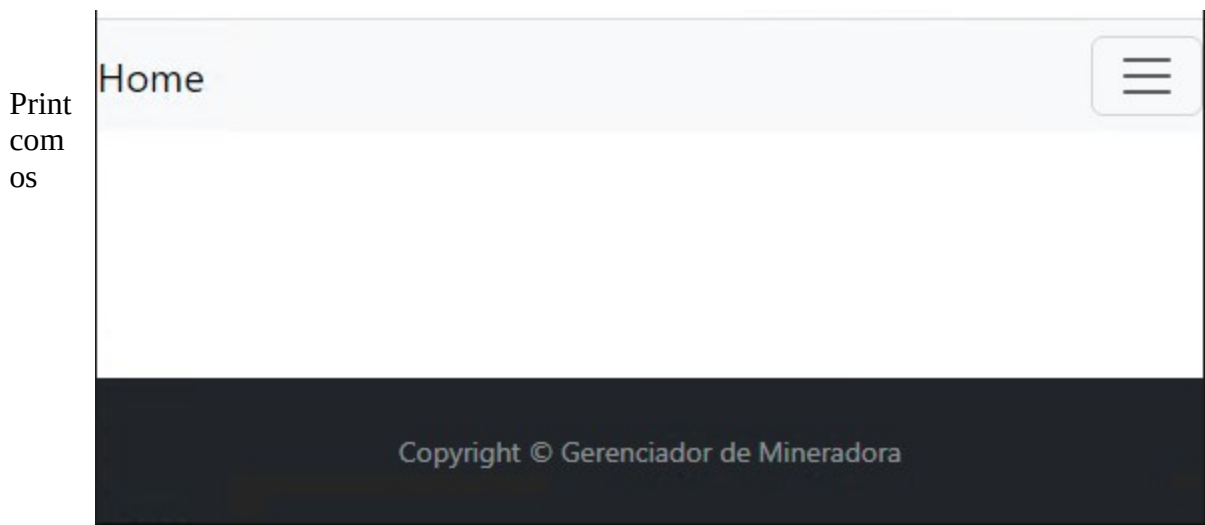
- Apresentar as telas com layout responsivo, adaptando-se a diferentes tamanhos de tela, usando um framework CSS ou implementações personalizadas.

Link para o leiaute das telas feito utilizando o figma →

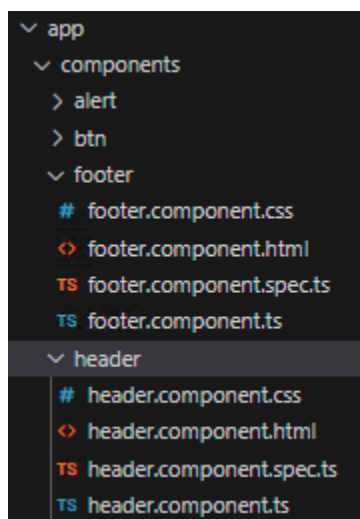
<https://www.figma.com/file/EJX2xgfntz8LtdCTd5hYPa/Gerenciamento-de-mineradora?type=design&node-id=3-5&mode=design&t=ng4w7zUC3qlgdJeJ-0>

- Desenvolver o layout da aplicação com componentes, tornando o cabeçalho e o rodapé componentes reutilizáveis.

Print com os modelo implementado:



componentes gerados:





- Aplicar pelo menos dois tipos de data-binding, como Interpolation, Property Binding, Event Binding, Two-Way Data Binding,

Como exemplo para esse tópico, usarei os componentes referentes ao cadastro de cliente:

```
<tbody>
  <tr class="align-middle" *ngFor="let u of filtering(); index as i">
    <div class="mb-3 col-md-4 col-sm-12">
      <label for="input-name" class="form-label">Nome</label>
      <input
        type="text"
        class="form-control form-control-sm"
        id="input-name"
        name="input-name"
        [(ngModel)]="this.customer.name"
        #name="ngModel"
        required>
    </div>
    <td>{{ u.status == "A" ? "Ativo" : "Inativo" }}</td>
  </tr>
</tbody>
```

- Empregar variáveis de template e a anotação ViewChild para interagir com elementos do DOM ou componentes diretamente no template ou no código TypeScript do aplicativo.

Como exemplo também estou printando os dados do cadastro de clientes:

```
export class CustomerPostComponent implements OnInit {
  @ViewChild(AlertComponent) alertComponent: AlertComponent | undefined;
  @ViewChild('form') form!: NgForm;
}
```

```

<app-loading [hidden]="!loading"></app-loading>
<form [hidden]="loading" class="col s12" #form="ngForm" (submit)="confirmCustomer()">
  <div class="row">...
  </div>
  <div class="row">...
  </div>
  <div class="row">...
  </div>
  <div class="error-message" [hidden]="cpfCnpj.valid || cpfCnpj.untouched">
    Por favor, informe um CPF/CNPJ.
  </div>
  <div>
    <app-btn classes="btn-success" type="submit" label="Confirmar" [disabled]="!form.valid"></app-btn>
    <app-btn classes="btn-danger" (click)="cancelCustomer()" label="Cancelar"></app-btn>
  </div>
</form>
<app-alert></app-alert>

```

- Estabelecer a passagem de dados entre componentes por meio da hierarquia de componentes, empregando as anotações @Input e @Output.

Exemplo do componente dos botões

```

import { Component, OnInit, Input, Output, EventEmitter } from '@angular/core';

You, 3 weeks ago | 1 author (You)
@Component({
  selector: 'app-btn',
  templateUrl: './btn.component.html',
  styleUrls: ['./btn.component.css'],
})
export class BtnComponent implements OnInit {
  @Input() type: string = '';
  @Input() classes: string = '';
  @Input() tooltip: string = '';
  @Input() label: string = '';
  @Input() disabled: boolean = false;

  constructor() {}

  ngOnInit(): void {}
}

```

Transferir dados, por meio de serviços, entre componentes que não estão diretamente relacionados.

Serviço usado para realizar os requests:

```
import { Injectable } from '@angular/core';
import { HttpClient } from '@angular/common/http';
import { environment } from 'src/environments/environment';
import { Observable } from 'rxjs';
```

You, 1 second ago | 1 author (You)

```
@Injectable({
  providedIn: 'root',
})
export class RequestService {
  constructor(private http: HttpClient) {}

  async get(url: string) {
    return new Promise(resolve => {
      this.http.get(environment.url + url).subscribe(
        (data) => {
          resolve(data);
        },
        (error) => {
          resolve(error);
        }
      );
    });
  }
}
```

(parameter) body: any

body: any

```
post(url: string, body: any): Observable<any> {
  let lUrl = environment.url + url;
  return this.http.post(lUrl, body);
}
```

```
put(url: string, body: any): Observable<any> {
  let lUrl = environment.url + url;
  return this.http.put(lUrl, body);
}
```

```
delete(url: string, body: any): Observable<any> {
  let lUrl = environment.url + url;
  return this.http.delete(lUrl, body);
}
```

- Mapear os componentes às rotas no módulo de rotas, criando uma estrutura de navegação eficiente.

```
import { NgModule } from '@angular/core';
import { RouterModule, Routes } from '@angular/router';
import { LoginComponent } from '../login/login.component';
import { HomeComponent } from '../home/home.component';
import { UsersComponent } from '../users/users.component';
import { CustomersComponent } from '../customers/customers.component';
import { EquipamentComponent } from '../equipament/equipament.component';
import { EventComponent } from '../event/event.component';
import { OsComponent } from '../os/os.component';
import { UserPostComponent } from '../user-post/user-post.component';
import { UserConsComponent } from '../user-cons/user-cons.component';
import { CustomerConsComponent } from '../customer-cons/customer-cons.component';
import { CustomerPostComponent } from '../customer-post/customer-post.component';
import { EquipamentConsComponent } from '../equipament-cons/equipament-cons.component';
import { EquipamentPostComponent } from '../equipament-post/equipament-post.component';
import { EventPostComponent } from '../event-post/event-post.component';
import { EventConsComponent } from '../event-cons/event-cons.component';
import { OsConsComponent } from '../os-cons/os-cons.component';
import { OsPostComponent } from '../os-post/os-post.component';
import { OsDayComponent } from '../os-day/os-day.component';
import { OsDayEventsComponent } from '../os-day-events/os-day-events.component';

const routes: Routes = [
  {
    path: '',
    component: LoginComponent,
  },
  {
    path: 'login',
    component: LoginComponent,
  },
  {
    path: 'home',
    component: HomeComponent,
    children: [
      // You, last month • Criação de componentes ...
    ],
  },
];
```

- • Permitir a navegação fluida entre as diferentes páginas do aplicativo por meio de links e botões de navegação.

```
<div class="container">
  newUser() {
    this.router.navigate(['/home/users/post', 0]);
  }

  editUser(u: any) {
    this.router.navigate(['/home/users/post/', u.id]);
  }
}
```

- Validar os campos do formulário com expressões regulares (REGEX) e apresentar as mensagens de erro.

```

<div class="mb-3 col-md-4 col-sm-12">
  <label for="input-password" class="form-label">Senha</label>
  <input
    type="password"
    class="form-control form-control-sm"
    id="input-password"
    name="input-password"
    [(ngModel)]="this.user.password"
    pattern="^[a-zA-Z0-9_@#\\&\\$]{6,18}$"
    #password="ngModel"
    required>
</div>

```

- Implementar máscaras em campos de formulário, quando necessário, para melhorar a experiência do usuário ao inserir dados.

```

<div class="mb-3 col-md-6 col-sm-12">
  <label for="input-fone" class="form-label">Telefone</label>
  <input
    type="text"
    class="form-control form-control-sm"
    id="input-fone"
    name="input-fone"
    [(ngModel)]="this.user.fone"
    #fone="ngModel"
    [mask]="'(00) 00000-0000'"
    maxLength="15"
    minLength="14"
    required>
</div>

```

- Desabilitar o botão de envio (submit) enquanto o formulário estiver em um estado inválido.

```

<div>
  <app-btn classes="btn-success" type="submit" label="Confirmar" [disabled]=!form.valid></app-btn>
  <app-btn classes="btn-danger" (click)="cancelUser()" label="Cancelar"></app-btn>
</div>

```

- Realizar requisições à API com tratamento adequado das respostas de sucesso e erro com Promises.

```

async get(url: string) {
  return new Promise((resolve) => {
    this.http.get(environment.url + url).subscribe(
      (data) => {
        resolve(data);
      },
      (error) => {
        resolve(error);
      }
    );
  });
}
}

```

```

getUsers() {
  this.loading = true;
  this.req.get('users').then(
    (ret: any) => {
      this.loading = false;
      if (ret.status == 'erro') {
        alert(ret.msg);
        return;
      }
      this.users = ret;
    },
    (err: any) => {
      this.loading = false;
      alert('ERRO ' + err);
    }
  );
}
}

```

- Realizar requisições à API com tratamento adequado das respostas de sucesso e erro com Observables.

```

put(url: string, body: any): Observable<any> {
  let lUrl = environment.url + url;
  return this.http.put(lUrl, body);
}

```

```

    this.req.put('users/' + this.user.id, this.user).subscribe(
      (ret: any) => {
        if (ret.status == 'erro') {
          this.loading = false;
          alert(ret.msg);
          return;
        }
        this.alertComponent?.showModal(
          'Sucesso',
          'Usuário salvo com sucesso.'
        );
        this.router.navigate(['/home/users']);
      },
      (err: any) => {
        this.loading = false;
        alert('ERRO ' + err);
      }
    );
  }
}

```

- Criar o cadastro completo de uma entidade, incluindo operações de criação, leitura, atualização e exclusão (CRUD) utilizando uma API, como o JSON Server.

```
import { Component, OnInit, ViewChild } from '@angular/core';
import { AlertComponent } from '../components/alert/alert.component';
import { NgForm } from '@angular/forms';
import { Customer } from '../models/Customer';
import { RequestService } from '../services/request.service';
import { ActivatedRoute, Router } from '@angular/router';
```

You, 1 minute ago | 1 author (You)

```
@Component({
  selector: 'app-customer-post',
  templateUrl: './customer-post.component.html',
  styleUrls: ['./customer-post.component.css'],
})
export class CustomerPostComponent implements OnInit {
  @ViewChild(AlertComponent) alertComponent: AlertComponent | undefined;
  @ViewChild('form') form!: NgForm;

  public loading: boolean = false;
  public customer!: Customer;

  constructor(
    private req: RequestService,
    private route: ActivatedRoute,
    private router: Router
  ) {}

  ngOnInit(): void { ...
  }

  confirmCustomer() {
    this.loading = true;
    if (this.customer?.id != undefined) {
      this.req.put('customers/' + this.customer.id, this.customer).subscribe(...
    );
    return;
  }
  this.req.post('customers', this.customer).subscribe(...
  );
}

cancelCustomer() { ...
}
}
```



```

You, 2 minutes ago | 1 author (You)
import { Component, OnInit } from '@angular/core';
import { FilterService } from '../services/filter.service';
import { RequestService } from '../services/request.service';
import { Router } from '@angular/router';

You, 2 minutes ago | 1 author (You)
@Component({
  selector: 'app-customer-cons',
  templateUrl: './customer-cons.component.html',
  styleUrls: ['./customer-cons.component.css'],
})
export class CustomerConsComponent implements OnInit {
  public loading: boolean = false;
  public formFilter: any = {};
  public customers: any = [];
  public page: number = 1;
  public pageSize: number = 8;
  public totalItens: number = 8;

  constructor(
    private filter: FilterService,
    private req: RequestService,
    private router: Router
  ) {}

  ngOnInit(): void {
    this.getCustomers();
  }

  > filtering() { ...
  }

  > newCustomer() { ...
  }

  > editCustomer(c: any) { ...
  }

  deleteCustomer(c: any) {
    if (!window.confirm('Deseja realmente excluir?')) {
      return;
    }
    this.loading = true;
    > this.req.delete('customers/' + c.id, c).subscribe( ...
    );
  }

  getCustomers() {
    this.loading = true;
    > this.req.get('customers').then( ...
    );
  }
}

```

- Utilizar o armazenamento local (LocalStorage ou SessionStorage) para armazenar dados temporários, quando necessário.

```

async newDay() {
  await window.localStorage.setItem('os', JSON.stringify(this.os));
  this.router.navigate(['/home/os/day', 0]);
}
You, 2 days ago • Final Atividade 13

```

```

let o = window.localStorage.getItem('os');
if (o) {
  this.os = JSON.parse(o);
  this.day.os = this.os;
}

```

- Aplicar a diretiva estrutural ngFor para apresentar uma lista dinâmica de dados em seu aplicativo.

```

<tbody>
  <tr class="align-middle" *ngFor="let u of filtering(); index as i">
    <td>...
  </td>
  <td>{{ u.name }}</td>|      You, 3 weeks ago • Atividade 11
  <td>{{ u.email }}</td>
  <td>{{ u.fone }}</td>
  <td>{{ u.status == "A" ? "Ativo" : "Inativo" }}</td>
</tr>
</tbody>

```

- Utilizar a diretiva ngIf para controlar a exibição ou ocultação de elementos com base em condições específicas.

```

Go to component | You, 3 weeks ago | 1 author (you)
<button [type]="type" class="btn btn btn-register" [ngClass]="classes" [title]="toolTip" [disabled]="disabled">
  <span *ngIf="label" [innerHTML]="label"></span>
</button>      You, 3 weeks ago • Atividade 09 finalizada

```

- Formatar a apresentação de dados com Pipes, de acordo com os requisitos do aplicativo.

```

<tbody>
  <tr class="align-middle" *ngFor="let u of filtering(); index as i">
    <td>
      <button class="btn btn-sm btn-success me-1" (click)="editOs(u)">
        Editar
      </button>
      <button class="btn btn-sm btn-danger me-1" (click)="deleteOs(u)">
        Excluir
      </button>
    </td>
    <td>{{ u.name }}</td>
    <td>{{ u.initialDate | date: 'dd/MM/YYYY' }}</td>
  </tr>|      You, 4 days ago • Inclusão de novos componentes

```

- Executar o processo de build da aplicação e realizar o deploy para tornar o aplicativo acessível online.

```
c> gerenciador@0.0.0 deploy
r> npm run build & npm run ghpages

t
r> gerenciador@0.0.0 build
t> ng build --configuration production

r✓ Browser application bundle generation complete.
✓ Copying assets complete.
l- Generating index html...3 rules skipped due to selector errors:
  legend+* -> Cannot read properties of undefined (reading 'type')
  .form-floating>~label -> Did not expect successive traversals.
  .form-floating>~label -> Did not expect successive traversals.
r✓ Index html generation complete.

Initial Chunk Files | Names | Raw Size | Estimated Transfer Size
main.f7ce1f71a9c700f6.js | main | 469.05 kB | 110.31 kB
styles.2c7b29ebf9dcdbdee.css | styles | 219.31 kB | 21.95 kB
polyfills.dee4d8c28fa93d35.js | polyfills | 34.38 kB | 11.12 kB
runtime.305e5dd7ef79734a.js | runtime | 900 bytes | 516 bytes
| Initial Total | 723.61 kB | 143.89 kB

lBuild at: 2023-11-16T18:44:37.908Z - Hash: 849bfa294d81fedb - Time: 75650ms

Warning: bundle initial exceeded maximum budget. Budget 500.00 kB was not met by 223.61 kB with a total of 723.61 kB.

> gerenciador@0.0.0 ghpages
> npx angular-cli-ghpages --dir=dist/gerenciador

ⓧ Uploading via git, please wait...
ⓧ Successfully published via angular-cli-ghpages! Have a nice day!
```