

Cadastro *create*

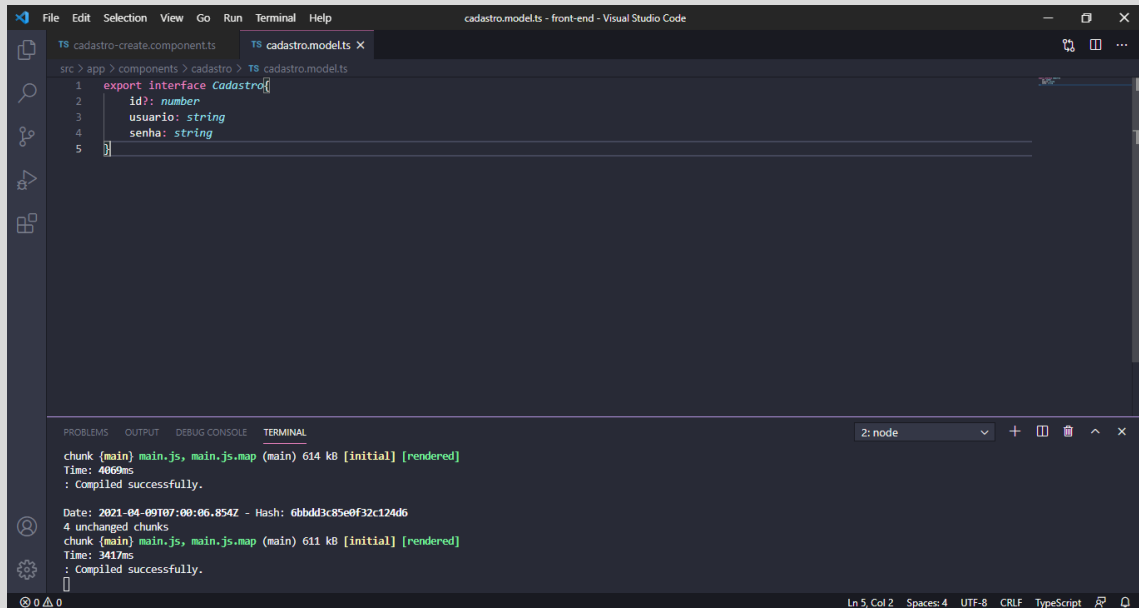
Aprenderemos, agora, a criar uma requisição para um formulário do tipo login. No arquivo Front-com-angular.zip, em `src\app\componentes`, você encontrará as pastas separadas por temas, de acordo com a arquitetura do Angular.

1. Para acessar a tela de cadastro, utilize o caminho: `src\app\componentes\cadastro\cadastro-create`.

2. Clique com o botão direito do mouse sobre a pasta **cadastro**, depois, em “*new file*” e crie um arquivo chamado **cadastro.model.ts**. Ao criarmos uma interação com o Back-End, essa nova interação irá procurar um arquivo que possua seus atributos definidos. No arquivo `db.json`, é possível apenas adicionar dados novos, além de atualizar, listar e remover dados que já estão adicionados.

Após criarmos o arquivo, devemos passar os atributos, que serão os mesmo que já estão no arquivo `db.json`. O `id` é opcional, pois o usuário não o fornece; é o próprio Back-End, ao salvar os dados, que gera esse `id`.

3. Abra o arquivo `cadastro.models.ts` no Visual Studio Code, adicione o comando ***export interface Cadastro*** (toda interface começa com letra maiúscula), abra chaves e passe os atributos com seus respectivos tipos. Salve o arquivo.



The screenshot shows the Visual Studio Code editor with the file `cadastro.models.ts` open. The code defines an interface `Cadastro` with three properties: `id` of type `number`, `usuario` of type `string`, and `senha` of type `string`. The terminal at the bottom shows the output of the `ng g s componentes/cadastro/cadastro` command, indicating that the service was created successfully.

```
src > app > components > cadastro > TS cadastro.models.ts
1 export interface Cadastro{
2   id?: number
3   usuario: string
4   senha: string
5 }
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

chunk (main) main.js, main.js.map (main) 614 kB [initial] [rendered]
Time: 4069ms
: Compiled successfully.

Date: 2021-04-09T07:00:06.854Z - Hash: 6bbd3c85e0f32c124d6
4 unchanged chunks
chunk (main) main.js, main.js.map (main) 611 kB [initial] [rendered]
Time: 3417ms
: Compiled successfully.

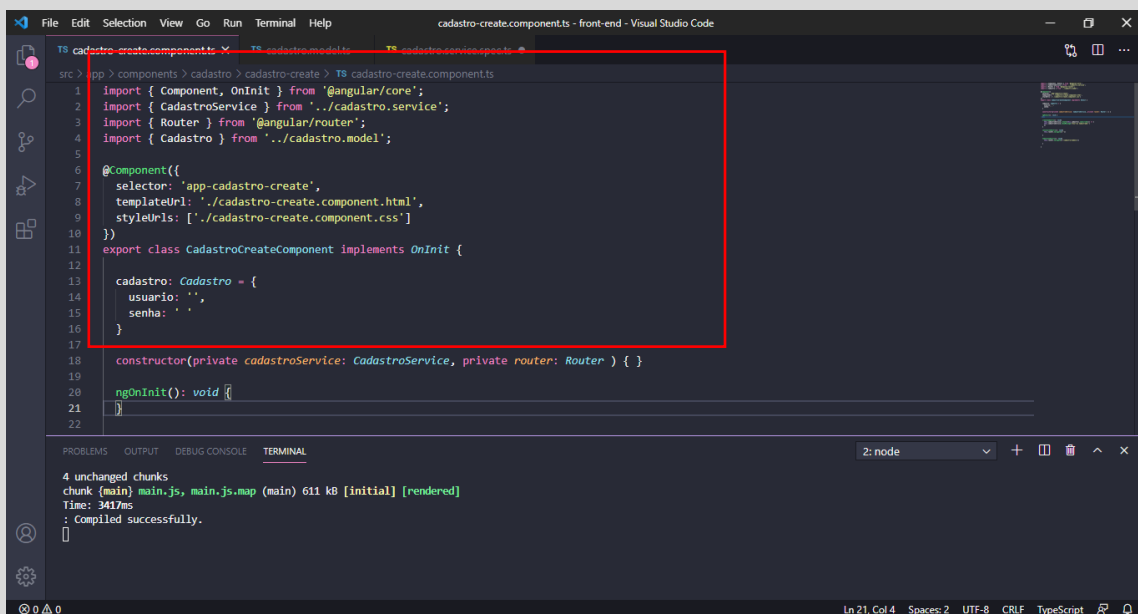
Ln 5, Col 2 Spaces: 4 UTF-8 CRLF TypeScript

4. Abra o projeto no terminal e crie o *service* digitando no terminal o comando ***ng g s componentes/cadastro/cadastro***; para finalizar, aperte “*enter*”. O sistema criará um arquivo *service*, que é usado para implementar regras, acessar o Back-End e outros arquivos HTTP, entre outras funções.

5. Acesse o arquivo **cadastro-create.component.ts** e faça o seguinte procedimento:

- importe os arquivos *model* e *service* que você criou (linhas 2 e 4).
- instancie os dados da *model* (linha 13).
- No *constructor*, insira o *cadastroService* (linha 18) e o *router* (importado na linha 3).

Agora, podemos migrar o conteúdo dos demais arquivos para o arquivo `typeScript (.ts)`.



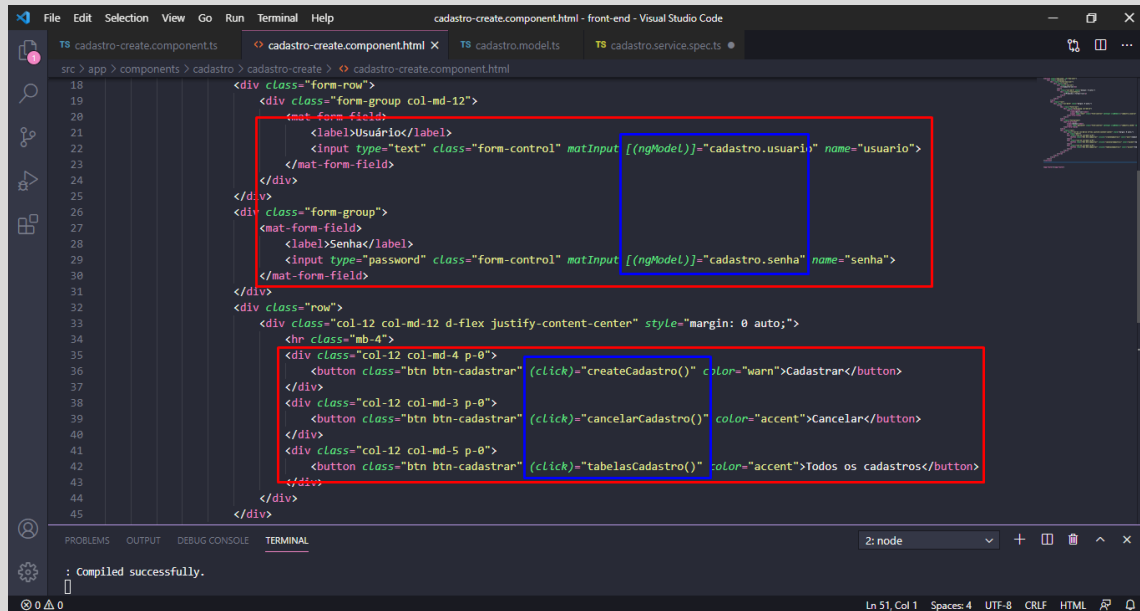
```
src > app > components > cadastro > cadastro-create > TS cadastro-create.component.ts
1 import { Component, OnInit } from '@angular/core';
2 import { CadastroService } from '../cadastro.service';
3 import { Router } from '@angular/router';
4 import { Cadastro } from '../cadastro.model';
5
6
7 @Component({
8   selector: 'app-cadastro-create',
9   templateUrl: './cadastro-create.component.html',
10  styleUrls: ['./cadastro-create.component.css']
11 })
12 export class CadastroCreateComponent implements OnInit {
13
14   cadastro: Cadastro = {
15     usuario: '',
16     senha: ''
17   }
18
19   constructor(private cadastroService: CadastroService, private router: Router) {}
20
21   ngOnInit(): void {}
22 }
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL 2: node

4 unchanged chunks
chunk {main} main.js, main.js.map (main) 611 kB [initial] [rendered]
Time: 3417ms
: Compiled successfully.

Ln 21, Col 4 Spaces: 2 UTF-8 CRLF TypeScript

6. No arquivo **cadastro-create.component.html**, adicione as seguintes linhas:



```
18
19 <div class="form-row">
20   <div class="form-group col-md-12">
21     <mat-form-field>
22       <label>Usuário</label>
23       <input type="text" class="form-control" matInput [(ngModel)]="cadastro.usuario" name="usuario">
24     </mat-form-field>
25   </div>
26   <div class="form-group">
27     <mat-form-field>
28       <label>Senha</label>
29       <input type="password" class="form-control" matInput [(ngModel)]="cadastro.senha" name="senha">
30     </mat-form-field>
31   </div>
32 </div>
33 <div class="row">
34   <div class="col-12 col-md-12 d-flex justify-content-center" style="margin: 0 auto;">
35     <div class="col-12 col-md-4 p-0">
36       <button class="btn btn-cadastrar" (click)="createCadastrar()" color="warn">Cadastrar</button>
37     </div>
38     <div class="col-12 col-md-3 p-0">
39       <button class="btn btn-cadastrar" (click)="cancelarCadastrar()" color="accent">Cancelar</button>
40     </div>
41     <div class="col-12 col-md-5 p-0">
42       <button class="btn btn-cadastrar" (click)="tabelasCadastrar()" color="accent">Todos os cadastros</button>
43     </div>
44   </div>
45 </div>
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

2: node

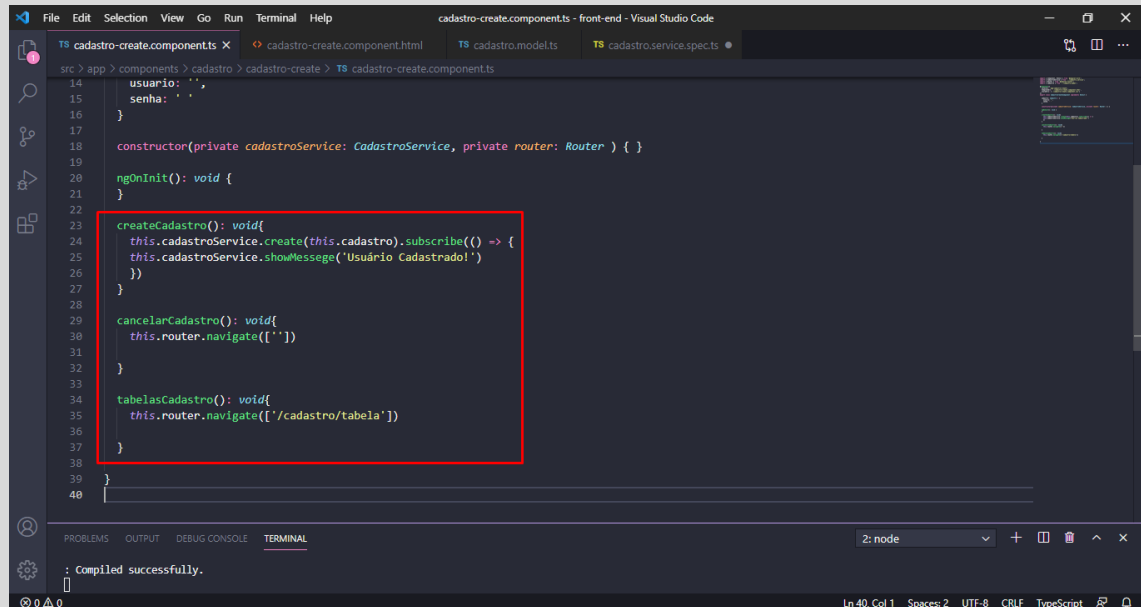
Ln 51, Col 1 Spaces: 4 UTF-8 CRLF HTML

Compiled successfully.

Em `[(ngModel)]` (primeiro destaque, em azul), estamos passando os dados desse formulário para os atributos criados na *model* e instanciados no arquivo `cadastro-create.component.ts`.

Já a função `(click)` realizará algum evento por meio da interação do usuário, ou seja, o clique (segundo destaque, em azul). Se você salvar desse modo, o VSC retornará erros no terminal, informando que os eventos do botão não foram criados.

7. Desse modo, antes de salvar, acesse o arquivo `cadastro-create.component.ts` e adicione as seguintes linhas:

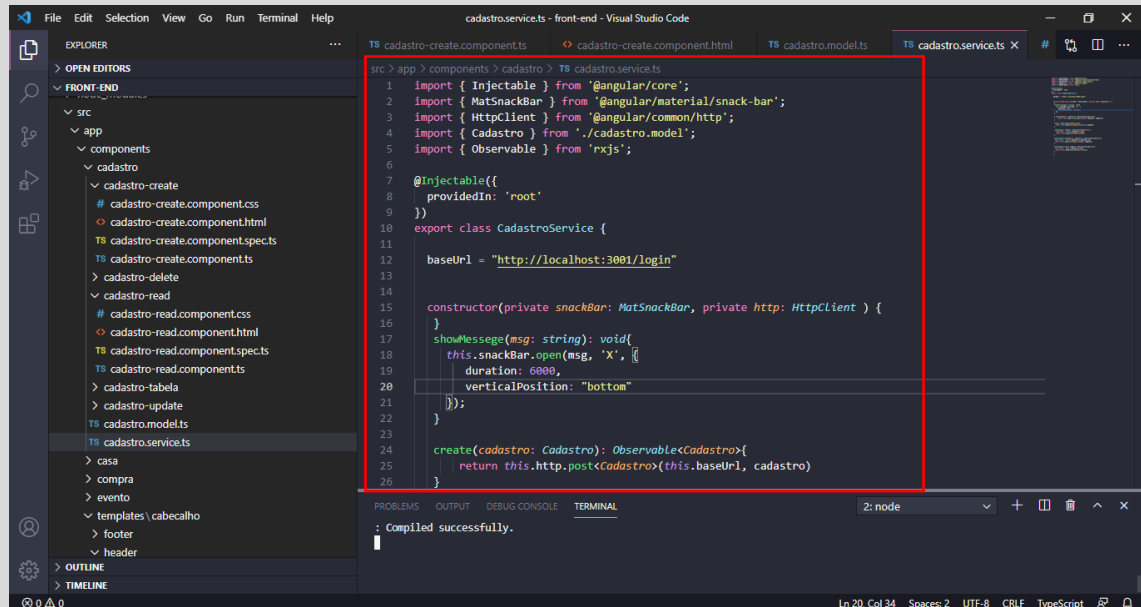


```
14     usuario: '',
15     senha: ''
16   }
17
18   constructor(private cadastroService: CadastroService, private router: Router ) { }
19
20   ngOnInit(): void {
21   }
22
23   createCadastro(): void{
24     this.cadastroService.create(this.cadastro).subscribe() => {
25       this.cadastroService.showMessege("Usuário Cadastrado!")
26     }
27   }
28
29   cancelarCadastro(): void{
30     this.router.navigate([''])
31   }
32
33   tabelasCadastro(): void{
34     this.router.navigate(['/cadastro/tabela'])
35   }
36 }
37
38
39
40 }
```

Na linha 24, estamos criando o cadastro com a palavra “*create*”.

Observação: O *router* é um arquivo de configuração Front-End que serve para fazer a rota de acesso às páginas internas. Para ver as rotas, acesse o arquivo `Front-com-angular\src\appapp-routing.module.ts`.

8. Volte a ao arquivo *service* e adicione as seguintes linhas:

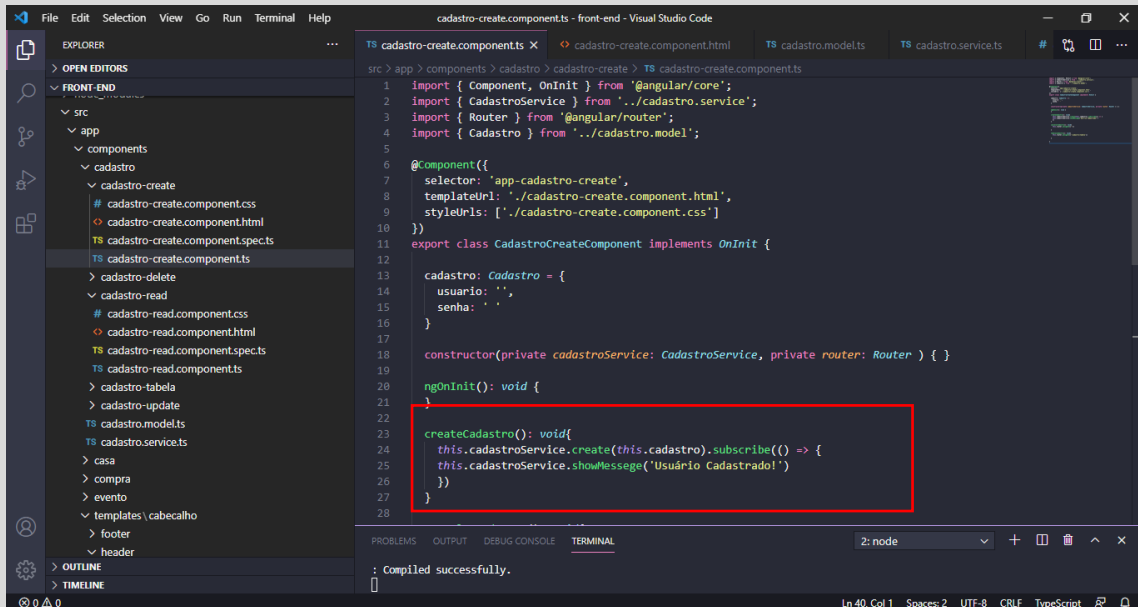


```
src > app > components > cadastro > TS cadastro.service.ts
1 import { Injectable } from '@angular/core';
2 import { MatSnackBar } from '@angular/material/snack-bar';
3 import { HttpClient } from '@angular/common/http';
4 import { Cadastro } from './cadastro.model';
5 import { Observable } from 'rxjs';
6
7 @Injectable({
8   providedIn: 'root'
9 })
10 export class CadastroService {
11
12   baseUrl = "http://localhost:3001/login"
13
14   constructor(private snackBar: MatSnackBar, private http: HttpClient) {
15
16   }
17   showMessege(msg: string): void {
18     this.snackBar.open(msg, 'X', {
19       duration: 6000,
20       verticalPosition: "bottom"
21     });
22   }
23
24   create(cadastro: Cadastro): Observable<Cadastro>{
25     return this.http.post<Cadastro>(this.baseUrl, cadastro)
26   }
27 }
```

Com esse código, estamos passando a URL e criando o método de *create*, a partir do qual o sistema adicionará as informações passadas pelo usuário, por meio de um *input* via POST, ao banco de dados.

Na linha 17, estamos criando um modal para apresentar uma mensagem quando a ação de deletar, atualizar ou criar um recurso for realizada (mensagem esta que ainda deverá ser configurada).

9. No arquivo **cadastro-create.component.ts**, adicione as seguintes linhas de código:



```
1 import { Component, OnInit } from '@angular/core';
2 import { CadastroService } from '../cadastro.service';
3 import { Router } from '@angular/router';
4 import { Cadastro } from '../cadastro.model';
5
6 @Component({
7   selector: 'app-cadastro-create',
8   templateUrl: './cadastro-create.component.html',
9   styleUrls: ['./cadastro-create.component.css']
10 })
11 export class CadastroCreateComponent implements OnInit {
12
13   cadastro: Cadastro = {
14     usuario: '',
15     senha: ''
16   }
17
18   constructor(private cadastroService: CadastroService, private router: Router) {}
19
20   ngOnInit(): void {
21
22
23     createCadastro(): void {
24       this.cadastroService.create(this.cadastro).subscribe(() => {
25         this.cadastroService.showMessage('Usuário Cadastrado!')
26       })
27     }
28
29   }
30 }
```

Com elas, estamos passando o método que criamos em *service* e chamando-o no código *components*. Perceba que, na linha 25, estamos passando a mensagem “Usuário Cadastrado”.

10. Salve o código (ctrl + s) e veja o resultado no navegador.

VP-EVENTO +1 CASA DE SHOW +1 EVENTO MINHA PÁGINA

Cadastre-se

Preencha o formulário

Usuário







Senha

CADASTRAR CANCELAR TODOS OS CADASTROS

Note que, ao clicar em “Todos os cadastros”, serão relacionados os cadastrados que adicionamos em db.json.

Usuários cadastrados

Informações mais relevantes

Id	Usuário	Senha	Ações
1	Vitória Pinho	admin	 
2	Stefany Oliveira	admin123	 
3	Maria da Silva	admin123	 

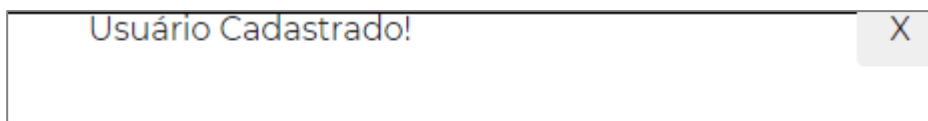
HOME CADASTRAR TODOS OS CADASTROS

11. Agora, ao clicar em “Cadastrar”, é possível adicionarmos um novo usuário. Para isso, será necessário preencher as informações: usuário e senha. Os dados informados são fictícios, ou seja, servem apenas para representar a funcionalidade do código. Como não existe um limite, você poderá adicionar quantos novos usuários quiser.



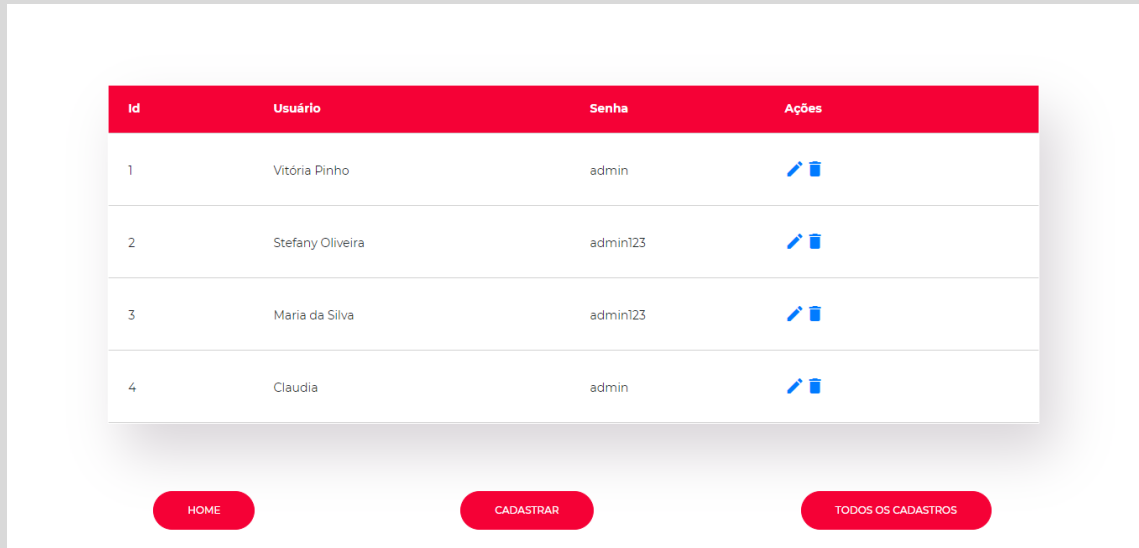
A interface de cadastro, intitulada "Cadastre-se", apresenta o subtítulo "Preencha o formulário". Ela contém dois campos de entrada: "Usuário" com o texto "Claudia" e "Senha" com caracteres ocultos por pontos. Abaixo dos campos, há três botões: "CADASTRAR", "CANCELAR" e "TODOS OS CADASTROS". No rodapé, uma linha preta contém o texto de direitos reservados: "© Vítória Pinho, 2020. Todos os direitos estão reservados."

Perceba que, ao clicar em “Cadastrar”, será apresentado o modal com a mensagem que inserimos:











O modal exibe a mensagem "Usuário Cadastrado!" em uma caixa de diálogo com uma borda cinza. No canto superior direito, há um botão de fechar com o ícone "X".

12. Para confirmar, clique em “Todos os cadastros” e repare que o novo usuário foi inserido.



The screenshot shows a web application interface. At the top, there is a table with four columns: 'Id', 'Usuário', 'Senha', and 'Ações'. The table contains four rows of user data. Below the table, there are three red buttons: 'HOME', 'CADASTRAR', and 'TODOS OS CADASTROS'.

Id	Usuário	Senha	Ações
1	Vitória Pinho	admin	 
2	Stefany Oliveira	admin123	 
3	Maria da Silva	admin123	 
4	Claudia	admin	 

HOME CADASTRAR TODOS OS CADASTROS

Se desejar, você pode testar utilizando o Postman, mas será necessário passar a URL.