



JAVA

*Ser acessível é abrir portas para um mundo onde todos têm igualdade de oportunidades,  
independentemente das suas diferenças.*

## Equipe

Cauã Ferrigoli Loureiro – RM 553093

Claudio Silva Bispo – RM 553472

Sara Ingrid da Silva Pereira – RM 554021

# Sumário

---

Introdução.....	4
Contexto.....	4
Desafio.....	4
Definição do projeto .....	5
Objetivo Geral .....	5
Objetivo Específico.....	6
Escopo do projeto - Java .....	8
Design do Diagrama de classe .....	8
Sprint 1.....	8
Sprint 2.....	9
Entidades .....	9
Atributos e métodos .....	10
1. Usuário: .....	10
2. CadastroUsuario: .....	10
3. Login .....	11
4. Gerenciamento de Banco de Dados: .....	11
5. Monitoramento da Jornada do Usuário:.....	11
6. Atualização do Conteúdo ou Sinalização de Melhorias:.....	12
7. Idioma de Interação: .....	12
8. Segurança.....	12
9. Funcionário .....	13
10. Plataforma .....	13
Primeira entrega .....	14
Cadastrar o usuário .....	14

Tela.....	14
Código .....	14
Login.....	17
Tela.....	17
Código .....	18
Autenticação .....	20
Código .....	20
Definição do idioma.....	21
Tela.....	21
Código .....	21
Main .....	23
Tela.....	24
Código .....	24
Segunda entrega .....	26
Objetivo e atividades realizadas.....	26
Fluxo no terminal .....	26
Código completo .....	31
Interface para o cadastro do usuário.....	31
Classe: Cadastro do usuário .....	31
Classe: Usuário.....	34
Classe: Login.....	35
Classe: Menu .....	37
Main – Processo principal .....	37
Conclusão .....	39

# Introdução

---

Imagine um mundo digital onde todos têm acesso fácil e igualitário à informação, independentemente de suas necessidades. Este é o visionário projeto que estamos empenhados em tornar realidade. Como acessar um site e poder escolher o tema? Como ler as informações contidas sem ter que dar um zoom na tela pois a fonte está pequena ou com uma cor que dificulta leitura e entendimento. Já pensou em fazer uma busca usando áudio ao invés de texto?

Nosso projeto tem o objetivo de criar facilidade, simplicidade e incluir suportes que vão ajudar a qualquer tipo de pessoa a localizar um conteúdo com facilidade e sem burocracia.

## Contexto

---

No início, identificamos uma carência significativa na acessibilidade de muitos elementos existentes no site em que estamos estudando e criando melhorias. Pessoas com diferentes habilidades e necessidades estavam sendo abandonadas de diversas maneiras, até mesmo pessoas sem deficiência, que não conhece sobre os produtos da companhia, tinham dificuldades para localizar informações de formas simples e sem burocracia. Imagina que toda vez que precisar entrar em contato com alguém ou ter que assistir a um vídeo, ter que preencher um formulário? Imagine ter que preencher um formulário a cada interação? Ou navegar em várias abas para localizar uma determinada informação e ela estiver no idioma inglês e não ter uma opção para trocar para um idioma base. Imagine deseja tirar dúvidas de forma dinâmica, apenas fazendo perguntas e desejar respostas rápidas e não ter isso disponível.

## Desafio

---

Desenvolver um site que fosse verdadeiramente inclusivo e que consiga atrair o máximo de pessoas possíveis, criando leads para novos negócios. Tínhamos que pensar em cada detalhe, desde o esquema de cores até a funcionalidade de leitura por voz, para que ninguém ficasse de fora e que atenda a todos os requisitos e necessidades. Tornar um site mais inclusivo para pessoas com ou sem deficiência. Com o uso da linguagem Java, poderemos integrar o front como back-end, além de criar uma interface onde o usuário vai realizar consultas no banco, sem precisar conhecer sobre linguagem de programação e gerar insights para novas oportunidades.

# Definição do projeto

---

Nosso projeto visa criar um site altamente acessível, que oferece uma experiência inclusiva para todos os usuários, sendo eles com ou sem deficiência, eles tendo ou não conhecimento sobre os produtos da empresa foco, independentemente de suas necessidades e capacidades. Ele integra funcionalidades avançadas, como autenticação simplificada através de plataformas populares (G-mail, LinkedIn, GitHub, Outlook, Apple), formulários de contato adaptáveis para entrada de texto ou voz, e um chatbot inteligente para assistência instantânea. Busca por texto e voz, leitura do conteúdo da tela, definição do idioma em várias linguagens, escolha do tema da preferência da pessoa.

## Objetivo Geral

---

1. Investimos tempo considerável em pesquisas e testes para garantir que nosso site atenda a todas as necessidades. Queremos implementar uma variedade de recursos inovadores, sendo eles:
2. Esquema de Cores Personalizável: Permitimos aos usuários escolher entre temas de tela clara ou escura, garantindo uma experiência confortável para todos.
3. Fontes Adaptáveis e Tamanhos Personalizáveis: Desenvolver uma interface que permite ajustar a fonte e seu tamanho, garantindo que a leitura seja fácil para todos. Aqui será criado uma opção visível no navegador onde o usuário vai conseguir configurar a fonte que deseja aplicar no site e tamanho, a cada acesso que fizer, esses padrões já estarão salvos e será aplicado de forma automática no site.
4. Aceite em vários tipos de tecnologias como celular com visualização na horizontal ou vertical, uso em tablets, computadores, Laptops e até mesmo TV.
5. Navegação no site apenas usando o teclado, sem a necessidade de um mouse. Teremos funcionalidades com foco que será acessado através da tecla TAB.
6. Leitura de Tela: Implementar suporte total ou parcial para leitores de tela, permitindo que usuários com deficiência visual acessem e naveguem em nosso site de maneira eficaz.
7. Pesquisa por Texto e Voz: Criação de uma funcionalidade de pesquisa robusta que aceita entrada de texto e voz, proporcionando uma experiência intuitiva para todos os usuários. Essa pesquisa será possível para navegação no site, para realizar busca na opção de busca e também no chatbot.
8. Assistência via chatbot: Introduzir um chatbot inovador que permite aos usuários obter respostas às suas perguntas, pesquisando por texto ou através de comandos de voz. Ele terá a opção de tirar suas dúvidas sobre os produtos e suas características, e falar com um consultor quando desejar.
9. Desenvolver uma funcionalidade de login simplificada, permitindo acesso através das contas do G-mail, LinkedIn, GitHub, Outlook, Apple, e outras plataformas renomadas. O usuário vai poder definir se fará login usando uma plataforma já existente ou se cadastrar direto no site.
10. Implementar formulários de contato mais intuitivos, capazes de serem preenchidos tanto por texto quanto por voz, proporcionando uma experiência inclusiva e eficiente.

11. Utilizar os dados de login simplificado para facilitar o preenchimento dos formulários de contato e para mapear a jornada do visitante pelo site. Isso nos ajuda a compreender suas percepções e necessidades de forma mais precisa.
12. Para os colaboradores, iremos oferecer um painel de consulta ao banco de dados, intuitivo e de fácil utilização. Isso permite que entendam o que o cliente procurou e quanto tempo passou navegando em nosso site. Também fornecemos insights sobre as interações no chat e a atualização de conteúdos, incentivando uma resposta ágil às necessidades dos usuários.
13. Imaginamos um ambiente onde os vídeos possuam legendas incorporadas ou disponíveis separadamente, fornecendo uma descrição abrangente do conteúdo. Isso proporciona a oportunidade para aqueles que não podem assistir, de ler e absorver o conteúdo de forma igualmente eficaz.
14. Os títulos, subtítulos, textos das páginas precisam ser fáceis de entender e intuitivo.
15. A opção de seleção de idioma para a apresentação dos conteúdos é claramente visível, permitindo que qualquer pessoa escolha como prefere consumir informações - seja por leitura ou audição - em seu idioma nativo ou no que está aprendendo.
16. Opção de definição do contraste da tela, seja para um ambiente Dark ou Light.
17. Vídeos sem play automáticos, o usuário quem vai determinar se vai assistir ou não. Caso não queira assistir, vai apenas ler a descrição fora dos vídeos.
18. Descrição em imagens. Todos os usuários precisam entender do que se refere a imagem e precisa trazer significado.
19. Uma página com a documentação de acessibilidade: Será explicado como os recursos de navegação funcionam e como o usuário pode aplicar durante sua experiência.
20. Criação de um canal para suporte e feedback sobre a experiência no site. Os usuários vão poder comunicar problemas existentes durante a navegação ou avaliar as funcionalidades.
21. Navegação por menus mais simples e acessível. Para uso com ou sem teclado.

## Objetivo Específico

---

1. Site mais acessível de todas as formas possíveis e para qualquer pessoa e em qualquer navegador/dispositivo.
2. Criação de um chatbot inteligente e que forneça todo o suporte para o usuário, sem a necessidade de ficar pesquisando no site ou até mesmo falar com uma pessoa para tirar dúvidas simples sobre produtos, características e informações gerais.
3. Aprimorar a Interface para Consulta ao Banco de Dados sem Necessidade de conhecimento em programação ou assistência externa. Isso permitirá que os funcionários obtenham informações de forma autônoma, potencialmente gerando insights valiosos para futuras iniciativas de negócio. Dentro desse contexto, um dos bancos de dados disponíveis conterá a experiência do cliente em todas as interações realizadas no site, abrangendo desde os produtos pesquisados até as dúvidas esclarecidas no chat. Essa riqueza de informações poderá ser utilizada para embasar estratégias de venda de forma mais eficaz.

## **Queremos atuar nas funcionalidades:**

### **Autenticação Simplificada:**

Os usuários podem acessar o site facilmente através de suas contas em plataformas existentes, como G-mail, LinkedIn, GitHub, Outlook e Apple.

### **Formulários de Contato Adaptáveis:**

O site oferece formulários de contato intuitivos que podem ser preenchidos por texto ou voz, proporcionando uma experiência inclusiva e eficiente.

### **Chatbot Inteligente:**

Um chatbot integrado permite aos usuários obter respostas às suas perguntas através de texto ou voz, proporcionando suporte instantâneo e sem a necessidade de interação humana.

### **Leitura de Tela e Legendas:**

O site é completamente compatível com leitores de tela, garantindo que usuários com deficiência visual possam navegar com facilidade. Além disso, os vídeos contêm legendas incorporadas ou disponíveis separadamente para garantir a acessibilidade total.

### **Monitoramento da Jornada do Usuário:**

O site rastreia a jornada do usuário, fornecendo insights sobre o que foi pesquisado, quanto tempo foi gasto navegando e quais conteúdos foram acessados.

### **Seleção de Idioma Personalizada:**

Os usuários têm a opção de escolher o idioma de apresentação dos conteúdos, permitindo que leiam ou ouçam em seu idioma nativo ou em um idioma que estão aprendendo.

### **Acesso para Funcionários:**

Os funcionários têm acesso a um painel intuitivo que permite a consulta ao banco de dados. Eles podem entender o que os clientes procuraram e quanto tempo passaram navegando, além de verificar a atualização de conteúdos e responder rapidamente às necessidades dos usuários.



Estamos buscando não apenas atender aos padrões de acessibilidade, mas também promover uma experiência inclusiva e agradável para todos os usuários. Proporcionando igualdade de acesso à informação e interação na web simples, dinâmica e rápida.

## Escopo do projeto - Java

A linguagem Java será a linguagem principal para o desenvolvimento do back-end do site e implementação de funcionalidades do lado do servidor. Vamos conseguir implementar com essa linguagem a parte de autenticação do login, gerenciamento do banco de dados, desenvolvimento dos formulários para que possam ser preenchidos por texto ou voz, leitura da tela, e parte de segurança. Além da criação de uma interface para o funcionário consultar o banco de dados onde vai constar a experiência do segurado, e com isso vai gerar novas oportunidades de negócio.

## Design do Diagrama de classe

### Sprint 1

Primeiro diagrama de Classe:

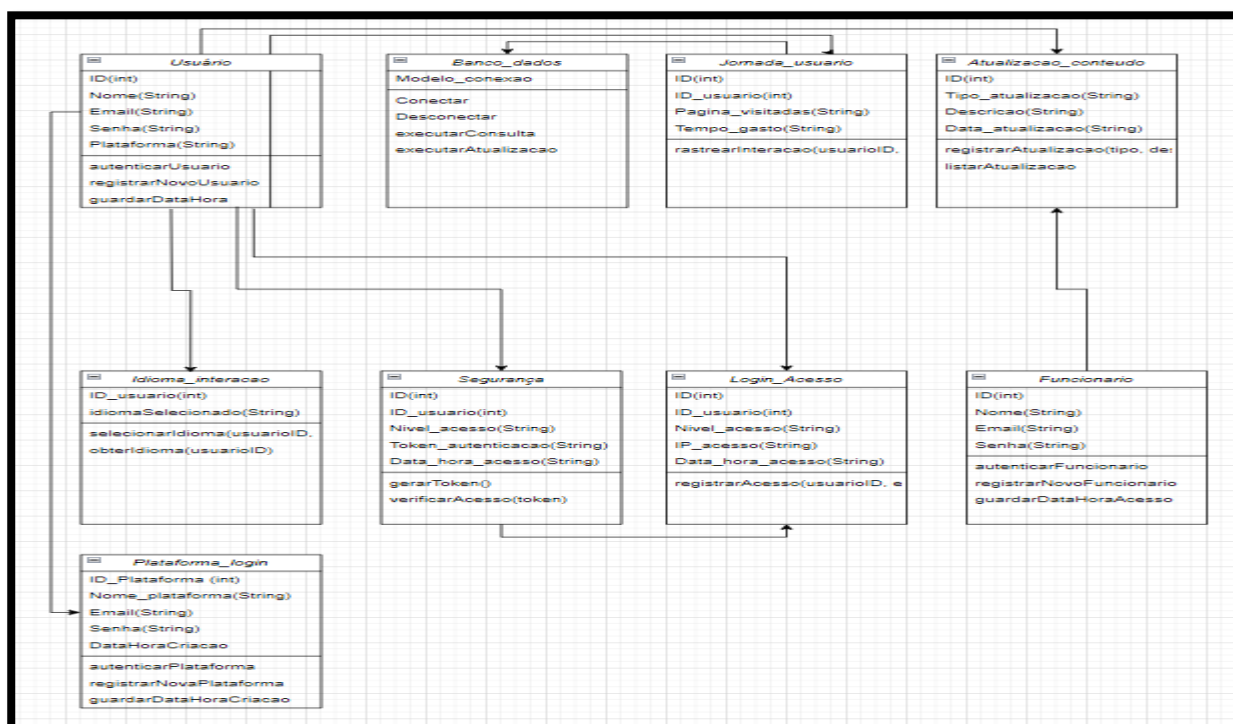
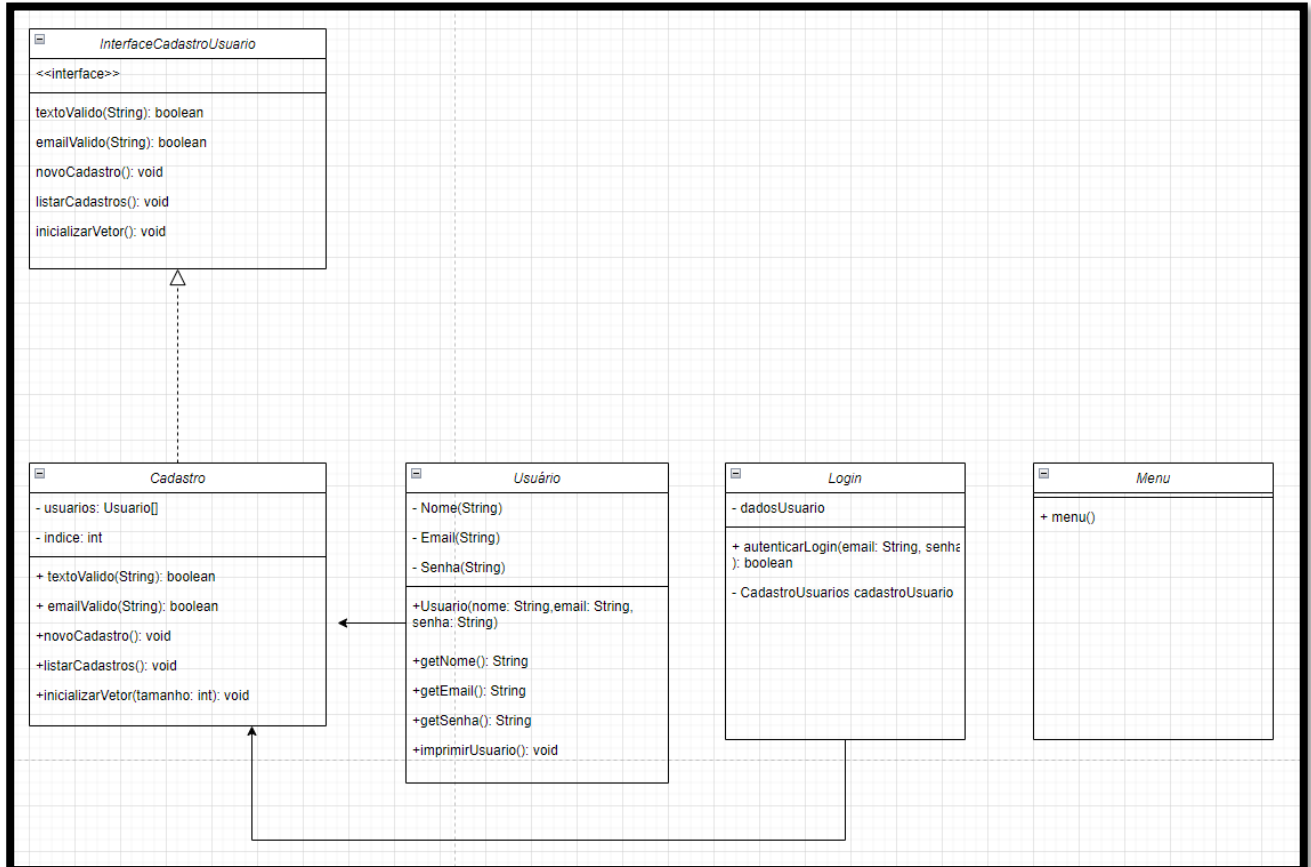


Diagrama ajustado para segunda entrega com foco no Cadastro, Usuário e Login. As demais deixamos no modelo, mas não vamos ajustar neste momento.



## Entidades

1. Usuario
2. CadastroUsuario
3. Login
4. Menu
5. Banco de dados: Gerenciamento do banco de dados para consultas
6. Jornada do usuário: Monitorar a jornada do usuário durante a navegação do site
7. Atualização do conteúdo ou sinalização de melhorias
8. Idioma de interação: Definição do idioma de interação no site e nos bancos de dados
9. Segurança: Desde autenticação do acesso, até o controle de visualização de conteúdo.
10. Funcionário

# Atributos e métodos

---

## 1. Usuário:

**Entidade:** Usuário

**Atributos:**

- Nome
- E-mail
- Senha

**Métodos:**

- imprimirUsuario()

**Getters e Setters:**

- getNome(): String
- setNome(String)
- getEmail(): String
- setEmail(String)
- getSenha(): String
- setSenha(String)

## 2. CadastroUsuario:

**Entidade:** Cadastro

**Atributos:**

- Usuario[]
- Indice

**Métodos:**

- textoValido(String): boolean
- emailValido(String): boolean
- novoCadastro(): Void
- listarCadastros(): void
- inicializarVetor(): void

### 3. Login

**Entidade:** Login

**Atributos:**

- DadosUsuario

**Métodos:**

- autenticarLogin(String nome, String email, String senha);
- CadastroUsuario cadastroUsuario()

### 4. Gerenciamento de Banco de Dados:

**Entidade:** Banco de Dados

**Atributos:**

- Conexão (fazer a conexão com o banco de dados – Aqui podemos usar o python para auxiliar no processo de API por exemplo)

**Métodos:**

- conectar(): Estabelece uma conexão com o banco de dados.
- desconectar(): Fecha a conexão com o banco de dados.
- executarConsulta(query): Executa uma consulta SQL no banco de dados.
- executarAtualizacao(query): Executa uma atualização no banco de dados (inserção, atualização, exclusão). Aqui depende do nível de acesso.

### 5. Monitoramento da Jornada do Usuário:

**Entidade:** Jornada do Usuário

**Atributos:**

- ID (Inteiro)
- ID do Usuário (Inteiro)
- Páginas Visitadas (Strings)
- Tempo Gasto (String)

**Métodos:**

- rastrearInteracao(usuarioID, paginaVisitada, tempoGasto): Registra a interação do usuário com o site.

## 6. Atualização do Conteúdo ou Sinalização de Melhorias:

**Entidade:** Atualizacao\_conteudo

**Atributos:**

- ID (Inteiro)
- Tipo de Atualização (String, por exemplo: "Novo Conteúdo", "Atualização de Informações", "Melhoria de Acessibilidade")
- Descrição (String)
- Data da Atualização (Data e Hora)

**Métodos:**

- registrarAtualizacao(tipo, descricao): Registra uma nova atualização de conteúdo no sistema.
- listarAtualizacoes(): Lista todas as atualizações registradas.

## 7. Idioma de Interação:

**Entidade:** IdiomaInteracao

**Atributos:**

- ID do Usuário (Inteiro)
- Idioma Selecionado (String, código do idioma, por exemplo: "pt-br", "en-us")

**Métodos:**

- selecionarIdioma(usuarioID, idioma): Permite ao usuário selecionar o idioma de interação preferencial.
- obterIdioma(usuarioID): Retorna o idioma selecionado para um usuário específico.

## 8. Segurança

Para parte de segurança da aplicação:

**Entidade:** Seguranca

**Atributos:**

- ID (Inteiro)
- ID do Usuário (Inteiro)
- Nível de Acesso (String)
- Token de Autenticação (String)
- Último Acesso (String)

**Métodos:**

- gerarToken(): Gera um novo token de autenticação para o usuário.
- verificarAcesso(token): Verifica se o token fornecido é válido e corresponde a um usuário autenticado.

## 9. Funcionário

**Entidade:** Funcionário

**Atributos:**

- ID (Inteiro)
- Nome (String)
- E-mail (String)
- Senha (String)

**Métodos:**

- autenticarFuncionario
- registrarNovoFuncionario
- guardarDataHoraAcesso

## 10. Plataforma

**Entidade:** Plataforma

**Atributos:**

- ID (Inteiro)
- nomePlataforma (String)
- email (String)
- senha (String)
- dataHoraCriacao (String)

**Métodos:**

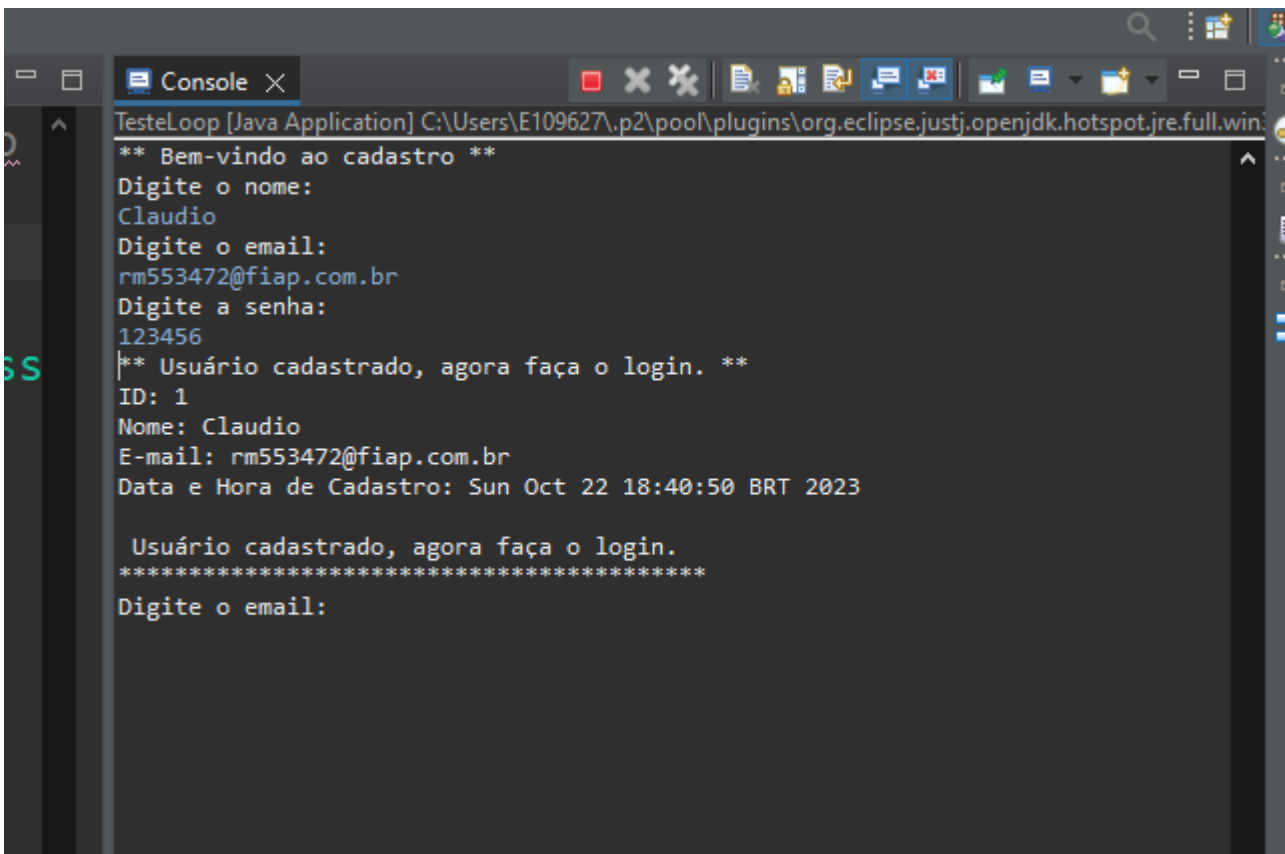
- autenticarPlataforma
- registrarNovaPlataforma
- guardarDataHoraCriacao

# Primeira entrega

---

## Cadastrar o usuário

Tela



```
TesteLoop [Java Application] C:\Users\E109627\p2\pool\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win:
** Bem-vindo ao cadastro **
Digite o nome:
Claudio
Digite o email:
rm553472@fiap.com.br
Digite a senha:
123456
** Usuário cadastrado, agora faça o login. **
ID: 1
Nome: Claudio
E-mail: rm553472@fiap.com.br
Data e Hora de Cadastro: Sun Oct 22 18:40:50 BRT 2023

  Usuário cadastrado, agora faça o login.
*****
Digite o email:
```

Código

```
package projetoChallenge;

import java.util.Scanner;
import java.util.Date;

public class Usuario {
    // Atributos
    private static int proximoID = 1;
    private int ID;
    private String nome;
    private String email;
```

```
private String senha;
private Date dataHoraCadastro;

// Construtor
public Usuario(String nome, String email, String senha) {
    this.ID = proximoID++;
    this.nome = nome;
    this.email = email;
    this.senha = senha;
}

// Getters e Setters
public void setDataHoraCadastro(Date dataHoraCadastro) {
    this.dataHoraCadastro = dataHoraCadastro;
}

public static int getProximoID() {
    return proximoID;
}

public static void setProximoID(int proximoID) {
    Usuario.proximoID = proximoID;
}

public int getID() {
    return ID;
}

public void setID(int iD) {
    ID = iD;
}

public String getNome() {
    return nome;
}

public void setNome(String nome) {
    this.nome = nome;
}

public String getEmail() {
    return email;
}

public void setEmail(String email) {
    this.email = email;
}
```



```
public String getSenha() {
    return senha;
}

public void setSenha(String senha) {
    this.senha = senha;
}

public static ArrayList<Usuario> getUsuarios() {
    return usuarios;
}

public static void setUsuarios(ArrayList<Usuario> usuarios) {
    Usuario.usuarios = usuarios;
}

public Date getDataHoraCadastro() {
    return dataHoraCadastro;
}

// Método
public void registrarNovoUsuario() {
    Scanner scanner = new Scanner(System.in);

    //System.out.println("Digite o ID: ");
    //int id = scanner.nextInt();
    //scanner.nextLine();

    System.out.println("Digite o nome: ");
    String nome = scanner.nextLine();

    System.out.println("Digite o email: ");
    String email = scanner.nextLine();

    System.out.println("Digite a senha: ");
    String senha = scanner.nextLine();

    Date dataHoraCadastro = new Date();

    //this.ID = id;
    this.nome = nome;
    this.email = email;
    this.senha = senha;
    this.dataHoraCadastro = dataHoraCadastro;

    Usuario novoUsuario = new Usuario(nome, email, senha);
    novoUsuario.setDataHoraCadastro(dataHoraCadastro);
}
```

```
usuarios.add(novoUsuario);

System.out.println("** Usuário cadastrado, agora faça o login. **");

//scanner.close();
}

public void imprimirUsuario() {
    System.out.println("ID: " + this.ID);
    System.out.println("Nome: " + this.nome);
    System.out.println("E-mail: " + this.email);
    System.out.println("Data e Hora de Cadastro: " + this.dataHoraCadastro);
}

public boolean autenticarUsuario(String email, String senha) {
    return this.email.equals(email) && this.senha.equals(senha);
}

public class PrincipalTeste {
    public static void main(String[] args) {

        System.out.println("** Bem-vindo ao cadastro **");

        Usuario novoUsuario = new Usuario(null, null, null);
        novoUsuario.registrarNovoUsuario();

        // Imprimir os dados na tela
        novoUsuario.imprimirUsuario();

        System.out.println("\n Usuário cadastrado, agora faça o login. ");
    }
}
```

## Login

### Tela

- Com autenticação positiva

```
*****
Digite o email:
rm553472@fiap.com.br
Digite a senha:
123456
Usuário autenticado com sucesso! Login realizado com sucesso
*****
```

- Com autenticação negativa de acesso

```
S
** Bem-vindo ao cadastro **
Digite o nome:
Claudio
Digite o email:
rm553472@fiap.com.br
Digite a senha:
123456
** Usuário cadastrado, agora faça o login. **
ID: 3
Nome: Claudio
E-mail: rm553472@fiap.com.br
Data e Hora de Cadastro: Sun Oct 22 18:49:37 BRT 2023

Usuário cadastrado, agora faça o login.
*****
Digite o email:
1@teste.com.br
Digite a senha:
1234567
Credenciais inválidas. Tente novamente.
*****
```

## Código

```
package projetoChallenge;

import java.util.Scanner;

public class Login {
    // Atributos
    private String email;
    private String senha;

    // Construtor
    public Login(String email, String senha) {
        this.email = email;
        this.senha = senha;
    }

    // Getters e Setters
```

```
public String getEmail() {
    return email;
}

public void setEmail(String email) {
    this.email = email;
}

public String getSenha() {
    return senha;
}

public void setSenha(String senha) {
    this.senha = senha;
}

// Método
public void acessar() {
    Scanner scanner = new Scanner(System.in);

    System.out.println("Digite o email: ");
    this.email = scanner.nextLine();

    System.out.println("Digite a senha: ");
    this.senha = scanner.nextLine();

    //System.out.println("Login realizado com sucesso!");

    //scanner.close();
}

public boolean autenticar() {
    // Chama o método autenticarUsuario da classe Usuario
    Usuario usuario = new Usuario(null, null, null);
    return usuario.autenticarUsuario(email, senha);
}

public void imprimirLogin() {
    System.out.println("E-mail: " + this.email);
    System.out.println("Senha: " + this.senha);
}

public class PrincipalTeste {
    public static void main(String[] args) {
```

```
        System.out.println("** Bem-vindo ao Login **");

        Login login = new Login(null, null);
        login.acessar();

    }

}
```

## Autenticação

### Código

```
package projeto_challenge;

public class Autenticador {
    public static boolean autenticar(Login login) {
        String email = login.getEmail();
        String senha = login.getSenha();

        if (email != null && senha != null) {
            for (Usuario usuario : Usuario.usuarios) {

                if (usuario.autenticarUsuario(email, senha)) {
                    return true;
                }
            }
        }
        return false;
    }
}
```

## Definição do idioma

Tela

```
*****
Selecione o idioma desejado:[0] Português
[1] Inglês
[2] Espanhol
[3] Francês
[4] Alemão

2
Idioma Espanhol selecionado!
Idioma ativo é: Espanhol
Deseja continuar? (s/n)
```

Código

```
package projetoChallenge;

import java.util.Scanner;

public class IdiomaInteracao {
    // Atributos
    private int ID_usuario;
    private String idiomaSelecionado;

    // Construtor
    public IdiomaInteracao(int ID_usuario, String idiomaSelecionado) {
        this.ID_usuario = ID_usuario;
        this.idiomaSelecionado = idiomaSelecionado;
    }

    // Getters and Setters
    public String getIdiomaSelecionado() {
        return idiomaSelecionado;
    }

    public void setIdiomaSelecionado(String idiomaSelecionado) {
        this.idiomaSelecionado = idiomaSelecionado;
    }

    public int getID_usuario() {
        return ID_usuario;
    }
}
```

```
public void setID_usuario(int ID_usuario) {
    this.ID_usuario = ID_usuario;
}

// Método
public void selecionarIdioma() {
    System.out.println("Selecione o idioma desejado:"
        + "[0] Português \n"
        + "[1] Inglês \n"
        + "[2] Espanhol \n"
        + "[3] Francês \n"
        + "[4] Alemão \n");

    Scanner scanner = new Scanner(System.in);

    int opcao = scanner.nextInt();

    if(opcao == 0 ) {
        this.idiomaSelecionado = "Português";
        System.out.println("Idioma Português selecionado! ");
    }else if(opcao == 1) {
        this.idiomaSelecionado = "Inglês";
        System.out.println("Idioma Inglês selecionado! ");
    }else if(opcao == 2) {
        this.idiomaSelecionado = "Espanhol";
        System.out.println("Idioma Espanhol selecionado! ");
    }else if(opcao == 3) {
        this.idiomaSelecionado = "Espanhol";
        System.out.println("Idioma Francês selecionado! ");
    }else if(opcao == 4) {
        this.idiomaSelecionado = "Espanhol";
        System.out.println("Idioma Alemão selecionado! ");
    }else {
        System.out.println("Opção inválida! ");
    }
}

// Estou testando se funcionou a seleção do idioma
public class IdiomaTeste {
    public static void main(String[] args) {

        // Usuário vai selecionar o idioma desejado.
        IdiomaInteracao idioma = new IdiomaInteracao(0, null);
        idioma.selecionarIdioma();

        // Preciso saber qual o idioma está ativo na tela do usuário.
        System.out.println("Idioma ativo é: " + idioma.getIdiomaSelecionado());
    }
}
```

```
}  
}  
}
```

## Main

Neste script consta um consolidado de todas as funcionalidades desenvolvidas até o momento. Serão etapas separadas mas no modelo abaixo, deixamos uma mostra de como será todo o processo.



```

TesteLoop [Java Application] C:\Users\E109627\p2\pool\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win
123456
** Usuário cadastrado, agora faça o login. **
ID: 1
Nome: Claudio
E-mail: rm553472@fiap.com.br
Data e Hora de Cadastro: Sun Oct 22 18:40:50 BRT 2023

  Usuário cadastrado, agora faça o login.
  *****
  Digite o email:
  rm553472@fiap.com.br
  Digite a senha:
  123456
  Usuário autenticado com sucesso! Login realizado com sucesso
  *****
  Selecione o idioma desejado:[0] Português
  [1] Inglês
  [2] Espanhol
  [3] Francês
  [4] Alemão

  1
  Idioma Inglês selecionado!
  Idioma ativo é: Inglês
  Deseja continuar? (s/n)
  s
  ** Bem-vindo ao cadastro **
  Digite o nome:
  Claudio
  Digite o email:
  rm553472@fiap.com.br
  Digite a senha:
  123456
  ** Usuário cadastrado, agora faça o login. **
  ID: 3
  Nome: Claudio
  E-mail: rm553472@fiap.com.br
  Data e Hora de Cadastro: Sun Oct 22 18:49:37 BRT 2023

  Usuário cadastrado, agora faça o login.
  *****
  Digite o email:
  1@teste.com.br
  Digite a senha:
  1234567
  Credenciais inválidas. Tente novamente.
  *****
  Selecione o idioma desejado:[0] Português
  [1] Inglês
  [2] Espanhol
  [3] Francês
  [4] Alemão

  2
  Idioma Espanhol selecionado!
  Idioma ativo é: Espanhol
  Deseja continuar? (s/n)
  
```

## Código

```

package projeto_challenge;

import java.util.Scanner;

public class TesteLoop {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);

        while (true) {
    
```

```
System.out.println("** Bem-vindo ao cadastro **");

// Primeiro preciso realizar o cadastro do usuário
Usuario novoUsuario = new Usuario(null, null, null);
novoUsuario.registrarNovoUsuario();

// Imprimir os dados na tela
novoUsuario.imprimirUsuario();

System.out.println("\n Usuário cadastrado, agora faça o login. ");

System.out.println("*****");

// Segundo pedir credenciais de login
Login login = new Login(null, null);
login.acessar();

// Terceiro preciso Autenticar usuário/Login com a base de cadastro e
fazer ou não o acesso.
boolean autenticado = Autenticador.autenticar(login);

if (autenticado) {
    System.out.println("Usuário autenticado com sucesso! Login
realizado com sucesso");
} else {
    System.out.println("Credenciais inválidas. Tente novamente.");
}

System.out.println("*****");

// Usuário vai selecionar o idioma desejado.
IdiomaInteracao idioma = new IdiomaInteracao(0, null);
idioma.selecionarIdioma();

// Preciso saber qual o idioma está ativo na tela do usuário.
System.out.println("Idioma ativo é: " + idioma.getIdiomaSelecionado());

// Perguntar se deseja continuar
System.out.println("Deseja continuar? (s/n)");
String continuar = scanner.nextLine();

if (!continuar.equalsIgnoreCase("s")) {
    break; // Se a resposta for diferente de "s", o loop é encerrado
}
}

// Fechar o scanner no final
scanner.close();
```

```
}  
}
```

**Importante:**

**Foi criado outros fluxos com atributos, Construtor, Getters, Setters e métodos que gostaríamos. Não possui as funcionalidades dentro pois ainda vamos aprender.**

## Segunda entrega

---

### Objetivo e atividades realizadas

Revisitamos e otimizamos nosso código, incorporando conceitos avançados discutidos nas últimas aulas. Introduzimos estruturas de dados mais eficientes, como vetores e matrizes, para armazenar os dados de cadastro. Essa abordagem não apenas aprimora a organização dos dados, mas também proporciona oportunidades adicionais, como facilitar o processo de login, autenticação e possibilitar a criação de mapas da experiência do cliente com base nos dados cadastrados. Essas melhorias contribuem para uma implementação mais robusta e versátil do nosso sistema de cadastro e login, elevando a qualidade e a capacidade funcional da aplicação.

Neste novo modelo vamos conseguir definir padrões para nomes, senhas, e-mails e um armazenamento mais estruturado, além de começar o direcionamento para aplicarmos a herança e Interface. Depois de criarmos a classe de usuário, a partir dele teremos as classes que vão herdar os dados básicos como nome, e-mail e senha, e serão adicionadas as suas especializações como usuário admin por exemplo, onde terá um nível de acesso diferente e outros atributos.

### Fluxo no terminal

Exemplo do sistema funcionando no terminal. Vai ser demonstrado algumas etapas e o que ocorre durante o processo.

1. Primeiro vai mostrar um menu onde vai direcionar o usuário durante sua navegação

```
TestePrincipal [Java Application] [pid: 14952]
Menu inicial - Selecione uma opção digitando um número da lista abaixo
0 - Retornar ao menu principal
1 - Cadastrar
2 - Listar cadastro
3 - Realizar o login
4 - Sair
Digite 4 a qualquer momento para sair/encerrar o programa ou 0 para retornar ao menu anterior.
```

2. A qualquer momento que o usuário digitar 0, ele retornará para o menu principal.

```
TestePrincipal [Java Application] [pid: 14952]
4 - Sair
Digite 4 a qualquer momento para sair/encerrar o programa ou 0 para retornar ao menu anterior.
0
Retornando ao menu principal...

Menu inicial - Selecione uma opção digitando um número da lista abaixo
0 - Retornar ao menu principal
1 - Cadastrar
2 - Listar cadastro
3 - Realizar o login
4 - Sair
Digite 4 a qualquer momento para sair/encerrar o programa ou 0 para retornar ao menu anterior.
```

3. Ao digitar 1, o usuário vai ser direcionado para o cadastro:

```
TestePrincipal [Java Application] [pid: 13588]
Menu inicial - Selecione uma opção digitando um número da lista abaixo
0 - Retornar ao menu principal
1 - Cadastrar
2 - Listar cadastro
3 - Realizar o login
4 - Sair
Digite 4 a qualquer momento para sair/encerrar o programa ou 0 para retornar ao menu anterior.
1
Digite o nome:
```

Ele vai precisar preencher o:

- nome,
- E-mail

- Senha

Dentro deste processo temos validações caso o usuário se perca ou digite algo errado.

Por exemplo se ele digitar um nome que não contenham strings/letras.

```
1
Realize seu cadastro
Digite o nome:
55
Nome inválido. Por favor, insira um texto válido.
```

Ou digitar um e-mail fora dos padrões, sem o @ por exemplo:

```
1
Realize seu cadastro
Digite o nome:
Claudio
Digite o email:
rm553425
Email inválido. Por favor, insira um e-mail válido.
```

No final, até a senha precisar ter letras ou dígitos.

Caso ele não digite nada, o sistema vai ficar pedindo informações e não sairá do loop até que ele digite algo válido ou 0 para retornar ao menu principal.

```
1
Digite o nome:
Claudio
Nome válido.

Digite o email:

Email inválido. Por favor, insira um e-mail válido.
Digite o email:

Email inválido. Por favor, insira um e-mail válido.
Digite o email:

Email inválido. Por favor, insira um e-mail válido.
Digite o email:
```

4. A segunda opção do menu é listar as constas cadastradas. Se não houver dados, ele vai inserir na tela uma mensagem de erro e pedindo vai voltar ao menu.

```

2
Não há cadastros para listar.

Menu inicial - Selecione uma opção indicando um número
1 - Cadastrar
2 - Listar cadastro
3 - Alterar senha
4 - Alterar senha
5 - Sair
  
```

```

Console X
TestePrincipal [Java Application] [pid: 13588]
3 - Realizar o login
4 - Sair
Digite 4 a qualquer momento para sair/encerrar o programa ou 0 para retornar ao menu anterior.
2
Cadastros realizados até o momento:

Nome: claudio
Email: rm553472@fiap.com.br
Senha: cs123456
*****
Menu inicial - Selecione uma opção digitando um número da lista abaixo
0 - Retornar ao menu principal |
1 - Cadastrar
2 - Listar cadastro
3 - Realizar o login
4 - Sair
Digite 4 a qualquer momento para sair/encerrar o programa ou 0 para retornar ao menu anterior.
<
  
```

5. Já a opção 3 é para fazer o login, ele vai pedir e-mail e senha que já contenham na listagem de cadastro, se não tiver, ele não vai validar o login e informar que ele precisa se cadastrar.

```

Console X
TestePrincipal [Java Application] [pid: 13588]
Digite 4 a qualquer momento para sair/encerrar o programa ou 0 para retornar ao menu anterior.
2
Cadastros realizados até o momento:

Nome: claudio
Email: rm553472@fiap.com.br
Senha: cs123456
*****
Menu inicial - Selecione uma opção digitando um número da lista abaixo
0 - Retornar ao menu principal
1 - Cadastrar
2 - Listar cadastro
3 - Realizar o login
4 - Sair
Digite 4 a qualquer momento para sair/encerrar o programa ou 0 para retornar ao menu anterior.
3
Digite seu email:
<
  
```

Na validação abaixo, ele mostra que os dados estão incorretos.

```
3
Digite seu email:
rm553472@fiap.com.br
Digite a senha cadastrada:
cs123
Credenciais incorretas. Tente novamente.
```

Já na listagem mostra que temos um cadastro e que se a pessoa inserir os mesmos dados, vai conseguir logar.

```
2
Cadastros realizados até o momento:

Nome: claudio
Email: rm553472@fiap.com.br
Senha: cs123
*****
```

```
2
Digite 7 a qualquer momento para sair/encerrar o programa ou 0 para retornar ao menu anterior.
Cadastros realizados até o momento:

Nome: claudio
Email: rm553472@fiap.com.br
Senha: cs123
*****
Menu inicial - Selecione uma opção digitando um número da lista abaixo
0 - Retornar ao menu principal
1 - Cadastrar
2 - Listar cadastro
3 - Realizar o login
```

6. O menu terá as opções 5,6 e 7 que será desenvolvida na próxima Sprint. Onde contemplaremos alteração de senha, alteração de dados pessoais e implementação da alteração do idioma com melhorias.

7. A opção 4 – Sair vai encerrar o programa:

```
<
Console X
<terminated> TestePrincipal [Java Application] C:\Users\E109627\p2\pool\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86_64_17.0.8
Menu inicial - Selecione uma opção digitando um número da lista abaixo
0 - Retornar ao menu principal
1 - Cadastrar
2 - Listar cadastro
3 - Realizar o login
4 - Sair
Digite 4 a qualquer momento para sair/encerrar o programa ou 0 para retornar ao menu anterior.
4
Saindo do programa. Até logo!
```

## Código completo

Interface para o cadastro do usuário

```
package projetoChallenge;

/*Classe CadastroUsuario:
 * Seu propósito é armazenar os usuários cadastrados e
 * fornecer métodos para manipular esse conjunto de usuários.
 * É responsável por operações relacionadas ao cadastro de usuários,
 * como adicionar um novo usuário à lista de usuários.
 */
public interface InterfaceCadastroUsuario {

    // Métodos que será utilizados na classe Cadastro.
    public void novoCadastro();
    public void listarCadastros();
    public void inicializarVetor(int tamanho);
    public boolean textoValido(String email);
    public boolean emailValido(String email);

}
```

Classe: Cadastro do usuário

```
package projetoChallenge;

import java.util.Scanner;

public class CadastroUsuario implements InterfaceCadastroUsuario {

    private Usuario[] usuarios;
    private int indice;

    public CadastroUsuario(int tamanho) {
        usuarios = new Usuario[tamanho];
        indice = 0;
    }
}
```



```
}

@Override
public boolean textoValido(String texto) {
    return texto.matches("[a-zA-Z]+");
}

@Override
public boolean emailValido(String email) {
    String regex = "^[a-zA-Z0-9._%+-]+@[a-zA-Z0-9.-]+\\.[a-zA-Z]{2,}$";
    return email.matches(regex);
}

@Override
public void novoCadastro() {
    Scanner input = new Scanner(System.in);

    String nome = null;
    String email = null;
    String senha = null;

    while (true) {
        // Coletar o nome
        System.out.println("Digite o nome: ");
        nome = input.nextLine();

        // Verificar se o usuário deseja voltar ao menu principal
        if ("0".equals(nome)) {
            System.out.println("Retornando ao menu principal...");
            return;
        }

        // Condição para validar o nome
        if (!textoValido(nome)) {
            System.out.println("Nome inválido. Por favor, insira um nome válido.\n");
        } else {
            System.out.println("Nome válido. \n");
            break; // Nome é válido, sair do loop
        }
    }

    while (true) {
        // Coletar o email
        System.out.println("Digite o email: ");
        email = input.nextLine();
    }
}
```

```
// Verificar se o usuário deseja voltar ao menu principal
if ("0".equals(email)) {
    System.out.println("Retornando ao menu principal...");
    return;
}

// Processo para validar formato do email.
if (!emailValido(email)) {
    System.out.println("Email inválido. Por favor, insira um email válido.");
} else {
    System.out.println("Email válido. \n");
    break; // Email é válido, sair do loop
}

while (true) {
    // Coletar a senha
    System.out.println("Digite a senha que contenha letras e números: ");
    senha = input.nextLine();

    // Verificar se o usuário deseja voltar ao menu principal
    if ("0".equals(senha)) {
        System.out.println("Retornando ao menu principal...");
        return;
    }

    if (!senha.matches("^(?=.*[a-zA-Z])(?=.*[0-9])[a-zA-Z0-9]+$")) {
        System.out.println("Senha inválida. Por favor, insira uma senha com pelo menos uma letra e um número.");
    } else {
        System.out.println("Senha válida. \n");
        break; // Senha é válida, sair do loop
    }
}

// Criar um novo usuário com os dados coletados
Usuario novoUsuario = new Usuario(nome, email, senha);

// Adicionar o novo usuário ao vetor
usuarios[indice] = novoUsuario;

indice++;
System.out.println("Cadastro realizado com sucesso!\n");
}
```

```
@Override
public void listarCadastros() {

    if (indice == 0) {
        System.out.println("Não há cadastros para listar.\n");
        return;
    }
    System.out.println("Cadastros realizados até o momento:\n");
    for (int i = 0; i < indice; i++) {
        System.out.println("Nome: " + usuarios[i].getNome());
        System.out.println("Email: " + usuarios[i].getEmail());
        System.out.println("Senha: " + usuarios[i].getSenha());
        System.out.println("*****");
    }
}

@Override
public void inicializarVetor(int tamanho) {
    usuarios = new Usuario[tamanho];
    indice = 0;
}

public Usuario[] getUsuarios() {
    return usuarios;
}
}
```

Classe: Usuário

```
package projetoChallenge;

public class Usuario {
    // Atributos

    private String nome;
    private String email;
    private String senha;

    // Construtor
    public Usuario(String nome, String email, String senha) {
        this.nome = nome;
        this.email = email;
    }
}
```

```
        this.senha = senha;

    }

    // Getters e Setters

    public String getSenha() {
        return senha;
    }

    public void setSenha(String senha) {
        this.senha = senha;
    }

    public String getNome() {
        return nome;
    }

    public void setNome(String nome) {
        this.nome = nome;
    }

    public String getEmail() {
        return email;
    }

    public void setEmail(String email) {
        this.email = email;
    }

    public void imprimirUsuario() {
        System.out.println("Nome: " + this.getNome());
        System.out.println("E-mail: " + this.getEmail());
        System.out.println("Senha: " + this.getSenha());
    }

}

}
```

Classe: Login

```
package projeto_challenge;

import java.util.Scanner;
```

```
public class Login {
    // Uso de composição entre as classes
    private CadastroUsuario cadastroUsuario;

    // Construtor
    public Login(CadastroUsuario cadastroUsuario) {
        this.cadastroUsuario = cadastroUsuario;
    }

    public boolean autenticarLogin() {
        Scanner scanner = new Scanner(System.in);

        String emailInput = "";
        String senhaInput = "";

        while (true) {
            // Coletar o email
            System.out.println("Digite seu email: ");
            emailInput = scanner.nextLine();

            if ("0".equals(emailInput)) {
                System.out.println("Saindo do processo de login...");
                return false;
            } else if (emailInput.isEmpty()) {
                System.out.println("Email não pode estar vazio. Por favor, insira um email válido.\n");
            } else {
                break; // Se o Email for válido, pode sair do loop
            }
        }

        while (true) {
            // Coletar a senha
            System.out.println("Digite a senha cadastrada: ");
            senhaInput = scanner.nextLine();

            if ("0".equals(senhaInput)) {
                System.out.println("Saindo do processo de login...");
                return false;
            } else if (senhaInput.isEmpty()) {
                System.out.println("Senha não pode estar vazia. Por favor, insira uma senha válida.\n");
            } else {
                break; // Senha válida, sair do loop
            }
        }
    }
}
```

```
// Verificar as credenciais
for (Usuario usuario : cadastroUsuario.getUsuarios()) {
    if (usuario != null && usuario.getEmail().equals(emailInput) &&
usuario.getSenha().equals(senhaInput)) {
        System.out.println("Login bem-sucedido!");
        return true; // Credenciais corretas
    }
}

System.out.println("Credenciais incorretas. Tente novamente.\n");
return false; // Credenciais incorretas
}
}
```

Classe: Menu

```
package projetoChallenge;

import java.util.Scanner;

public class Menu {
    public int menu() {
        Scanner input = new Scanner(System.in);

        System.out.println(" Menu inicial - Selecione uma opção digitando um número da
lista abaixo");
        System.out.println(" 0 - Retornar ao menu principal ");
        System.out.println(" 1 - Cadastrar ");
        System.out.println(" 2 - Listar cadastro ");
        System.out.println(" 3 - Realizar o login ");
        System.out.println(" 4 - Sair");
        System.out.println(" Digite 4 a qualquer momento para sair/encerrar o programa
ou 0 para retornar ao menu anterior.");

        int opcao = input.nextInt();

        return opcao;
    }
}
```

Main – Processo principal

```
package projetoChallenge;
```

```
import java.util.Scanner;

public class TestePrincipal {

    //public Usuario[] cadastros;

    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);

        // Define o tamanho do vetor de cadastros
        int tamanhoCadastro = 10;

        //Usuario dadosUsuario = new Usuario(null, null, null);

        CadastroUsuario dadosUsuario = new CadastroUsuario(tamanhoCadastro);

        dadosUsuario.inicializarVetor(tamanhoCadastro);

        // Criar instância de Login passando DadosUsuario
        Login login = new Login(dadosUsuario);
        Menu menu = new Menu();

        while (true) {
            int opcao;
            do {
                // Exibe menu inicial
                opcao = menu.menu();

                switch (opcao) {
                    case 0:
                        // Opção de retornar ao menu principal
                        System.out.println("Retornando ao menu principal...\n");
                        break;

                    case 1:
                        // Opção de cadastro
                        dadosUsuario.novoCadastro();
                        break;

                    case 2:
                        // Opção de listar cadastros
                        dadosUsuario.listarCadastros();
                        break;

                    case 3:
                        // Opção de realizar login
                        boolean autenticado = login.autenticarLogin();
                        break;
                }
            } while (opcao != 0);
        }
    }
}
```

```
        case 4:
            // Opção de sair
            System.out.println("Saindo do programa. Até logo!\n");
            System.exit(0);
        default:
            System.out.println("Opção inválida. Tente novamente.\n");
            break;
    }

    } while (opcao != 0);
}
}
```

## Conclusão

---

Nossa dedicação à acessibilidade é contínua e vai além. Continuaremos a aprimorar nossa proposta de site e a explorar novas formas de tornar a web um ambiente verdadeiramente inclusivo, em total alinhamento com os objetivos da nossa empresa.

A implementação de um chatbot ainda mais avançado reforça nossa abordagem inovadora. Ao empregar tecnologias de processamento de linguagem natural de última geração, nosso chatbot proporciona respostas precisas e relevantes em tempo real, oferecendo suporte eficaz e personalizado aos visitantes do site. Com essa integração, as empresas desfrutam de ganhos significativos. Têm a capacidade de criar um ambiente digital que promove inclusão, aprimorar a eficiência operacional e proporcionar experiências excepcionais aos clientes.

Além disso, a combinação de um site acessível e um chatbot tecnologicamente avançado simplifica a captação de leads, tornando o processo mais intuitivo e eficaz. A interação personalizada e a facilidade de acesso contribuem para uma maior geração de leads qualificados, impulsionando o crescimento e a competitividade no mercado.

Ao alinharmos nossa visão com os princípios da Salesforce, Sociedade 5.0 e 4.0, não apenas incorporamos tecnologias avançadas como IA e automação, mas também capacitamos as empresas/clientes a proporcionarem experiências mais ricas e personalizadas aos seus clientes e funcionários. Ao criar um site extremamente acessível, reforçamos nosso compromisso com a inclusão, tornando a experiência online acessível a todos, independentemente de suas necessidades e habilidades.