

**Guilherme Carvalho Siqueira Santos**

**Victor Hugo Francisco do Nascimento**

**Kauã Henrique da Silva Araujo**

**Leandro Bergamini dos Santos Júnior**

**TECHVAGAS**

Jales

2024

# SUMÁRIO

1. [INTRODUÇÃO 3](#_bookmark0)
2. [LEVANTAMENTO DE REQUISITOS DE SOFTWARE 4](#_bookmark1)
   1. [Descrição dos objetivos do sistema 4](#_bookmark2)
   2. [Descrição do sistema atual 4](#_bookmark3)
   3. [Descrição dos principais problemas 4](#_bookmark4)
   4. [Descrição dos requisitos funcionais 4](#_bookmark5)
   5. [Descrição dos requisitos não funcionais 4](#_bookmark6)
3. [VISÃO DE CASO DE USO – UML 5](#_bookmark7)
   1. [Diagrama de Classes 5](#_bookmark8)
   2. [Dicionário de classes 5](#_bookmark9)
   3. [Definição dos Atores 5](#_bookmark10)
   4. [Lista de Casos de Uso 5](#_bookmark11)
   5. [Diagrama de Casos de Uso 5](#_bookmark12)
   6. [Diagrama de Casos de uso individuais 6](#_bookmark13)
   7. [Diagrama de Sequência 7](#_bookmark14)
   8. [Diagrama de Comunicação 7](#_bookmark15)
   9. [Diagrama de Atividade 7](#_bookmark16)

[3.9 Diagrama de Máquina de Estados 7](#_bookmark17)

1. [DEFINIÇÃO DA INTERFACE COM O USUÁRIO (UX) 8](#_bookmark18)
   1. [Descrição de cenário 8](#_bookmark19)
   2. [Descrição de personas 8](#_bookmark20)
   3. [Esboços de tela (wireframes) 8](#_bookmark21)
   4. [Protótipos de tela 8](#_bookmark22)
   5. [Acessibilidade 8](#_bookmark23)
2. [BANCO DE DADOS 9](#_bookmark24)
   1. [Modelo Entidade Relacionamento 9](#_bookmark25)
   2. [Script das tabelas 9](#_bookmark26)
   3. [Mapeamento Objeto Relacional – ORM 10](#_bookmark27)
3. [ARQUITETURA DE SOFTWARE 11](#_bookmark28)
   1. [Arquitetura de desenvolvimento 11](#_bookmark29)
      1. [BACK-END 11](#_bookmark30)
      2. [FRONT-END - WEB 11](#_bookmark31)
      3. [Front-End - Mobile 11](#_bookmark32)
      4. [Serviços de Mensageria 11](#_bookmark33)
   2. [Segurança da informação 12](#_bookmark34)
   3. [Implantação 12](#_bookmark35)
4. [CONCLUSÃO 13](#_bookmark36)
5. [REFERÊNCIAS 14](#_bookmark37)

# 1 INTRODUÇÃO

A história do estágio no Brasil remonta a décadas passadas. O primeiro registro legal sobre a prática do estágio no país ocorreu em 1942, com a promulgação do Decreto-Lei nº 4.073, que instituiu a Lei Orgânica do Ensino Normal. No entanto, foi somente em 1967 que o estágio escolar foi oficialmente estabelecido, por meio da Portaria 1.002 do Ministro do Trabalho, que disciplinava a relação entre as empresas e os estudantes.

Historicamente, o conceito de Estágio Supervisionado foi introduzido no Brasil pela Lei Orgânica do Ensino Normal, promulgada em 1946. Essa lei definiu o estágio supervisionado como uma etapa fundamental da formação de professores e estabeleceu diretrizes para sua realização. A partir daí o estágio passou a ser reconhecido como uma prática essencial para a preparação profissional dos estudantes.

Ao longo dos anos, a legislação de estágio no Brasil passou por várias alterações. Diversas leis e decretos foram promulgados para regulamentar o estágio, visando garantir direitos e deveres tanto para as empresas como para os estudantes. Essas mudanças refletiram a evolução do mercado de trabalho e das necessidades educacionais.

A Lei nº 11.788, de 25 de setembro de 2008, conhecida como Lei do Estágio, é a principal legislação que trata do assunto. Essa lei define o estágio como um ato educativo escolar supervisionado, desenvolvido no ambiente de trabalho, que tem como objetivo complementar o processo de ensino e aprendizagem dos estudantes. Ela trouxe importantes mudanças na forma como o estágio é realizado no Brasil, regulamentando a contratação de estagiários e buscando soluções para os desafios enfrentados nessa área.

Com a Lei do Estágio estabelece direitos e responsabilidades tanto para os estagiários quanto para as empresas que os contratam. Além disso, define os papéis e responsabilidades dos agentes envolvidos no estágio, como as instituições de ensino, as empresas e os estagiários.

Atualmente, no Brasil é visto como uma porta de entrada para o mercado de trabalho, sendo bastante valorizado por empresas e instituições. Muitas vezes, empresas utilizam o estágio como uma forma de recrutamento, contratando estagiários que se destacam durante o período de estágio. Além disso, o estágio pode ser uma oportunidade para os estudantes demonstrarem suas habilidades e adquirirem experiência prática na área em que estão estudando. Em relação às estatísticas, em 2021, as atividades jurídicas foram as que mais empregaram estagiários no Brasil, com um total de 56,7 mil vagas em todo o país. Além disso, as escolas dos ensinos infantil e fundamental também apresentaram uma demanda significativa por estagiários. No entanto, não há dados estatísticos disponíveis sobre a participação de estudantes e empresas no programa de estágio em geral. A Associação Brasileira de Estágios (Abres) realiza anualmente uma pesquisa para obter informações sobre o número de estagiários no Brasil, utilizando fontes como o Inep, MEC e IBGE para compilar os dados estatísticos sobre a participação de estudantes e empresas no programa de estágio. De acordo com a pesquisa divulgada pelo Centro de Integração Empresa-Escola (CIEE), o número de contratos de estágio cresceu 18% em 2022. Nos primeiros seis meses de 2023, o Instituto Euvaldo Lodi (IEL) inseriu mais de 43 mil estudantes em empresas de todo o país, representando um aumento de 30% em comparação com o mesmo período do ano anterior. As oportunidades de estágio são destinadas a estudantes de diversas áreas, como Administração, Biologia, Biomedicina, Comércio Exterior, Economia, Marketing, Jornalismo, Letras, entre outras.

# 2 LEVANTAMENTO DE REQUISITOS DE SOFTWARE

O levantamento de requisitos é um estágio crucial no ciclo de desenvolvimento de software, desempenhando um papel fundamental na compreensão das necessidades dos usuários e na definição clara do escopo do projeto. Para Roger S. Pressman:

A engenharia de requisitos fornece o mecanismo apropriado para entender o que o cliente deseja, analisando as necessidades, avaliando a exequibilidade, negociando uma condição razoável, especificando e gerindo os requisitos à medida que eles são transformados em um sistema operacional (PRESSMAN, 2011, p.118).

Conforme observado por Sommerville (2018), o levantamento de requisitos não apenas estabelece a base fundamental, mas também se configura como um estágio essencial no ciclo de vida do desenvolvimento de software. Sua importância reside na capacidade de construir um software que atenda às demandas dos usuários e agregue valor ao negócio. Esse processo crítico assegura o sucesso do projeto ao garantir uma compreensão clara e abrangente das expectativas e requisitos do cliente, delineando diretrizes cruciais para a construção de um sistema que verdadeiramente satisfaça as necessidades e objetivos estabelecidos.

2.1 DESCRIÇÃO DOS OBJETIVOS DO SISTEMA

Este projeto tem como objetivo principal promover um acesso mais amplo e dinâmico aos alunos da Fatec Jales, facilitando a interação entre estudantes e empresas no contexto de estágios e oportunidades de emprego. Pretende-se desenvolver um sistema intuitivo e eficiente, que simplifique o processo de cadastro de vagas pelos alunos e a disponibilização dessas oportunidades pelas empresas parceiras.

O sistema será projetado para ser limpo, amigável e de fácil navegação, garantindo que tanto os alunos quanto as empresas possam utilizá-lo sem dificuldades. Para os alunos, o sistema oferecerá um portal onde poderão se cadastrar para as vagas de estágio e emprego de maneira rápida e organizada, além de permitir a consulta de informações detalhadas sobre cada oportunidade, incluindo requisitos, descrições de atividades e benefícios oferecidos.

Para as empresas, o sistema proporcionará uma plataforma onde poderão publicar vagas, gerenciar candidatos e acompanhar todo o processo de seleção de maneira prática e transparente. Com isso, espera-se não apenas facilitar a inserção dos alunos no mercado de trabalho, mas também fortalecer a conexão entre a Fatec Jales e o setor empresarial, promovendo uma colaboração mais estreita e produtiva.

Além disso, o projeto visa assegurar que todos os docentes da Faculdade tenham acesso às informações sobre as oportunidades de estágio e emprego, permitindo que possam orientar e apoiar seus alunos de forma mais eficaz. Dessa forma, este sistema contribuirá para uma formação mais completa e alinhada às demandas do mercado de trabalho, reforçando o compromisso da Fatec Jales com a excelência educacional e profissional de seus estudantes.

2.2 DESCRIÇÃO DO SISTEMA ATUAL

Atualmente, o sistema de estágio no Brasil é regulamentado pela Lei nº 11.788/2008, popularmente conhecida como Lei do Estágio. Essa legislação define o estágio como um ato educativo escolar supervisionado, desenvolvido no ambiente de trabalho com o objetivo de proporcionar ao estudante uma experiência prática diretamente relacionada à sua área de estudo. A Lei do Estágio abrange estágios de ensino médio, técnico e superior, incluindo estudantes brasileiros e estrangeiros.

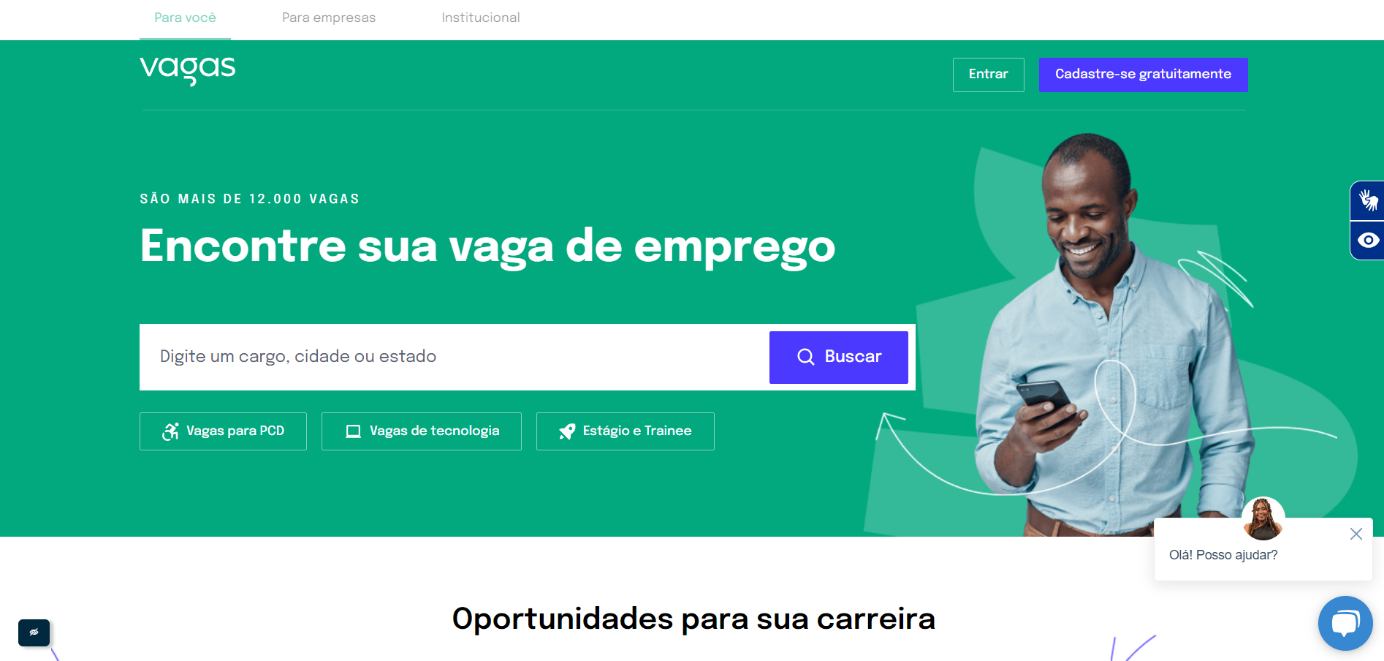
Em vigor desde setembro de 2008, a Lei do Estágio trouxe importantes regulamentações para garantir a qualidade e a segurança dos estágios. Entre as principais disposições, destaca-se a carga horária máxima de 6 horas diárias e 30 horas semanais para os estagiários. Além disso, a lei estabelece a necessidade de um termo de compromisso, que deve ser firmado entre o estagiário, a instituição de ensino e a empresa, visando assegurar a conformidade e a supervisão adequada das atividades realizadas.

2.3 Análise de Sistemas Existentes

No mundo dinâmico da tecnologia e dos negócios, a análise de sistemas existentes desempenha um papel crucial na identificação e solução de problemas, bem como na melhoria contínua das operações. Este processo envolve o exame detalhado dos sistemas que estão em operação, com o objetivo de entender seu funcionamento, identificar suas forças e fraquezas e descobrir oportunidades para aprimoramento.

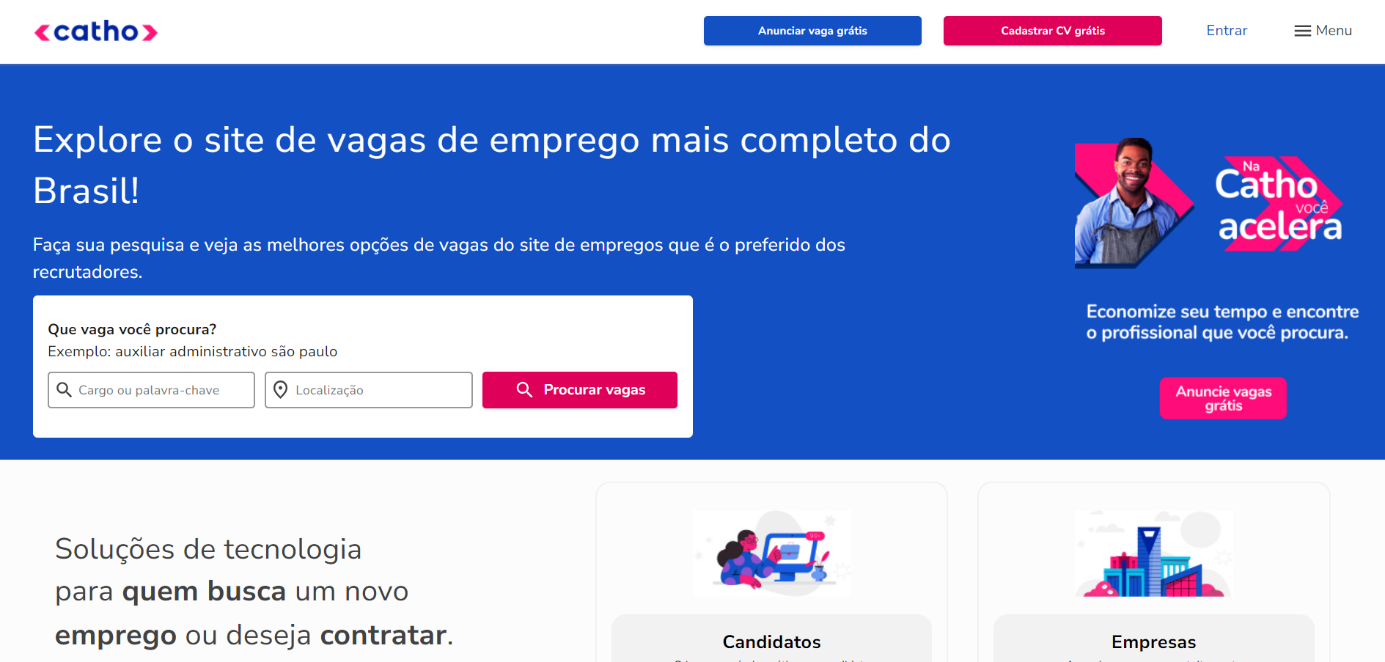
O Vagas – Sistema de divulgação de vagas (Figura 1), é um sistema que tem como objetivo principal conectar candidatos a oportunidades de emprego em diversas áreas e níveis de experiência. Ele oferece uma plataforma gratuita onde empresas podem divulgar vagas e conduzir processos seletivos, enquanto os candidatos podem cadastrar seus currículos, buscar vagas, e se preparar para entrevistas através de conteúdos especializados. O site facilita a interação entre candidatos e empresas, ajudando na recolocação profissional e no desenvolvimento de carreiras.

**Figura 1 –** Tela do Sistema “Vagas – Sistema de divulgação de vagas”

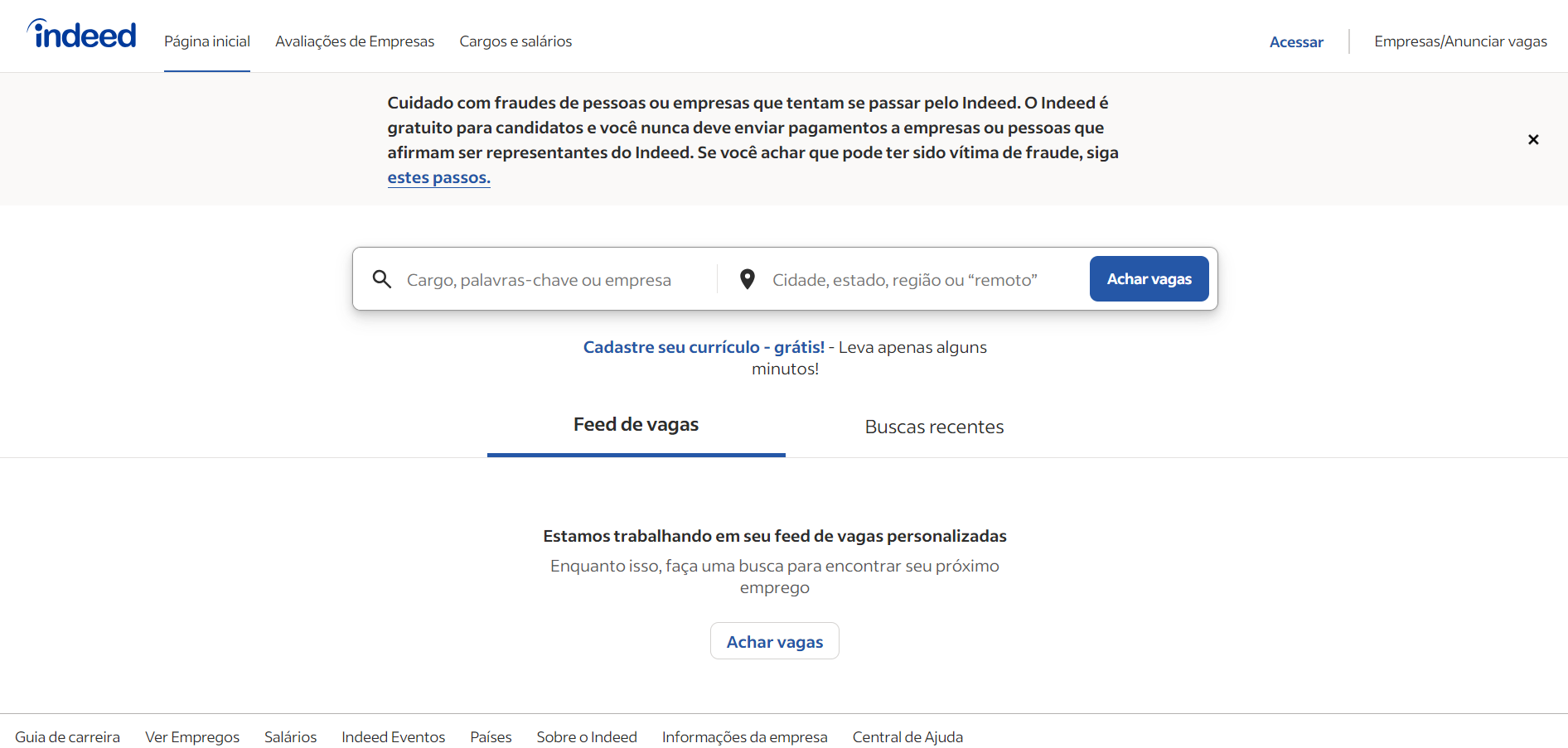


O Catho (**Figura 2**), é um sistema que tem como objetivo de conectar candidatos a oportunidades de emprego em todo o Brasil. Ele oferece uma plataforma onde os usuários podem criar e cadastrar currículos gratuitamente, buscar vagas por área ou cidade, e candidatar-se a posições disponíveis. Para as empresas, o site permite anunciar vagas e buscar currículos de candidatos qualificados. Além disso, a Catho oferece ferramentas e conteúdo para ajudar candidatos na preparação para o mercado de trabalho, como guias de profissões, calculadoras trabalhistas e modelos de currículo.

**Figura 2 –** Tela do Sistema “Catho”



O Indeed (Figura 3), é sistema que tem como objetivo principal ajudar pessoas a encontrar empregos. Ele permite que os candidatos pesquisem vagas de emprego publicadas em milhares de sites e empresas diretamente em um só lugar. Além disso, os usuários podem cadastrar seus currículos, configurar alertas de emprego, e acessar informações sobre empresas e salários para facilitar a busca de emprego. O Indeed também fornece ferramentas e recursos para ajudar candidatos a se prepararem para o mercado de trabalho, como conselhos de carreira e dicas para entrevistas.

**Figura 3 –** Tela do Sistema “Indeed”

2.4 Descrição dos principais problemas

Um desafio potencial é o manuseio inicial do software, considerando que a Fatec Jales nunca utilizou um sistema de inscrição para estágios. Esse fator pode tornar o processo um pouco complicado e confuso para os usuários de primeira viagem. Para mitigar esse problema, é essencial implementar um programa de treinamento adequado e fornece suporte contínuo, garantindo que tanto alunos quanto empresas parceiras se familiarizem rapidamente com a nova plataforma.

2.5 Descrição dos requisitos funcionais

|  |  |  |
| --- | --- | --- |
|  | Requisitos  Funcionais | Descrição |
| 1 | Cadastro de usuarios | O sistema deve ser capaz de cadastrar usuários para a sua utilização. |
| 2 | Cadastro de vagas | O sistema deve ser capaz de cadastrar a vaga que o usuário escolheu. |
| 3 | Alteração de dados do usuário | O sistema deve ser capaz de relaziar a alteração de dados do usuário. |
| 4 | Cadastro de currículo | O sistema deve ser capaz de cadastrar o currículo do usuário caso o mesmo opite por fazer no sistema. |
| 6 | Carregar currículo em PDF | O sistema deve ser capaz de conseguir realizar uploads externos caso o usuário deseje enviar seu currículo. |
| 7 | Cadastro de vaga | O sistema deve ser capaz de realizar o cadastro de vagas das empresas cadastradas no sistema. |
| 8 | Alteração de vaga | O sistema deve ser capaz de realizar alterações na vaga publicada. |
| 9 | Visualizar candidatos para vaga | O sistema deve ser capaz de mostrar a visulização dos candidatos pretentendes a vaga. |
| 10 | Cancelar vaga | O sistema deve ser capaz de fazer o cancelamento da vaga. |

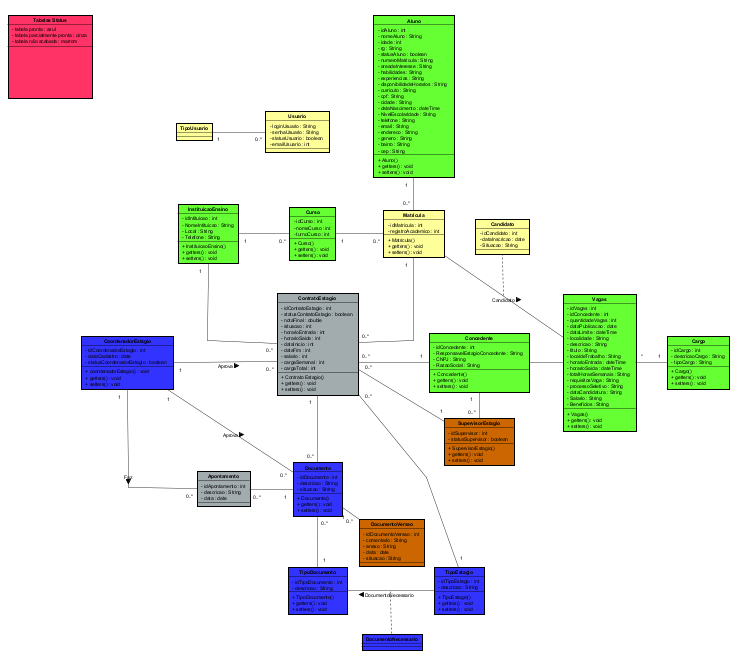
2.6 DESCRIÇÃO DOS REQUISITOS NÃO FUNCIONAIS

|  |  |  |
| --- | --- | --- |
|  | Requisitos não funcionais | Descrição |
| 1 | Atualização do banco de dados | Aprimorar a agilidade na atualização do banco de dados para que nao contenha informações erradas ou desatualizadas. |
| 2 | Máquina capaz de utilizar o sistema | É de suma importancia que o dispositivo tenha alguns componentes que façam ele conseguir suportar o sistema |
| 3 | Auxiliar na utilização do sistema | Oferecemos suporte aos usuário na hora de fazer a atualização explicando as mudanças e como utiliza-lo da melhor maneira. |

# 3 VISÃO DE CASO DE USO – UML

3.1 DIAGRAMA DE CLASSES

O diagrama de classes destaca-se como um dos elementos mais cruciais da UML, desempenhando um papel fundamental ao oferecer suporte para a maioria dos demais diagramas. Conforme abordado na obra, esse diagrama assume a responsabilidade de definir a estrutura das classes utilizadas pelo sistema, delineando seus atributos, métodos e estabelecendo as inter-relações entre as classes. (GUEDES, 2018).

**Figura 3 —**Diagrama de Classes

3.2 DICIONÁRIO DE CLASSES

O dicionário de classes é uma ferramenta essencial na modelagem orientada a objetos, que descreve as classes e seus atributos, métodos e relacionamentos em um sistema. Segundo Sommerville (2011), o dicionário de classes auxilia na definição formal dos objetos que compõem um sistema, estabelecendo uma visão clara e organizada das entidades que serão implementadas. Ele serve como um guia para desenvolvedores e projetistas, garantindo que todos os elementos de um sistema sejam representados de maneira consistente e alinhada aos requisitos funcionais. O dicionário descreve não apenas os atributos de cada classe, mas também detalha os métodos que essas classes devem implementar, facilitando a comunicação entre os membros da equipe de desenvolvimento.

**Quadro 1** – Descrição Classe Aluno

|  |  |  |
| --- | --- | --- |
| Atributo | Tipo | Descrição |
| idAluno | Integer | Utilizado para localizar o aluno dentro do sistema. |
| NomeAluno | String | Nomeia os usuarios do sistema. |
| Idade | Integer | Informa a idade dos usuários . |
| RG | String | Registro geral dos usuários. |
| StatusAluno | Boolean | Determina e informa qual o cargo do aluno a qual etapa esta da contratação. |
| NumeroMatricula | String | Ordena e numera as matrículas. |
| AreaInteresse | String | Determina qual área o estagiário tem interesse em atuar. |
| Habilidades | String | É informado quais são as habilidades que o usuários possui. |
| DisponibilidadesHorários | String | Quais os horários o usuário tem disponivel. |
| Currículo | String | É apresentado o currículo do usuário. |
| CPF | String | É apresentado o currículo do usuário . |
| DataNascimento | DataTime | É informado a data de nascimento do usuário. |
| NívelEscolaridade | String | Informa qual o nível de escolaridade do usuário. |
| Telefone | String | Mostra o telefone do usuário. |
| E-mail | String | E-mail do usuário. |
| Endereço | String | Informa o endereço do aluno. |
| Gênero | String | Especifica o gênero do usuário. |
| Bairro | String | Informa o bairro do usuário. |
| Cep | String | É informado o código de endereçamento postal. |

Fonte: Elaborado pelos autores

**Quadro 2** — Descrição Classe Vagas

|  |  |  |
| --- | --- | --- |
| Atributo | Tipo | Descrição |
| idVagas | Integer | É utilizado para localizar dentro dos sistemas cada uma das vagas |
| IdConcendente | Integer | É usado para informar qual o identificador da empresa que está contratando o estagiário |
| QuantidadeVagas | Integer | Informa exatamente o número de vagas existentes de determinada empresa |
| DataPublicaçao | Date | Informa e oficializa a postagem da vaga |
| DataLimite | dateTime | É defina uma data limite para a inscrição do estagiário para determinada vaga |
| Localidade | String | Informa a localidade em que a empresa se encontra rua, bairro e cidade |
| Descriçao | String | Na descrição será descrito todas as informações referentes a vaga de estágio |
| Titulo | String | Informa como um previa do que o estagiário irá exercer na vaga |
| LocaldeTrabalho | String | Local de trabalho se refere a empresa ou local físico que o estagiário irá trabalhar |
| HorarioEntrada | dateTime | Ira definir a entrada em que o estagiário deverá estar para cumprir sua carga horária |
| HorarioSaida | DataTime | Hora de saída de acordo com a carga horaria presente no contrato e as horas que precisam ser cumpridas na semana |
| totalHorasSemanais | String | Define a carga horaria total estipulada pelo contrato de acordo com as regras de estágio |

Fonte: Elaborado pelos autores

**Quadro 3** — Descrição Classe Usuário

|  |  |  |
| --- | --- | --- |
| Atributo | Tipo | Descrição |
| loginUsuario | String | É o que autêntica o usuário no sistema |
| senhaUsuario | String | É o utilizado juntamente com o login para a autenticação do usuário |
| statusUsuario | Boolean | Mostra qual o status do usuário dentro dos sistemas |
| emailUsuario | Integer | É um meio de comunicação com o usuário |

Fonte: Elaborado pelos autores

**Quadro 4** — Descrição Classe Candidato

|  |  |  |
| --- | --- | --- |
| Atributo | Tipo | Descrição |
| idCandidato | Integer | Identifica o candidato |
| dataInscricao | Date | Refere-se ao dia em que o aluno se inscreveu para aquela vaga |
| Situacao | String | Mostra a situação do candidato perante a vaga |

Fonte: Elaborado pelos autores

**Quadro 5** — Descrição Classe Concedente

|  |  |  |
| --- | --- | --- |
| Atributo | Tipo | Descrição |
| idConcedente | Integer | Identifica qual é a empresa que irá conceder o estágio |
| ResponsavelEstagioConcedente | String | Mostra quem é a pessoa de dentro da empresa que é responsável por conceder o estágio |

Fonte: Elaborado pelos autores

**Quadro 6** — Descrição Classe Cargo

|  |  |  |
| --- | --- | --- |
| Atributo | Tipo | Descrição |
| idCargo | Integer | Identifica qual é o cargo |
| descricaoCargo | String | Descreve como será o cargo exercido com detalhes |
| tipoCargo | String | Define quais funções o estagiário irá exercer na instituição |

Fonte: Elaborado pelos autores

3.3 DEFINIÇÃO DOS ATORES

O diagrama de caso de uso concentra-se predominantemente em dois elementos fundamentais: atores e casos de uso. Os atores representam os usuários do sistema, assim como qualquer hardware especial ou sistema integrado que possa utilizar as funcionalidades e serviços oferecidos pelo sistema. Na representação gráfica, símbolos como bonecos são empregados para identificar os atores, acompanhados de breves descrições que detalham suas funções específicas no sistema. Essa representação visual simplifica a compreensão dos papéis desempenhados por diferentes componentes. Cada descrição associada a um ator destaca sua função específica no contexto do sistema, facilitando a análise e compreensão durante o processo de modelagem, conforme preconizado no livro "UML 2 - Uma Abordagem Prática".

**Aluno:** responsável pelo uso da aplicação. Esse usuário deve realizar o seu cadastro, e interagir com o sistema sendo o estagiário ou um possível candidato à vaga.

**Candidato:** responsável pelo uso da aplicação. É o usuário que passa do “status” de aluno para entrar de fato como um postulante a vaga de estágio.

**Concedente:** responsável por conceder o estágio. Esse usuário pode ser uma empresa, uma pessoa ou a própria instituição, é responsável por conceder as vagas de estágio para os alunos.

**Supervisor Estágio:** responsável por supervisionar o estagiário. Esse usuário deve realizar o supervisionamento do estagiário, atribuir tarefas e assim contratos por exemplo, pode ser da própria instituição de ensino ou da concedente de estágio.

**Coordenador Estágio:** responsável por coordenar as vagas de estágio. Esse usuário na maioria das vezes é da própria instituição, é o responsável da instituição por distribuir e criar vagas de estágio.

* 1. DIAGRAMA DE CASOS DE USO

Conforme enfatizado na obra "UML 2 - Uma Abordagem Prática", o Diagrama de Caso de Uso é o mais abstrato, sendo, portanto, o mais flexível e informal entre as representações, utilizado para proporcionar uma visão global do sistema. Seu propósito central é a identificação dos atores que interagirão com o software, acompanhados de suas respectivas funcionalidades. Este diagrama desfruta de ampla aplicação nas fases iniciais de levantamento e análise de requisitos, sendo, no entanto, igualmente valioso ao longo de todo o processo de modelagem, podendo inclusive servir como alicerce para a elaboração de outros diagramas.

**Quadro 7 – Mensagens de saída**

| **Identificação** | **Mensagem** | **Descrição** |
| --- | --- | --- |
| Msg01 | Ações bem-sucedidas | Está mensagem ocorre quando o ator cadastra um novo registro no banco de dados. |
| Msg02 | Alterado com sucesso. | Está mensagem ocorre quando o ator altera um registro no banco de dados. |
| Msg03 | Excluído com sucesso. | Está mensagem ocorre quando o ator exclui um registro no banco de dados. |
| Msg4 | Deseja realmente excluir? | Está mensagem ocorre quando o ator requisita a exclusão de um registro para o sistema. |
| Msg5 | Dados vazios! | Está mensagem ocorre quando os dados para cadastro ou alteração estão vazios. |
| Msg6 | Dados inválidos! | Está mensagem ocorre quando os dados para cadastro ou alteração são inválidos. |

Fonte: Elaborado pelos autores.

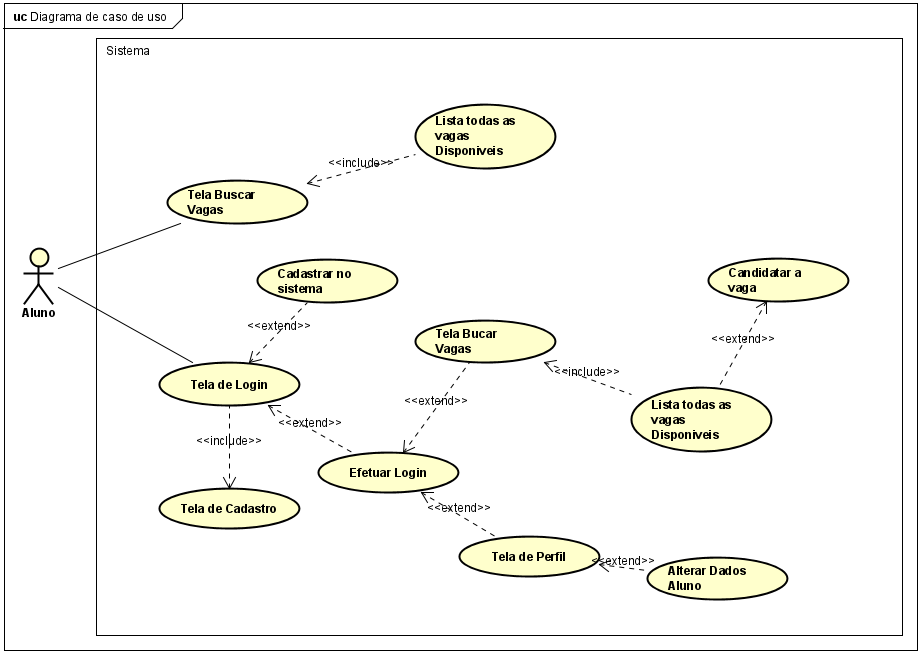
**Quadro 8 – Lista de Casos de Uso**

| **Nº** | **Descrição do Caso de Uso** | **Entrada** | **Caso de Uso** | **Resposta** |
| --- | --- | --- | --- | --- |
| **1** | Alterar Administrador | Informa dados administrador | Alterar administrador | Retorna mensagem “Alterado com sucesso” |
| **2** | Administrador altera matrícula | Informa dados da matrícula | Alterar matrícula | Retorna mensagem “Alterado com sucesso” |
| **3** | Administrador altera curso | Informa dados do curso | Alterar curso | Retorna mensagem “Alterado com sucesso” |
| **4** | Administrador altera candidato | Informa dados do candidato | Alterar candidato | Retorna mensagem “Alterado com sucesso” |
| **5** | Administrador altera instituição de ensino | Informa dados instituição de ensino | Altera instituição de ensino | Retorna mensagem “Alterado com sucesso” |
| **6** | Administrador altera coordenador de estágio | Informa dados coordenador de estágio | Altera coordenador de estágio | Retorna mensagem “Alterado com sucesso” |
| **7** | Administrador altera contrato estágio | Informa dados contrato estágio | Altera contrato estágio | Retorna mensagem “Alterado com sucesso” |
| **8** | Administrador altera concedente | Informa dados concedente | Altera concedente | Retorna mensagem “Alterado com sucesso” |
| **9** | Administrador altera vagas | Informa dados vagas | Altera vagas | Retorna mensagem “Alterado com sucesso” |
| **10** | Administrador altera cargo | Informa dados cargo | Altera cargo | Retorna mensagem “Alterado com sucesso” |
| **11** | Administrador altera requisito vagas | Informa dados requisito vaga | Altera requisito vagas | Retorna mensagem “Alterado com sucesso” |
| **12** | Administrador altera supervisor estágio | Informa dados supervisor estágio | Altera supervisor estágio | Retorna mensagem “Alterado com sucesso” |
| **13** | Administrador altera processo seletivo | Informa dados processo seletivo | Altera processo seletivo | Retorna mensagem “Alterado com sucesso” |
| **14** | Administrador altera apontamento | Informa dados apontamento | Altera apontamento | Retorna mensagem” Alterado com sucesso” |
| **15** | Administrador altera documento | Informa dados documento | Altera documento | Retorna mensagem” Alterado com sucesso” |
| **16** | Administrador altera tipo documento | Informa dados tipo documento | Altera tipo documento | Retorna mensagem” Alterado com sucesso” |
| **17** | Administrador altera documento versão | Informa dados documento versão | Altera documento versão | Retorna mensagem” Alterado com sucesso” |
| **18** | Administrador altera documento necessário | Informa documento necessário | Altera documento necessário | Retorna mensagem” Alterado com sucesso” |
| **19** | Administrador altera tipo estágio | Informa tipo estágio | Altera tipo estágio | Retorna mensagem” Alterado com sucesso” |
| **20** | Cadastrar Administrador | Informa dados administrador | Cadastrar administrador | Retorna mensagem “Cadastrado com sucesso” |
| **21** | Administrador cadastra matrícula | Informa dados da matrícula | Cadastrar matrícula | Retorna mensagem “Cadastrado com sucesso” |
| **22** | Administrador cadastra curso | Informa dados do curso | Cadastrar curso | Retorna mensagem “Cadastrado com sucesso” |
| **23** | Administrador cadastra candidato | Informa dados do candidato | Cadastra candidato | Retorna mensagem “Cadastrado com sucesso” |
| **24** | Administrador cadastra instituição de ensino | Informa dados instituição de ensino | Cadastra instituição de ensino | Retorna mensagem “Cadastrado com sucesso” |
| **25** | Administrador cadastra coordenador de estágio | Informa dados coordenador de estágio | Cadastra coordenador de estágio | Retorna mensagem “Cadastrado com sucesso” |
| **26** | Administrador cadastra contrato estágio | Informa dados contrato estágio | Cadastra contrato estágio | Retorna mensagem “Cadastrado com sucesso” |
| **27** | Administrador cadastra concedente | Informa dados concedente | Cadastra concedente | Retorna mensagem “Cadastrado com sucesso” |
| **28** | Administrador cadastra vagas | Informa dados vagas | Cadastra vagas | Retorna mensagem “Cadastrado com sucesso” |
| **29** | Administrador cadastra cargo | Informa dados cargo | Cadastra cargo | Retorna mensagem “Cadastrado com sucesso” |
| **30** | Administrador cadastra requisito vagas | Informa dados requisito vaga | Cadastra requisito vagas | Retorna mensagem “Cadastrado com sucesso” |
| **31** | Administrador cadastra supervisor estágio | Informa dados supervisor estágio | Cadastra supervisor estágio | Retorna mensagem “Cadastrado com sucesso” |
| **32** | Administrador cadastra processo seletivo | Informa dados processo seletivo | Cadastra processo seletivo | Retorna mensagem “Cadastrado com sucesso” |
| **33** | Administrador cadastra apontamento | Informa dados apontamento | Cadastra apontamento | Retorna mensagem” Cadastrado com sucesso” |
| **34** | Administrador cadastra documento | Informa dados documento | Cadastra documento | Retorna mensagem” Cadastrado com sucesso” |
| **35** | Administrador cadastra tipo documento | Informa dados tipo documento | Cadastra tipo documento | Retorna mensagem” Cadastrado com sucesso” |
| **36** | Administrador cadastra documento versão | Informa dados documento versão | Cadastra documento versão | Retorna mensagem” Cadastrado com sucesso” |
| **37** | Administrador cadastra documento necessário | Informa documento necessário | Cadastra documento necessário | Retorna mensagem” Cadastrado com sucesso” |
| **38** | Administrador cadastra tipo estágio | Informa tipo estágio | Cadastra tipo estágio | Retorna mensagem” Cadastrado com sucesso” |
| **39** | Desativar Administrador | Id Administrador | Desativa Administrador | Retorna mensagem “Desativado com sucesso” |
| **40** | Administrador desativa matrícula | Id matrícula | Desativa matrícula | Retorna mensagem “Desativado com sucesso” |
| **41** | Administrador desativa curso | Id curso | Desativa curso | Retorna mensagem “Desativado com sucesso” |
| **42** | Administrador desativa candidato | Id candidato | Desativa candidato | Retorna mensagem “Desativado com sucesso” |
| **43** | Administrador desativa instituição de ensino | Id instituição de ensino | Desativa instituição de ensino | Retorna mensagem “Desativado com sucesso” |
| **44** | Administrador desativa coordenador de estágio | Id coordenador de estágio | Desativa coordenador de estágio | Retorna mensagem “Desativado com sucesso” |
| **45** | Administrador desativa contrato estágio | Id contrato estágio | Desativa contrato estágio | Retorna mensagem “Desativado com sucesso” |
| **46** | Administrador desativa concedente | Id concedente | Desativa concedente | Retorna mensagem “Desativado com sucesso” |
| **47** | Administrador desativa vagas | Id vagas | Desativa vagas | Retorna mensagem “Desativado com sucesso” |
| **48** | Administrador desativa cargo | Id cargo | Desativa cargo | Retorna mensagem “Desativado com sucesso” |
| **49** | Administrador desativa requisito vagas | Id requisito vaga | Desativa requisito vagas | Retorna mensagem “Desativado com sucesso” |
| **50** | Administrador desativa supervisor estágio | Id supervisor estágio | Desativa supervisor estágio | Retorna mensagem “Desativado com sucesso” |
| **51** | Administrador desativa processo seletivo | Id processo seletivo | Desativa processo seletivo | Retorna mensagem “Desativado com sucesso” |
| **52** | Administrador desativa apontamento | Id apontamento | Desativa apontamento | Retorna mensagem” Desativado com sucesso” |
| **53** | Administrador desativa documento | Id documento | Desativa documento | Retorna mensagem” Desativado com sucesso” |
| **54** | Administrador desativa tipo documento | Id tipo documento | Desativa tipo documento | Retorna mensagem” Desativado com sucesso” |
| **55** | Administrador desativa documento versão | Id documento versão | Desativa documento versão | Retorna mensagem” Desativado com sucesso” |
| **56** | Administrador desativa documento necessário | Id documento necessário | Desativa documento necessário | Retorna mensagem” Desativado com sucesso” |
| **57** | Administrador desativa tipo estágio | Id tipo estágio | Desativa tipo estágio | Retorna mensagem” Desativado com sucesso” |
| **58** | Exclui Administrador | Id Administrador | Exclui Administrador | Retorna mensagem “Excluído com sucesso” |
| **59** | Administrador exclui matrícula | Id matrícula | Exclui matrícula | Retorna mensagem “Desativado com sucesso” |
| **60** | Administrador exclui curso | Id curso | Exclui curso | Retorna mensagem “Excluído com sucesso” |
| **61** | Administrador exclui candidato | Id candidato | Exclui candidato | Retorna mensagem “Excluído com sucesso” |
| **62** | Administrador exclui instituição de ensino | Id instituição de ensino | Exclui instituição de ensino | Retorna mensagem “Excluído com sucesso” |
| **63** | Administrador exclui coordenador de estágio | Id coordenador de estágio | Exclui coordenador de estágio | Retorna mensagem “Excluído com sucesso” |
| **64** | Administrador exclui contrato estágio | Id contrato estágio | Exclui contrato estágio | Retorna mensagem “Excluído com sucesso” |
| **65** | Administrador exclui concedente | Id concedente | Desativa concedente | Retorna mensagem “Excluído com sucesso” |
| **66** | Administrador exclui vagas | Id vagas | Exclui vagas | Retorna mensagem “Excluído com sucesso” |
| **67** | Administrador exclui cargo | Id cargo | Exclui cargo | Retorna mensagem “Excluído com sucesso” |
| **68** | Administrador exclui requisito vagas | Id requisito vaga | Exclui requisito vagas | Retorna mensagem “Excluído com sucesso” |
| **69** | Administrador desativa supervisor estágio | Id supervisor estágio | Exclui supervisor estágio | Retorna mensagem “Excluído com sucesso” |
| **70** | Administrador exclui processo seletivo | Id processo seletivo | Exclui processo seletivo | Retorna mensagem “Excluído com sucesso” |
| **71** | Administrador exclui apontamento | Id apontamento | Exclui apontamento | Retorna mensagem” Excluído com sucesso” |
| **72** | Administrador exclui documento | Id documento | Exclui documento | Retorna mensagem” Excluído com sucesso” |
| **73** | Administrador exclui tipo documento | Id tipo documento | Exclui tipo documento | Retorna mensagem” Excluído com sucesso” |
| **74** | Administrador exclui documento versão | Id documento versão | Exclui documento versão | Retorna mensagem” Excluído com sucesso” |
| **75** | Administrador exclui documento necessário | Id documento necessário | Exclui documento necessário | Retorna mensagem” Excluído com sucesso” |
| **76** | Administrador exclui tipo estágio | Id tipo estágio | Exclui tipo estágio | Retorna mensagem” Excluído com sucesso” |
| **77** | Consulta Administrador | Id Administrador | Consulta Administrador | Retorna dados administrador |
| **78** | Administrador consulta matrícula | Id matrícula | Consulta matrícula | Retorna dados administrador |
| **79** | Administrador consulta curso | Id curso | Consulta curso | Retorna dados curso |
| **80** | Administrador consulta candidato | Id candidato | Consulta candidato | Retorna dados candidato |
| **81** | Administrador consulta instituição de ensino | Id instituição de ensino | Consulta instituição de ensino | Retorna dados instituição de ensino |
| **82** | Administrador consulta coordenador de estágio | Id coordenador de estágio | Consulta coordenador de estágio | Retorna dados coordenador de estágio |
| **83** | Administrador consulta contrato estágio | Id contrato estágio | Consulta contrato estágio | Retorna dados contrato estágio |
| **84** | Administrador consulta concedente | Id concedente | Consulta concedente | Retorna dados concedente |
| **85** | Administrador consulta vagas | Id vagas | Consulta vagas | Retorna dados vagas |
| **86** | Administrador consulta cargo | Id cargo | Consulta cargo | Retorna dados cargo |
| **87** | Administrador consulta requisito vagas | Id requisito vaga | Consulta requisito vagas | Retorna dados requisito vagas |
| **88** | Administrador consulta supervisor estágio | Id supervisor estágio | Consulta supervisor estágio | Retorna dados supervisor estágio |
| **89** | Administrador consulta processo seletivo | Id processo seletivo | Consulta processo seletivo | Retorna dados processo seletivo |
| **90** | Administrador consulta apontamento | Id apontamento | Consulta apontamento | Retorna dados apontamento |
| **91** | Administrador consulta documento | Id documento | Consulta documento | Retorna dados documento |
| **92** | Administrador consulta tipo documento | Id tipo documento | Consulta tipo documento | Retorna dados tipo documento |
| **93** | Administrador consulta documento versão | Id documento versão | Consulta documento versão | Retorna dados documento versão |
| **94** | Administrador consulta documento necessário | Id documento necessário | Consulta documento necessário | Retorna dados documento necessário |
| **95** | Administrador consulta tipo estágio | Id tipo estágio | Consulta tipo estágio | Retorna dados tipo estágio |
| **96** | Coordenador Estágio altera apontamento | Informa dados apontamento | Altera apontamento | Retorna mensagem” Alterado com sucesso” |
| **97** | Coordenador Estágio altera documento | Informa dados documento | Altera documento | Retorna mensagem” Alterado com sucesso” |
| **98** | Coordenador Estágio altera tipo documento | Informa dados tipo documento | Altera tipo documento | Retorna mensagem” Alterado com sucesso” |
| **99** | Coordenador Estágio altera documento versão | Informa dados documento versão | Altera documento versão | Retorna mensagem” Alterado com sucesso” |
| **100** | Coordenador Estágio altera documento necessário | Informa documento necessário | Altera documento necessário | Retorna mensagem” Alterado com sucesso” |
| **101** | Coordenador Estágio altera tipo estágio | Informa tipo estágio | Altera tipo estágio | Retorna mensagem” Alterado com sucesso” |
| **102** | Coordenador Estágio cadastra apontamento | Informa dados apontamento | Cadastra apontamento | Retorna mensagem” Cadastrado com sucesso” |
| **103** | Coordenador Estágio cadastra documento | Informa dados documento | Cadastra documento | Retorna mensagem” Cadastrado com sucesso” |
| **104** | Coordenador Estágio cadastra tipo documento | Informa dados tipo documento | Cadastra tipo documento | Retorna mensagem” Cadastrado com sucesso” |
| **105** | Coordenador Estágio cadastra documento versão | Informa dados documento versão | Cadastra documento versão | Retorna mensagem” Cadastrado com sucesso” |
| **106** | Coordenador Estágio cadastra documento necessário | Informa documento necessário | Cadastra documento necessário | Retorna mensagem” Cadastrado com sucesso” |
| **107** | Coordenador Estágio cadastra tipo estágio | Informa tipo estágio | Cadastra tipo estágio | Retorna mensagem” Cadastrado com sucesso” |
| **108** | Coordenador Estágio desativa apontamento | Id apontamento | Desativa apontamento | Retorna mensagem” Desativado com sucesso” |
| **109** | Coordenador Estágio desativa documento | Id documento | Desativa documento | Retorna mensagem” Desativado com sucesso” |
| **110** | Coordenador Estágio desativa tipo documento | Id tipo documento | Desativa tipo documento | Retorna mensagem” Desativado com sucesso” |
| **111** | Coordenador Estágio desativa documento versão | Id documento versão | Desativa documento versão | Retorna mensagem” Desativado com sucesso” |
| **112** | Coordenador Estágio desativa documento necessário | Id documento necessário | Desativa documento necessário | Retorna mensagem” Desativado com sucesso” |
| **113** | Coordenador Estágio desativa tipo estágio | Id tipo estágio | Desativa tipo estágio | Retorna mensagem” Desativado com sucesso” |
| **114** | Coordenador Estágio exclui apontamento | Id apontamento | Exclui apontamento | Retorna mensagem” Excluído com sucesso” |
| **115** | Coordenador Estágio exclui documento | Id documento | Exclui documento | Retorna mensagem” Excluído com sucesso” |
| **116** | Coordenador Estágio exclui tipo documento | Id tipo documento | Exclui tipo documento | Retorna mensagem” Excluído com sucesso” |
| **117** | Coordenador Estágio exclui documento versão | Id documento versão | Exclui documento versão | Retorna mensagem” Excluído com sucesso” |
| **118** | Coordenador Estágio exclui documento necessário | Id documento necessário | Exclui documento necessário | Retorna mensagem” Excluído com sucesso” |
| **119** | Coordenador Estágio exclui tipo estágio | Id tipo estágio | Exclui tipo estágio | Retorna mensagem” Excluído com sucesso” |
| **120** | Coordenador Estágio consulta apontamento | Id apontamento | Consulta apontamento | Retorna dados apontamento |
| **121** | Coordenador Estágio consulta documento | Id documento | Consulta documento | Retorna dados documento |
| **122** | Coordenador Estágio consulta tipo documento | Id tipo documento | Consulta tipo documento | Retorna dados tipo documento |
| **123** | Coordenador Estágio consulta documento versão | Id documento versão | Consulta documento versão | Retorna dados documento versão |
| **124** | Coordenador Estágio consulta documento necessário | Id documento necessário | Consulta documento necessário | Retorna dados documento necessário |
| **125** | Coordenador Estágio consulta tipo estágio | Id tipo estágio | Consulta tipo estágio | Retorna dados tipo estágio |
| **126** | Supervisor Estágio altera apontamento | Informa dados apontamento | Altera apontamento | Retorna mensagem” Alterado com sucesso” |
| **127** | Supervisor Estágio altera documento | Informa dados documento | Altera documento | Retorna mensagem” Alterado com sucesso” |
| **128** | Supervisor Estágio altera tipo documento | Informa dados tipo documento | Altera tipo documento | Retorna mensagem” Alterado com sucesso” |
| **129** | Supervisor Estágio altera documento versão | Informa dados documento versão | Altera documento versão | Retorna mensagem” Alterado com sucesso” |
| **130** | Supervisor Estágio altera documento necessário | Informa documento necessário | Altera documento necessário | Retorna mensagem” Alterado com sucesso” |
| **131** | Supervisor Estágio altera tipo estágio | Informa tipo estágio | Altera tipo estágio | Retorna mensagem” Alterado com sucesso” |
| **132** | Supervisor Estágio cadastra apontamento | Informa dados apontamento | Cadastra apontamento | Retorna mensagem” Cadastrado com sucesso” |
| **133** | Supervisor Estágio cadastra documento | Informa dados documento | Cadastra documento | Retorna mensagem” Cadastrado com sucesso” |
| **134** | Supervisor Estágio cadastra tipo documento | Informa dados tipo documento | Cadastra tipo documento | “Cadastrado com sucesso” |
| **135** | Supervisor Estágio cadastra documento versão | Informa dados documento versão | Cadastra documento versão | Retorna mensagem” Cadastrado com sucesso” |
| **136** | Supervisor Estágio cadastra documento necessário | Informa documento necessário | Cadastra documento necessário | Retorna mensagem” Cadastrado com sucesso” |
| **137** | Supervisor Estágio cadastra tipo estágio | Informa tipo estágio | Cadastra tipo estágio | Retorna mensagem” Cadastrado com sucesso” |
| **138** | Supervisor Estágio desativa apontamento | Id apontamento | Desativa apontamento | Retorna mensagem” Desativado com sucesso” |
| **139** | Supervisor Estágio desativa documento | Id documento | Desativa documento | Retorna mensagem” Desativado com sucesso” |
| **140** | Supervisor Estágio desativa tipo documento | Id tipo documento | Desativa tipo documento | Retorna mensagem” Desativado com sucesso” |
| **141** | Supervisor Estágio desativa documento versão | Id documento versão | Desativa documento versão | Retorna mensagem” Desativado com sucesso” |
| **142** | Supervisor desativa documento necessário | Id documento necessário | Desativa documento necessário | Retorna mensagem” Desativado com sucesso” |
| **143** | Supervisor Estágio desativa tipo estágio | Id tipo estágio | Desativa tipo estágio | Retorna mensagem” Desativado com sucesso” |
| **144** | Supervisor Estágio exclui apontamento | Id apontamento | Exclui apontamento | Retorna mensagem” Excluído com sucesso” |
| **145** | Supervisor Estágio exclui documento | Id documento | Exclui documento | Retorna mensagem” Excluído com sucesso” |
| **146** | Supervisor Estágio exclui tipo documento | Id tipo documento | Exclui tipo documento | Retorna mensagem” Excluído com sucesso” |
| **147** | Supervisor Estágio exclui documento versão | Id documento versão | Exclui documento versão | Retorna mensagem” Excluído com sucesso” |
| **148** | Supervisor Estágio exclui documento necessário | Id documento necessário | Exclui documento necessário | Retorna mensagem” Excluído com sucesso” |
| **149** | Supervisor Estágio exclui tipo estágio | Id tipo estágio | Exclui tipo estágio | Retorna mensagem” Excluído com sucesso” |
| **150** | Supervisor Estágio consulta apontamento | Id apontamento | Consulta apontamento | Retorna dados apontamento |
| **151** | Supervisor Estágio consulta documento | Id documento | Consulta documento | Retorna dados documento |
| **152** | Supervisor Estágio consulta tipo documento | Id tipo documento | Consulta tipo documento | Retorna dados tipo documento |
| **153** | Supervisor Estágio consulta documento versão | Id documento versão | Consulta documento versão | Retorna dados documento versão |
| **154** | Supervisor Estágio consulta documento necessário | Id documento necessário | Consulta documento necessário | Retorna dados documento necessário |
| **155** | Supervisor Estágio consulta tipo estágio | Id tipo estágio | Consulta tipo estágio | Retorna dados tipo estágio |
| **156** | Concedente altera contrato estágio | Informa dados contrato estágio | Cadastra contrato estágio | Retorna mensagem “Cadastrado com sucesso” |
| **157** | Concedente altera vagas | Informa dados vagas | Altera vagas | Retorna mensagem “Alterado com sucesso” |
| **158** | Concedente altera requisito vagas | Informa dados requisito vaga | Altera requisito vagas | Retorna mensagem “Alterado com sucesso” |
| **159** | Concedente cadastra contrato | Informa dados contrato | Cadastra contrato | Retorna mensagem” Cadastrado com sucesso” |
| **160** | Concedente cadastra vagas | Informa dados vagas | Cadastra vagas | Retorna mensagem” Cadastrado com sucesso” |
| **161** | Concedente cadastra requisito vagas | Informa dados requisito vagas | Cadastra requisito vagas | Retorna mensagem” Cadastrado com sucesso” |
| **162** | Concedente desativa contrato | Id contrato | Desativa contrato | Retorna mensagem” Desativado com sucesso” |
| **163** | Concedente desativa requisito vagas | Id requisito vagas | Desativa requisito vagas | “Desativado com sucesso” |
| **164** | Concedente desativa vagas | Id vagas | Desativa vagas | Retorna mensagem” Desativado com sucesso” |
| **165** | Concedente exclui contrato | Id documento contrato | Exclui documento contrato | Retorna mensagem” Excluído com sucesso” |
| **166** | Concedente exclui requisito vagas | Id documento requisito vagas | Exclui documento requisito vagas | Retorna mensagem” Excluído com sucesso” |
| **167** | Concedente exclui vagas | Id documento vagas | Exclui documento vagas | Retorna mensagem” Excluído com sucesso” |
| **168** | Concedente consulta contrato | Id contrato | Consulta contrato | Retorna dados contrato |
| **169** | Concedente consulta requisito vagas | Id requisito vagas | Consulta requisito vagas | Retorna dados requisito vagas |
| **170** | Concedente consulta vagas | Id vagas | Consulta vagas | Retorna dados vagas |
| **171** | Sistema Lista Administrador |  | Listar administrador | Retorna lista administrador |
| **172** | Sistema Lista matrícula |  | Listar matrícula | Retorna lista matrícula |
| **173** | Sistema Lista curso |  | Listar curso | Retorna lista curso |
| **174** | Sistema Lista candidato |  | Listar candidato | Retorna lista candidato |
| **175** | Sistema Lista instituição de ensino |  | Listar instituição de ensino | Retorna lista instituição de ensino |
| **176** | Sistema Lista coordenador de estágio |  | Listar coordenador de estágio | Retorna lista coordenador de estágio |
| **177** | Sistema Lista contrato estágio |  | Listar contrato estágio | Retorna lista contrato estágio |
| **178** | Sistema Lista concedente |  | Listar concedente | Retorna lista concedente |
| **179** | Sistema Lista vagas |  | Listar vagas | Retorna lista vagas |
| **180** | Sistema Lista cargo |  | Listar cargo | Retorna lista cargo |
| **181** | Sistema Lista requisito vagas |  | Listar requisito vagas | Retorna lista requisito vagas |
| **182** | Sistema Lista supervisor estágio |  | Listar supervisor estágio | Retorna lista supervisor estágio |
| **183** | Sistema Lista processo seletivo |  | Listar processo seletivo | Retorna lista processo seletivo |
| **184** | Sistema Lista apontamento |  | Listar apontamento | Retorna lista apontamento |
| **185** | Sistema Lista documento |  | Listar documento | Retorna lista documento |
| **186** | Sistema Lista tipo documento |  | Listar tipo documento | Retorna lista tipo documento |
| **187** | Sistema Lista documento versão |  | Listar documento versão | Retorna lista documento versão |
| **188** | Sistema Lista documento necessário |  | Listar documento necessário | Retorna lista documento necessário |
| **189** | Sistema Lista estágio |  | Listar tipo estágio | Retorna lista estágio |

## Diagrama de Casos de Uso

Conforme enfatizado na obra "UML 2 - Uma Abordagem Prática", o Diagrama de Caso de Uso é o mais abstrato, sendo, portanto, o mais flexível e informal entre as representações, utilizado para proporcionar uma visão global do sistema. Seu propósito central é a identificação dos atores que interagirão com o software, acompanhados de suas respectivas funcionalidades. Este diagrama desfruta de ampla aplicação nas fases iniciais de levantamento e análise de requisitos, sendo, no entanto, igualmente valioso ao longo de todo o processo de modelagem, podendo inclusive servir como alicerce para a elaboração de outros diagramas

**Figura 04 —** Diagrama de Contexto – Visão do Aluno



Fonte: Elaborado pelos autores.

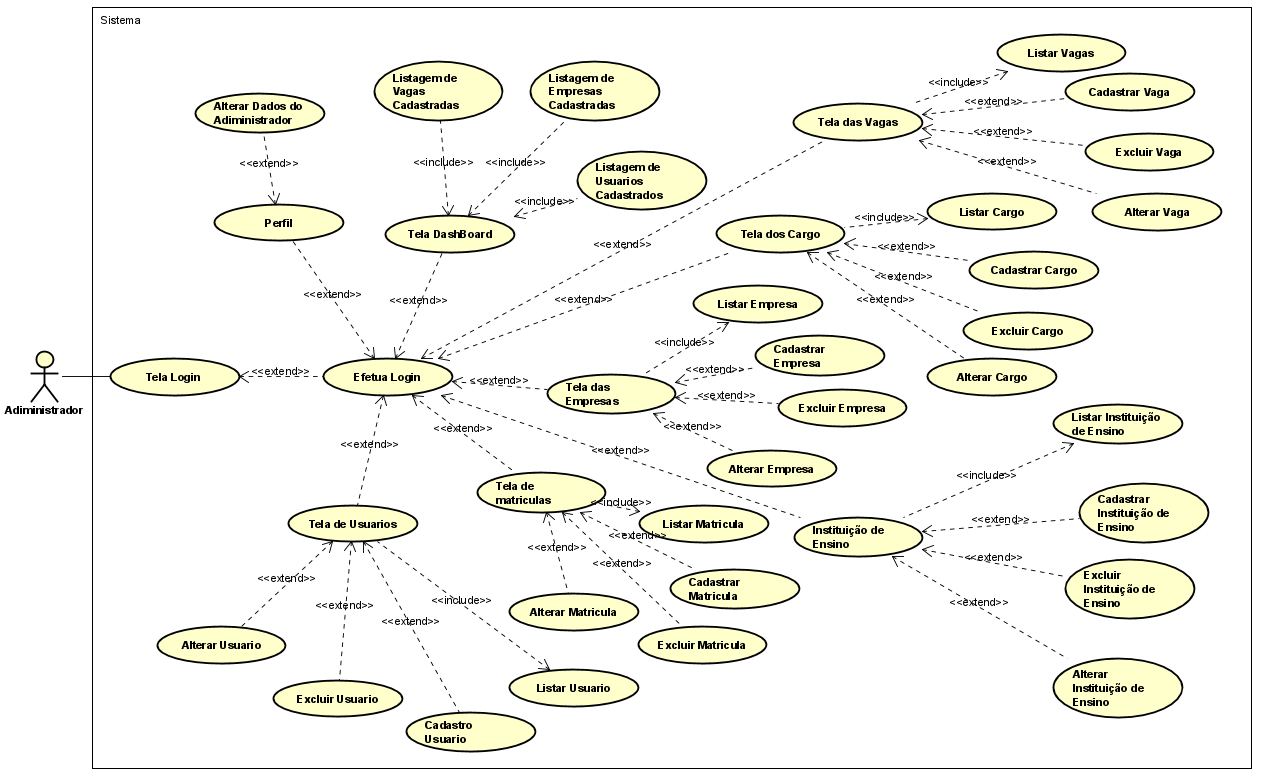
De acordo com a figura 04, o aluno pode interagir com várias funcionalidades do sistema. Inicialmente, ele acessa a tela de cadastro, onde pode criar uma nova conta fornecendo informações essenciais, como nome, e-mail e senha. Após o cadastro, o aluno utiliza a tela de login para acessar o sistema. Uma vez autenticado, ele pode explorar a tela de busca de vagas, onde é possível listar todas as vagas disponíveis. Essa interface inclui filtros para facilitar a busca por oportunidades específicas. O aluno também pode se candidatar a uma vaga diretamente através do sistema. Além disso, há a opção de acessar a tela de perfil para visualizar e alterar informações pessoais, garantindo que os dados estejam sempre atualizados. O sistema utiliza validações em tempo real para assegurar que as informações sejam inseridas corretamente em todas as etapas.

### **Figura 05** — Diagrama de Contexto – Visão do Empresa

### Fonte:Elaborado pelos autores

Segundo a figura 05, a empresa pode realizar diversas ações dentro do sistema. Inicialmente, a empresa acessa a tela de cadastro, onde pode criar uma nova conta fornecendo informações essenciais. Após o cadastro, a empresa utiliza a tela de login para acessar suas funcionalidades. Uma vez autenticada, a empresa pode cadastrar novas vagas, acessando uma interface específica para inserir detalhes das oportunidades oferecidas. Além disso, há a possibilidade de listar vagas já cadastradas, permitindo uma visão geral e controle sobre as mesmas. A empresa também pode editar ou excluir vagas conforme necessário. Outra funcionalidade importante é a capacidade de alterar dados da empresa, garantindo que as informações estejam sempre atualizadas. Todas essas ações incluem validações em tempo real para assegurar que os dados sejam inseridos corretamente e que as operações sejam realizadas de forma eficiente.

**Figura 06** — Diagrama de Contexto – Visão do Administrador



Fonte: Elaborado pelos autores.

Pertinente a figura 06, detalha-se as funções disponíveis para um administrador no sistema, abrangendo a gestão de vagas, cursos, empresas e usuários. Após acessar o sistema pela tela de login, o administrador pode listar, cadastrar, alterar e excluir vagas e cursos, garantindo que as informações estejam sempre atualizadas. Além disso, ele pode gerenciar empresas, incluindo a listagem e edição de dados empresariais. A administração de usuários é completa, permitindo ações como cadastro, alteração e exclusão de perfis, além de ajustes em matrículas. O administrador também tem acesso a um perfil personalizado e a um dashboard, que oferece uma visão geral das operações. Essas funcionalidades, junto com a capacidade de gerar relatórios, asseguram que o administrador tenha controle total sobre o sistema, promovendo uma gestão eficaz e organizada.

**Figura 07** — Diagrama de Contexto – Visão geral do sistema

Fonte: Elaborado pelos autores.Diagrama

Descrição gerada automaticamente

Coerente a figura 07, mostra-se as funcionalidades de listagem disponíveis no sistema. Ele permite que usuários acessem listas detalhadas de diversos elementos, como empresas, vagas e alunos, facilitando a gestão e visualização de dados. Além disso, o sistema oferece listagens de coordenadores de estágio e curso, supervisores de estágio e contratos relacionados, garantindo um controle abrangente sobre informações acadêmicas e administrativas. Listas de administradores, usuários, cursos e documentos também estão disponíveis, proporcionando uma visão completa e organizada de todos os componentes essenciais dentro do sistema. Essas funcionalidades visam simplificar o acesso a informações críticas, tornando a gestão mais eficiente e prática

**Figura 08** — Diagrama de Contexto – Visão geral do sistema

Diagrama

Descrição gerada automaticamente

Fonte: Elaborado pelos autores.

Relativo a figura 08, ilustra as funções relacionadas à gestão de alunos no sistema. O administrador tem a capacidade de cadastrar novos alunos, inserindo informações essenciais para que sejam registrados corretamente. Após o cadastro, o sistema permite a listagem dos alunos, facilitando o acesso às informações e garantindo que os dados estejam organizados e facilmente acessíveis. Essa estrutura simplifica o gerenciamento dos registros estudantis, permitindo que administradores mantenham um controle eficiente e atualizado sobre o corpo discente.

* 1. DIAGRAMA DE CASOS DE USO INDIVIDUAIS

Conforme destacado na obra "Engenharia de Software" de Ian Sommerville, os diagramas de casos de uso individuais são fundamentais para detalhar interações específicas entre usuários e o sistema. Eles oferecem uma visão focada em casos de uso particulares, permitindo uma compreensão mais clara de funcionalidades específicas. Esses diagramas são essenciais durante a análise de requisitos, ajudando a esclarecer como cada parte do sistema deve responder a interações distintas. Além disso, servem como base para o desenvolvimento de outros diagramas de design e implementação.

* + 1. – Diagrama de Contexto – Consultar Aluno.

O administrador pode consultar dados de alunos no sistema, obtendo informações detalhadas e uma visão geral. A função de consulta facilita o acesso rápido e organizado aos dados estudantis.

**Figura 09** — Diagrama de Contexto – Consultar Aluno

Diagrama

Descrição gerada automaticamente

Fonte: Elaborado pelos autores.

Quadro 9 – Documentação do caso de uso Consultar Aluno

|  |  |
| --- | --- |
| Documentação | |
| Nome do caso de uso | Consultar Aluno |
| Ator principal | Usuário (Administrador) |
| Resumo | Esse caso de uso descreve as ações necessárias para que o Administrador consulte as informações do aluno |
| Pré-condições | Aluno cadastradas |
| Pós-condições | O usuário é direcionado a parte desejada do sistema |
| Fluxo normal | |
| Ações do ator | Ações do sistema |
| 1. Usuário solicita alunos cadastrados |  |
|  | 1. Sistema recebe solicitação e mostra tela em lista os alunos cadastrados |
| 1. Usuário seleciona aluno que deseja saber mais detalhes |  |
|  | 1. Sistema valida solicitação |
|  | 1. Sistema carrega tela de consulta. |
| Fluxo alternativo | |
|  | 4. Sistema lista solicitação |
|  | 4.1 Sistema não consegue conectar com o banco |
|  | 4.1.2 Sistema envia mensagem “Erro com o banco de dados” |
|  | 4.1.3 Sistema retorna ao fluxo normal |

Fonte: Elaborado pelos autores.

* + 1. – Diagrama de Contexto – Alterar Aluno

Relacionado a figura 10, ilustra que o administrador pode alterar dados de alunos no sistema. Ao utilizar a função de alteração, ele atualiza informações específicas, como detalhes do curso. Essa ação está integrada à listagem de alunos, permitindo uma visão geral atualizada. O sistema facilita essas modificações, garantindo que os dados sejam precisos e organizados.

**Figura 10** — Diagrama de Contexto – Alterar Aluno

Diagrama

Descrição gerada automaticamente

Fonte: Elaborado pelos autores.

Quadro 10 — Documentação do caso de uso Alterar Aluno.

|  |  |
| --- | --- |
| Documentação | |
| Nome do caso de uso | Alterar Aluno |
| Ator principal | Usuário (Administrador) |
| Resumo | Esse caso de uso descreve as ações necessárias para que o Administrador altere as informações do aluno |
| Pré-condições | Aluno já cadastrado |
| Pós-condições | O usuário é direcionado a parte desejada do sistema |
| Fluxo normal | |
| Ações do ator | Ações do sistema |
| 1. Usuário solicita alterar um aluno |  |
|  | 1. Sistema recebe solicitação e mostra a tela com os dados já cadastrados |
| 1. Após alterar os dados do aluno o usuário aperta no botão “Salvar alterações” |  |
|  | 1. Sistema valida solicitação e altera dados. |
|  | 1. Sistema envia mensagem “Alterado com sucesso” |
|  | 1. Sistema retorna para a tela com a tabela de listagem. |
| Fluxo alternativo | |
|  | 4. Sistema valida solicitação e altera dados. |
|  | 4.1 Sistema falha ao se comunicar com a controller |
|  | 4.1.1 Sistema envia a mensagem de erro |
|  | 4.1.2 Sistema retorna ao fluxo normal pelo item 2 |

Fonte: Elaborado pelos autores.

* + 1. – Diagrama de Contexto – Excluir Aluno

Com base na figura 11, pode se ver administrador tem a capacidade de excluir alunos do sistema. Ao iniciar a exclusão, ele remove o aluno selecionado da listagem geral. Essa ação está vinculada à listagem de alunos, garantindo que a informação seja atualizada imediatamente. O sistema assegura que a exclusão seja feita de forma eficiente e organizada, mantendo os dados precisos.

**Figura 11** — Diagrama de Contexto – Excluir Aluno

Diagrama

Descrição gerada automaticamente

Fonte: Elaborado pelos autores.

Quadro 11 — Documentação do caso de uso Excluir Aluno.

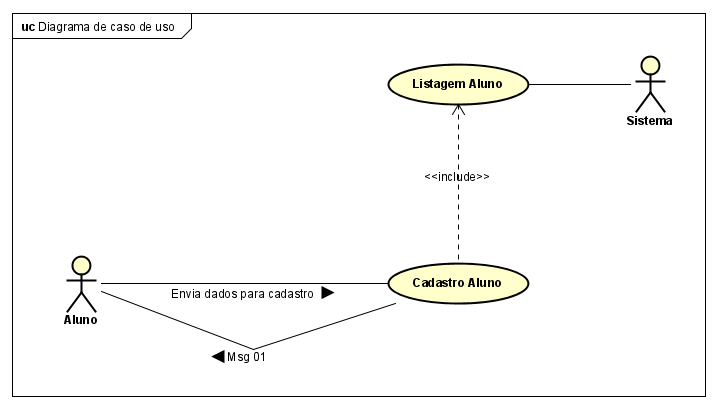
|  |  |
| --- | --- |
| Documentação | |
| Nome do caso de uso | Excluir Aluno |
| Ator principal | Usuário (Administrador) |
| Resumo | Esse caso de uso descreve as ações necessárias para que o Administrador apague as informações do aluno |
| Pré-condições | Aluno excluido |
| Pós-condições | O usuário é direcionado a parte desejada do sistema |
| Fluxo normal | |
| Ações do ator | Ações do sistema |
| 1.Usuário solicita excluir um aluno |  |
|  | 2.Sistema abre um dropdown com o botão “Excluir” |
| 3.Usuário aperta o botão “Excluir” |  |
|  | 4.Sistema valida solicitação |
|  | 5.Sistema volta para a tela de listar alunos |
| Fluxo alternativo | |
|  | 4.Sistema lista solicitação |
|  | 4.1Sistema não consegue conectar com o banco |
|  | 4.1.1Sistema envia mensagem “Erro com o banco de dados” |
|  | 4.1.2Sistema retorna ao fluxo normal |

Fonte: Elaborado pelos autores.

* + 1. – Diagrama de Contexto – Cadastrar Aluno

Nesta figura, mostra-se que o administrador possui a função de cadastrar alunos no sistema, registrando suas informações para garantir sua inclusão na listagem geral.

**Figura 12** — Diagrama de Contexto – Cadastrar Aluno



Quadro 12 – Documentação do caso de uso Cadastrar Aluno

|  |  |
| --- | --- |
| Documentação | |
| Nome do caso de uso | Alterar Aluno |
| Ator principal | Usuário (Administrador) |
| Resumo | Esse caso de uso descreve as ações necessárias para que (Administrador) adicione novos alunos |
| Pré-condições | Preencher todos os campos para concluir o cadastro do aluno |
| Pós-condições | O usuário é direcionado a parte desejada do sistema |
| Fluxo normal | |
| Ações do ator | Ações do sistema |
| 1. Usuário solicita o cadastro de um novo aluno |  |
|  | 2. Sistema recebe solicitação e mostra tela com formulário para cadastro |
| 3. Usuário informa dados do aluno e clica no botão “Cadastrar” |  |
|  | 4. Sistema valida os campos adicionados |
|  | 5. Sistema retorna mensagem “Aluno cadastrado com sucesso” |
| Fluxo alternativo | |
|  | 4. Sistema valida campos e dados |
|  | 4.1 Sistema envia mensagem “Dados inválidos” |
|  | 4.1.1 Sistema envia a mensagem de erro |
|  | 4.1.2 Sistema retorna ao fluxo normal pelo item 2 |

Fonte: Elaborado pelos autores.

* + 1. – Diagrama de Contexto – Cadastrar Curso

Na figura 13, mostra-se que o administrador possui a função de cadastrar cursos, que são listados pelo sistema após o cadastro, garantindo que as informações sejam atualizadas imediatamente.

**Figura 13** — Diagrama de Contexto – Cadastrar Curso

Diagrama

Descrição gerada automaticamente

Fonte: Elaborado pelos autores.

Quadro 13 — Documentação do caso de uso Cadastrar Curso.

|  |  |
| --- | --- |
| Documentação | |
| Nome do caso de uso | Cadastrar Curso |
| Ator principal | Usuário (Administrador) |
| Resumo | Esse caso de uso descreve as ações necessárias para que o Administrador cadastre as informações do curso |

|  |  |
| --- | --- |
| **Documentação** | |
| **Nome do caso de uso** | **Cadastrar Curso** |
| **Ator principal** | Usuário (Administrador) |
| **Resumo** | Esse caso de uso descreve as ações necessárias para que (Administrador) adicione novos cursos |
| **Pré-condições** | Preencher todos os campos para concluir o cadastro do curso |
| **Pós-condições** | O usuário é direcionado a parte desejada do sistema |
| **Fluxo normal** | |
| **Ações do ator** | **Ações do sistema** |
| 1. Usuário solicita o cadastro de um novo curso |  |
|  | 1. Sistema recebe solicitação e mostra tela com formulário para cadastro |
| 1. Usuário informa dados do curso e clica no botão “Cadastrar” |  |
|  | 1. Sistema valida os campos adicionados |
|  | 1. Sistema retorna mensagem “Curso cadastrado com sucesso” |
| **Fluxo alternativo** | |
|  | 1. Sistema valida campos e dados |
|  | * + 1. Sistema identifica dados diferente do esperado ou em branco |
|  | * + 1. Sistema envia mensagem “Dados inválidos” |
|  | * + 1. Sistema retorna ao fluxo normal pelo item 7 |

* + 1. – Diagrama de Contexto – Consultar Curso

Segundo a figura 10, mostra-se como o administrador interage com o sistema para consultar cursos. Primeiro, ele acessa a função de consulta para ver detalhes específicos. Em seguida, há uma listagem que oferece uma visão geral dos cursos, facilitando o acesso às informações de forma organizada.

**Figura 10** — Diagrama de Contexto – Consultar Curso

Diagrama

Descrição gerada automaticamente

Fonte: Elaborado pelos autores.

Quadro 14 — Documentação do caso de uso Consultar Curso.

|  |  |
| --- | --- |
| Documentação | |
| Nome do caso de uso | Consultar Curso |
| Ator principal | Usuário (Administrador) |
| Resumo | Esse caso de uso descreve as ações necessárias para que o Administrador consulte as informações do curso |
| Pré-condições | Curso cadastrados |
| Pós-condições | O usuário é direcionado a parte desejada do sistema |
| Fluxo normal | |
| Ações do ator | Ações do sistema |
| 1. Usuário solicita cursos cadastrados |  |
|  | 2. Sistema recebe solicitação e mostra tela em lista os cursos cadastrados |
| 3. Usuário seleciona curso que deseja saber mais detalhes |  |
|  | 4. Sistema valida solicitação |
|  | 5. Sistema carrega tela de consulta. |
| Fluxo alternativo | |
|  | 4.Sistema lista solicitação |
|  | 4.1 Sistema não consegue conectar com o banco |
|  | 4.1.1 Sistema envia mensagem “Erro com o banco de dados” |
|  | 4.1.2 Sistema retorna ao fluxo normal |

Fonte: Elaborado pelos autores.

* + 1. – Diagrama de Contexto – Alterar Curso

Conforme a figura 11, o diagrama mostra como o administrador interage com o sistema para alterar cursos. O administrador envia novos dados para a função "Altera Curso", que atualiza as informações. A listagem de cursos é automaticamente ajustada, garantindo que as alterações sejam refletidas de forma organizada no sistema. Esse processo assegura que o administrador mantenha o controle eficiente das informações de curso.

**Figura 11** — Diagrama de Contexto – Alterar Curso

Diagrama

Descrição gerada automaticamente

Fonte: Elaborado pelos autores.

Quadro 15 – Documentação do caso de uso Alterar Curso

|  |  |
| --- | --- |
| Documentação | |
| Nome do caso de uso | Alterar Curso |
| Ator principal | Usuário (Administrador) |
| Resumo | Esse caso de uso descreve as ações necessárias para que o Administrador altere as informações do curso |
| Pré-condições | Curso já cadastrado |
| Pós-condições | O usuário é direcionado a parte desejada do sistema |
| Fluxo normal | |
| Ações do ator | Ações do sistema |
| 1. Usuário solicita alterar um curso |  |
|  | 2. Sistema recebe solicitação e mostra a tela com os dados já cadastrados |
| 3. Após alterar os dados do curso o usuário aperta no botão “Salvar alterações” |  |
|  | 4. Sistema valida solicitação e altera dados. |
|  | 5. Sistema envia mensagem “Alterado com sucesso” |
|  | 6. Sistema retorna para a tela com a tabela de listagem. |
| Fluxo alternativo | |
|  | 4. Sistema valida solicitação e altera dados. |
|  | 4.1 Sistema falha ao se comunicar com a controller |
|  | 4.1.1 Sistema envia a mensagem de erro |
|  | 4.1.2 Sistema retorna ao fluxo normal pelo item 10 |

Fonte: Elaborado pelos autores.

* + 1. – Diagrama de Contexto – Excluir Curso

Conforme a figura 12, o administrador interage com o sistema para excluir cursos. Ele envia a solicitação de exclusão através da função "Excluir Curso". A listagem de cursos é atualizada automaticamente, garantindo que as alterações sejam refletidas no sistema. Esse processo assegura um controle eficiente das informações de curso.

**Figura 12** — Diagrama de Contexto – Excluir Curso

Diagrama

Descrição gerada automaticamente

Fonte: Elaborado pelos autores.

Quadro 16 — Documentação do caso de uso Excluir Curso.

|  |  |
| --- | --- |
| Documentação | |
| Nome do caso de uso | Excluir Curso |
| Ator principal | Usuário (Administrador) |
| Resumo | Esse caso de uso descreve as ações necessárias para que o Administrador apague as informações do curso |
| Pré-condições | Curso já cadastrado |
| Pós-condições | O usuário é direcionado a parte desejada do sistema |
| Fluxo normal | |
| Ações do ator | Ações do sistema |
| 1. Usuário solicita excluir um curso |  |
|  | 2. Sistema abre um dropdown com o botão “Excluir” |
| 3. Usuário aperta o botão “Excluir” |  |
|  | 4. Sistema valida solicitação |
|  | 5. Sistema volta para a tela de listar cursos |
| Fluxo alternativo | |
|  | 4. Sistema lista solicitação |
|  | 4.1 Sistema não consegue conectar com o banco |
|  | 4.1.1Sistema envia mensagem “Erro com o banco de dados” |
|  | 4.1.2Sistema retorna ao fluxo normal |

Fonte: Elaborado pelos autores

3.7 - DIAGRAMA DE SEQUÊNCIA

O diagrama de sequência adota um modelo que ilustra o comportamento do usuário ao realizar ações dentro da aplicação, observando a ordem temporal das ações no desenvolvimento do processo. Seu propósito é identificar o evento que inicia o processo, o ator responsável por esse evento e a sequência de desenvolvimento que deve ocorrer. Em resumo, proporciona uma representação visual destacando a interação entre diferentes elementos da aplicação, facilitando a compreensão da lógica temporal e das responsabilidades associadas às ações realizadas (GUEDES, 2018).

### 3.7.1 **—** Diagrama de Sequência – Cadastro Aluno

Referente a figura 13, o diagrama apresenta-se a interação entre diferentes atores e sistemas ao longo do tempo, destacando a troca de mensagens entre eles. Ele ilustra o fluxo do processo de autenticação de um usuário, iniciando com a solicitação de login e a validação das credenciais pela API e o serviço de autenticação. Após a validação, o sistema retorna a resposta ao usuário, confirmando o sucesso ou falha da operação. Esse diagrama é útil para entender como os componentes se comunicam e o sequenciamento das operações, facilitando a análise de requisitos e o comportamento do sistema em diferentes cenários.

Diagrama

Descrição gerada automaticamente**Figura 13 —** Diagrama de Sequência – Cadastro Aluno

Fonte: Desenvolvidos pelos autores.

### 3.7.2 **—** Diagrama de Sequência – Altera Aluno

De acordo com a figura 14, ilustra-se o processo de alteração de dados de um aluno no sistema. O usuário inicia a solicitação para alterar as informações do aluno. A API recebe essa solicitação e, após validar os dados, se comunica com o serviço de autenticação para garantir que o usuário tem permissão para realizar a alteração. Em seguida, a API envia a atualização para o banco de dados, onde os novos dados do aluno são salvos. Após a conclusão, o sistema confirma a alteração e notifica o usuário de que a operação foi realizada com sucesso. O diagrama ajuda a entender o fluxo de informações e como os componentes do sistema interagem para completar a tarefa.

Gráfico de caixa estreita

Descrição gerada automaticamente**Figura 14 —** Diagrama de Sequência – Altera Aluno

Fonte: Desenvolvidos pelos autores.

### 3.7.3 **—** Diagrama de Sequência – Exclui Aluno

Referente a figura 15, mostra o processo de exclusão de um aluno no sistema. Primeiro, o usuário inicia a solicitação de exclusão, que é enviada pelo Front-end para a API. A API valida a solicitação e consulta o serviço de autenticação para verificar as permissões do usuário. Após a autenticação, a API envia a instrução para excluir os dados do aluno no banco de dados. Depois que a exclusão é confirmada, o sistema notifica o usuário de que o aluno foi removido com sucesso. O diagrama detalha como cada parte do sistema trabalha em conjunto para completar essa ação.

Diagrama

Descrição gerada automaticamente**Figura 15 —** Diagrama de Sequência – Exclui Aluno

Fonte: Desenvolvidos pelos autores.

3.7.4 **—** Diagrama de Sequência – Lista Aluno

De forma coerente a figura 16, demonstra-se o processo de listagem de alunos no sistema. O usuário solicita a visualização dos alunos, e essa requisição é processada, verificando-se as permissões necessárias através do serviço de autenticação. Após a autorização, o sistema acessa o banco de dados para recuperar a lista de alunos. Os dados são então enviados de volta para o usuário, que pode visualizá-los. O diagrama destaca como cada parte do sistema colabora para completar essa tarefa de forma eficiente.

Diagrama, Gráfico de caixa estreita

Descrição gerada automaticamente**Figura 16 —** Diagrama de Sequência – Lista Aluno

Fonte: Desenvolvidos pelos autores.

3.7.5 **—** Diagrama de Sequência – Cadastro Aluno

Associado a figura 17, ilustra-se o processo de cadastro de um curso no sistema. O usuário inicia a solicitação de cadastro, que é processada pelo sistema. Primeiramente, as informações do curso são verificadas quanto à validade. Após essa validação, o sistema garante que o usuário tenha as permissões necessárias para continuar. Em seguida, os dados são armazenados no banco de dados. Depois de concluído o cadastro, o sistema envia uma confirmação ao usuário, indicando que o curso foi registrado com sucesso. O diagrama detalha como os componentes interagem para executar essa tarefa.

Diagrama

Descrição gerada automaticamente**Figura 17** — Diagrama de Sequência – Cadastro Curso

Fonte: Desenvolvidos pelos autores.

3.7.6 **—** Diagrama de Sequência – Exclui Curso

Pertinente a figura 18, o processo de exclusão de um curso no sistema é iniciado pelo usuário. A solicitação é verificada para garantir que o usuário possua as permissões necessárias. Após a confirmação, o sistema remove os dados do curso do banco de dados. Uma vez concluída a exclusão, o sistema notifica o usuário sobre o sucesso da operação. O diagrama detalha a interação entre os componentes do sistema para realizar essa tarefa de forma eficiente.

Diagrama, Desenho técnico

Descrição gerada automaticamente**Figura 18 —** Diagrama de Sequência – Exclui Curso

Fonte: Desenvolvidos pelos autores.

3.7.8 **—** Diagrama de Sequência – Altera Curso

Baseado na figura 19 demonstra-se o processo de alteração de um curso, desde a solicitação do usuário até a confirmação da mudança. O usuário inicia o pedido na interface do sistema, que envia os dados para o serviço responsável por gerenciar cursos. O serviço verifica a autorização do token e, se tudo estiver correto, passa as informações para o repositório de cursos. O repositório, então, realiza a atualização no banco de dados. Após a alteração ser concluída, uma confirmação é enviada de volta, passando pelo serviço e retornando ao usuário, indicando que a operação foi bem-sucedida.

Diagrama

Descrição gerada automaticamente com confiança baixa**Figura 19 —** Diagrama de Sequência – Altera Curso

Fonte: Desenvolvidos pelos autores.

3.7.9 **—** Diagrama de Sequência – Lista Curso

Concordante a figura 20, mostra-se o processo de listagem dos cursos disponíveis, desde a solicitação do usuário até a exibição dos dados. O usuário inicia o pedido na interface do sistema, que encaminha a solicitação para o serviço de cursos. O serviço de cursos verifica a autenticação e, se estiver tudo correto, consulta o repositório de cursos. O repositório, por sua vez, busca os dados no banco de dados e retorna a lista de cursos para o serviço. Por fim, a resposta é encaminhada de volta à interface do usuário, exibindo as informações solicitadas.

Diagrama

Descrição gerada automaticamente**Figura 20 —** Diagrama de Sequência – Lista Curso

Fonte: Desenvolvidos pelos autores.

# 4 DEFINIÇÃO DA INTERFACE COM O USUÁRIO (UX)

User Experience (UX), ou Experiência do Usuário, é um conceito central no design de sistemas interativos, focando na qualidade da interação entre o usuário e o produto. A UX envolve todos os aspectos dessa interação, desde a usabilidade e acessibilidade do sistema até a satisfação do usuário ao utilizá-lo. Em outras palavras, UX não é apenas sobre se um sistema é funcional, mas também sobre como ele é percebido e sentido pelo usuário.

De acordo com Sommerville (2011), a experiência do usuário vai além da simples funcionalidade de um sistema, abrangendo fatores como eficácia, eficiência e satisfação durante sua utilização. Ele destaca que a usabilidade — um dos componentes centrais da UX — está diretamente relacionada à facilidade com que os usuários podem aprender a usar um sistema, o quão eficiente ele é em suportar suas tarefas, e a satisfação que proporciona ao longo do uso.

4.1 DESCRIÇÃO DE CENÁRIO

Conforme escrito no livro "O Guia para Projetar UX" de Russ Unger e Carolyn Chandler, os cenários representam narrativas detalhadas que ilustram as interações entre um usuário e um sistema. Estas histórias oferecem uma compreensão abrangente ao contextualizar a persona, seus objetivos, as ações específicas em busca desses objetivos, os desafios enfrentados e os desfechos das interações. Os cenários de persona são ferramentas fundamentais no direcionamento do design de experiência do usuário, permitindo uma avaliação minuciosa da eficácia do sistema em diferentes circunstâncias.

Figura 21 – Imagem do primeiro cenário



Fonte: Elaborado pelos autores.

**Figura 22** – Imagem do segundo cenário

Fonte: Elaborado pelos autores.

4.2 DESCRIÇÃO DE PERSONAS

De acordo com "O Guia para Projetar UX" de Russ Unger e Carolyn Chandler, personas constituem representações fictícias de usuários envolvidos na interação com um sistema específico. Seu desenvolvimento é direcionado à compreensão aprofundada das necessidades, comportamentos e motivações desses usuários. A criação de personas visa proporcionar aos designers e desenvolvedores uma visão mais humanizada e centrada no usuário, possibilitando a concepção de soluções que correspondam de maneira eficaz às expectativas e requisitos dos usuários finais.

**Figura 23** – Imagem da segunda persona

Fonte: Elaborado pelos autores.

**Figura 24** – Imagem da segunda persona

Fonte: Elaborado pelos autores.

4.3 ESBOÇOS DE TELA (WIREFRAMES)

Como afirma Don Norman, autor de "The Design of Everyday Things", "protótipos são uma ferramenta fundamental para explorar possíveis soluções de design antes da implementação completa". Os protótipos de tela, são representações tangíveis e interativas da proposta de design, permitindo a exploração, comunicação e refinamento de ideias".

Assim, os modelos iniciais de tela não só promovem a comunicação efetiva entre as equipes de design e desenvolvimento, como também têm um papel crucial na detecção antecipada de adaptações necessárias, resultando em economia de tempo e recursos.

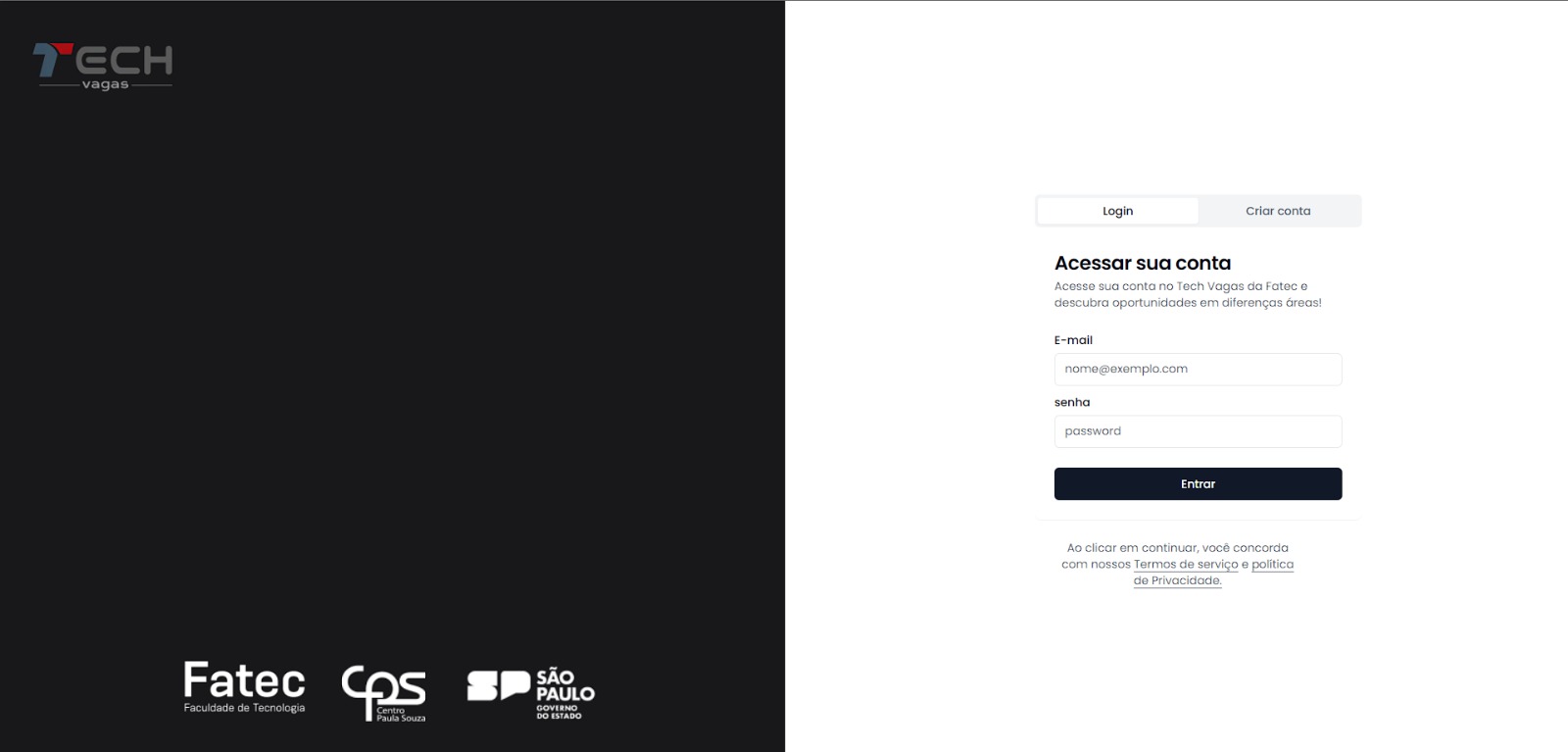
**Figura 25** – Tela de cadastro do usuário



Fonte: Elaborado pelos autores.

De acordo com a figura 25, tela de cadastro de usuário, o visitante pode criar uma nova conta ao fornecer informações essenciais, como nome, e-mail e senha. Essa interface geralmente inclui validações em tempo real para garantir que os dados sejam inseridos corretamente, além de opções para aceitar os termos de uso e políticas de privacidade.

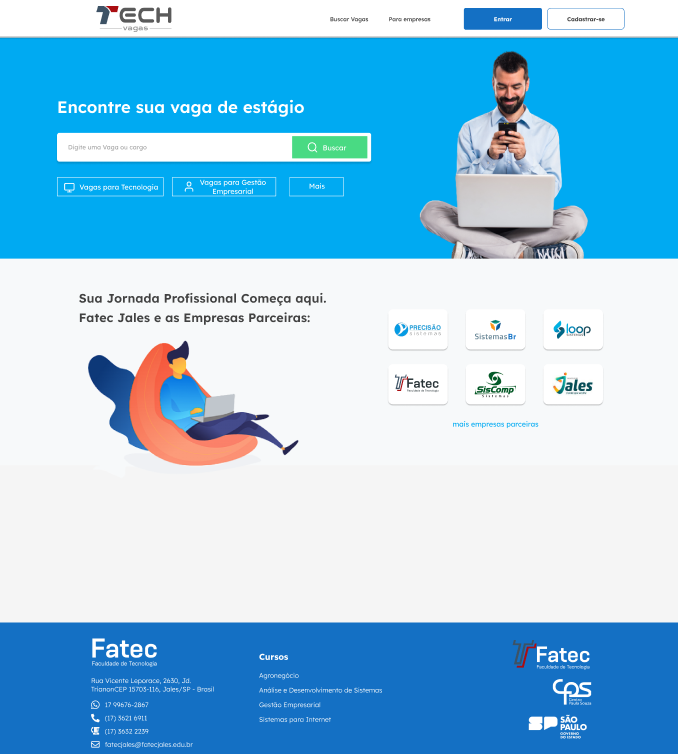
**Figura 26** – Tela de login do usuário



Fonte: Elaborado pelos autores.

Relacionado a figura 26, tela de login do usuário possibilita o visitante pode inserir suas credenciais, como nome de usuário e senha, para acessar sua conta de forma segura. Além disso, há opções para recuperar a senha em caso de esquecimento e, muitas vezes, um recurso que permite lembrar as informações de login em dispositivos confiáveis. A interface geralmente inclui feedback visual, informando ao usuário sobre possíveis erros de autenticação.

**Figura 27** – Tela inicial



Fonte: Elaborado pelos autores.

Referente a figura 27, a tela inicial do sistema é projetada para ser intuitiva e acolhedora, incentivando os alunos a explorar oportunidades de estágio de forma eficiente. No centro, há uma barra de pesquisa que permite aos usuários procurar vagas específicas, acompanhada de botões de acesso rápido para categorias como “Tecnologia” e “Gestão Empresarial”. Essa organização facilita a navegação e ajuda os alunos a encontrar rapidamente oportunidades alinhadas aos seus interesses. A seção de parcerias destaca a colaboração com empresas locais, reforçando a conexão entre a academia e o mercado de trabalho.

**Figura 28** – Tela de empresas cadastradas

Interface gráfica do usuário, Aplicativo, Email

Descrição gerada automaticamente

Fonte: Elaborado pelos autores.

Na figura 28, tela de cadastro de empresas, faz com que o sistema permite que organizações parceiras criem uma conta fornecendo informações importantes, como nome da empresa, CNPJ, e-mail e telefone de contato. A interface é projetada para ser clara e eficiente, incorporando validações em tempo real para garantir que os dados sejam inseridos corretamente. Além disso, há opções para que as empresas aceitem os termos de uso e políticas de privacidade, assegurando conformidade legal e transparência. Após o cadastro, a empresa pode cadastrar, alterar ou excluir vagas de estágio, inserindo detalhes como descrição da vaga, requisitos, local de trabalho e benefícios oferecidos. Esse processo não só facilita a entrada de empresas na plataforma, mas também fortalece a rede de parcerias, criando um ecossistema robusto de oportunidades de estágio para os alunos.

4.4 PROTÓTIPOS DE TELA

O termo protótipo vem do grego PRÓTOS (primeiro) e TYPOS (tipo), no momento em que traduzimos para o sentido literal é a mesma coisa que “PRIMEIRO TIPO” OU “PRIMEIRO MODELO”. Protótipo em si já diz respeito a muita coisa quando falamos sobre projetar ou planejar alguma coisa, seja uma viagem em família por exemplo que necessita de um “plano” de viagem ou até mesmo algum projeto em que se está sendo trabalhado seja na escola, trabalho até mesmo em faculdades precisa-se de um protótipo para a tarefa ser bem efetuada e tenha ótimos desempenho.

"Prototipar é uma maneira de experimentar e aprender. Não é uma simples representação, mas sim um meio de entender melhor as necessidades dos usuários." (BROWN, 2010, p. 95).

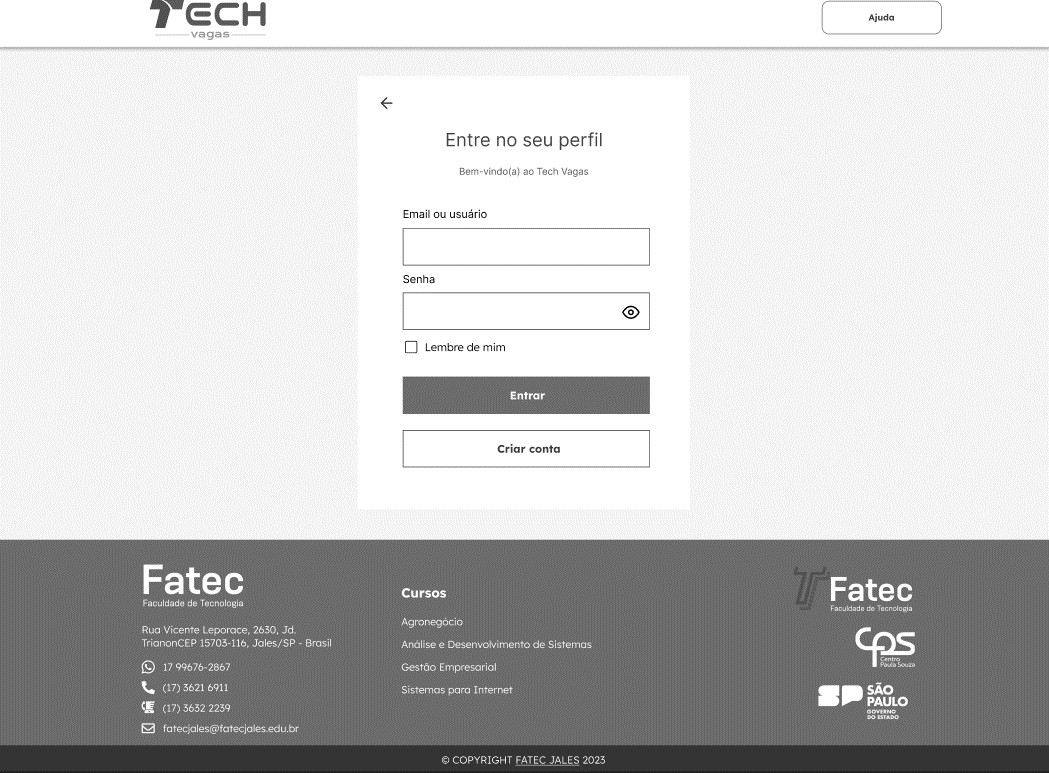
A prototipagem centrada no usuário é um verdadeiro diálogo entre criadores e usuários, onde a empatia se torna a chave para o sucesso. Imagine um designer não apenas como alguém que cria, mas como um investigador que busca entender as histórias e sentimentos das pessoas que usarão seu produto. Ao criar protótipos, eles podem testar ideias em um ambiente real, ouvindo atentamente o que os usuários têm a dizer e fazendo ajustes baseados nessas interações. Esse processo é mais do que funcionalidade; é sobre construir conexões emocionais, garantindo que cada detalhe do design ressoe com as necessidades e desejos de quem o utiliza. Quando os usuários se tornam co-criadores, o resultado é uma solução que não só resolve problemas, mas que também faz as pessoas se sentirem valorizadas e compreendidas. Essa abordagem transforma a inovação em uma experiência compartilhada, onde cada feedback é uma oportunidade de crescimento. Portanto, a prototipagem vai além de um simples passo no design; é um compromisso com a criação de experiências significativas e humanas.

# 4.4.2 Demonstração dos Protótipos das Telas Baseadas nos Wireframes

Figura 1 – Protótipo Tela de Cadastro



Esta seria a Tela Inicial do sistema “TECHVAGAS”, porém após várias reuniões e debates o grupo que está responsabilizado por este projeto ao consentimento de todos os integrantes, decidimos mudar para uma tela mais versátil e de fácil acesso para os usuários.

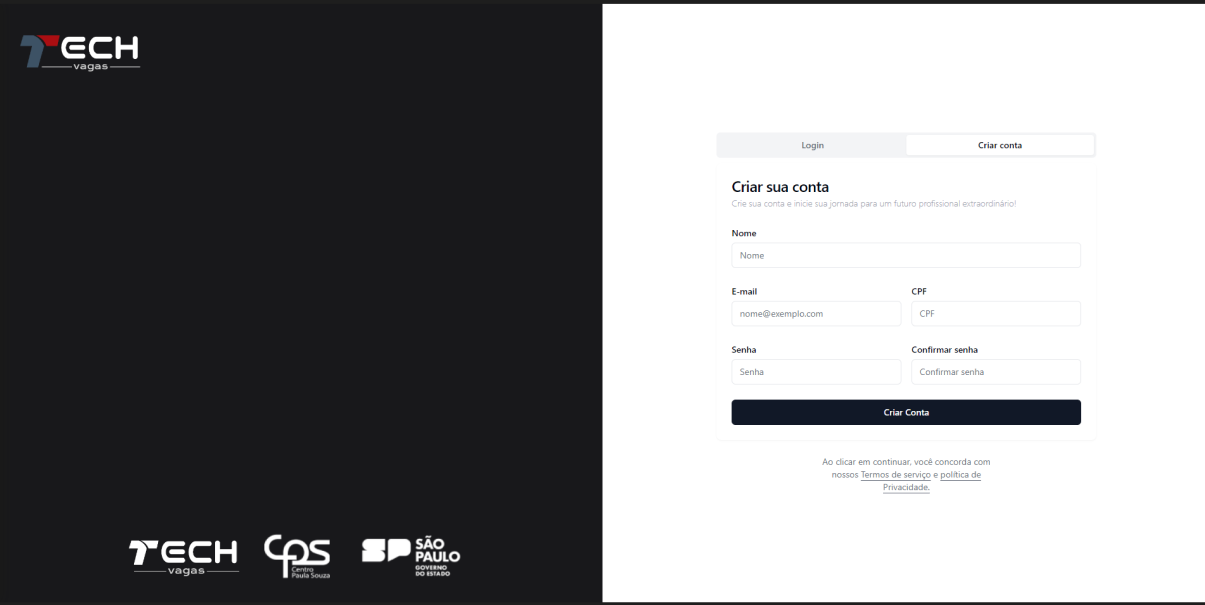


Agora partindo para uma outra abordagem, temos a tela de quando já se está cadastrado. Acontecerá igual a tela de Cadastro citado a cima, iremos substituir por uma tela mais versátil de fácil acessibilidade para os usuários.

* 4.4.3 Discussão das telas do TECHVAGAS

As telas do sistema TECHVAGAS foram projetadas com foco em proporcionar uma experiência fluida e acessível, tanto para os estudantes quanto para as empresas que utilizam a plataforma. Ao desenvolver a interface, foram considerados aspectos cruciais de usabilidade, acessibilidade, componentização de interface, e padronização visual, garantindo que a navegação seja intuitiva e consistente em todos os dispositivos, desde desktops até smartphones. Cada tela foi pensada para atender aos requisitos funcionais e não funcionais do sistema, assegurando que as interações dos usuários sejam realizadas de forma eficiente e sem frustrações.

* Tela de cadastro



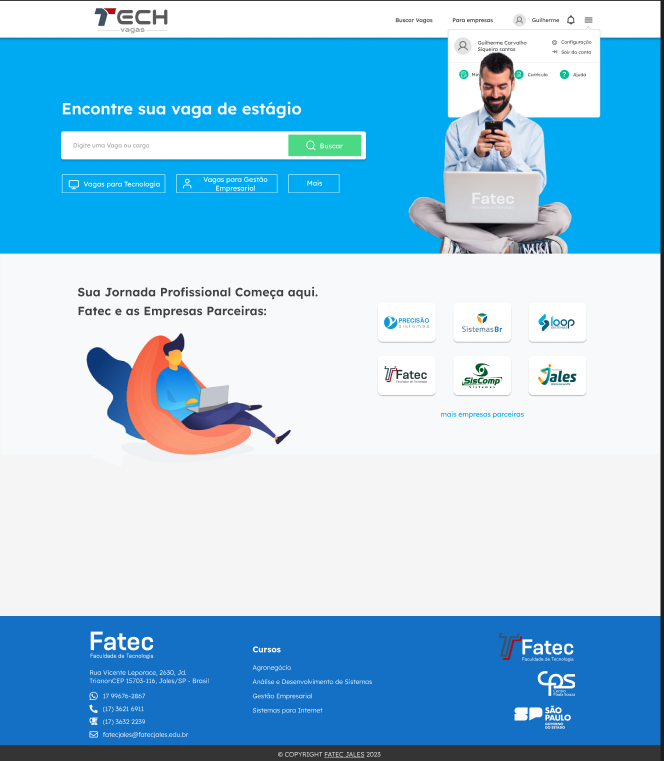
A tela de cadastro de usuário é a primeira interação que os alunos e as empresas terão com o sistema. Seu design ocorreu intencionalmente simplificado, focando em uma estrutura clara com campos básicos de preenchimento, como nome, e-mail e senha. Os campos possuem **validação em tempo real**, o que ajuda a reduzir erros durante o cadastro, notificando imediatamente o usuário sobre campos incorretos ou ausentes.

* **Feedback Visual**: O sistema utiliza feedback visual claro (em tons de verde para confirmações e vermelho para erros), o que melhora a experiência do usuário, permitindo correções rápidas.
* **Segurança**: Durante o processo de cadastro, são utilizadas práticas de segurança como **hashing** de senhas, garantindo que os dados dos usuários estejam protegidos.
* Tela de Login



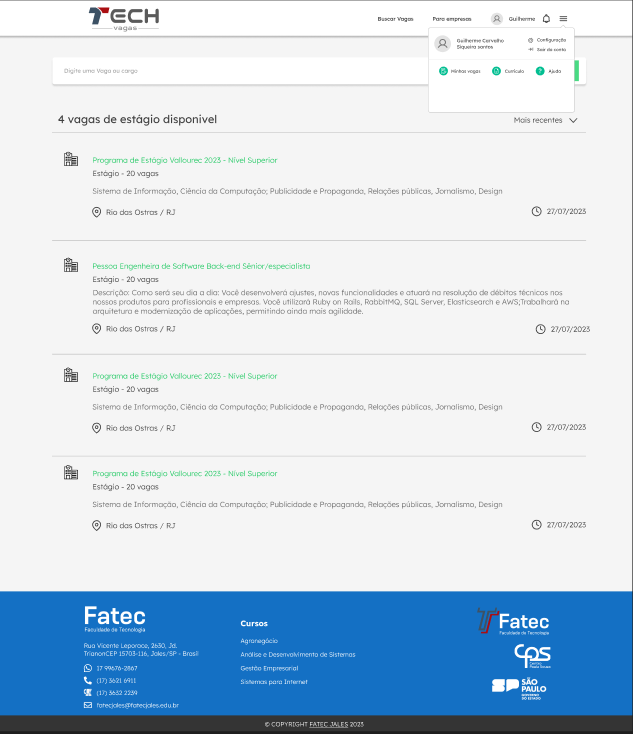
A tela de login foi projetada com foco em segurança e usabilidade. Ela permite que o usuário insira suas credenciais de forma rápida e segura. A implementação de recursos como **autenticação multifatorial (MFA)** está planejada para versões futuras, aumentando a segurança da aplicação.

* **Recuperação de senha**: Há um processo robusto de recuperação de senha, utilizando e-mails de verificação e validação de identidade. Isso reduz o risco de acesso não autorizado.
* **Design Responsivo**: A interface foi desenhada para se adaptar a diferentes tamanhos de tela, o que é essencial para o uso em dispositivos móveis.
* Tela Inicial



A tela inicial foi criada com uma abordagem minimalista, exibindo uma barra de busca centralizada para facilitar a navegação dos usuários pelas oportunidades de estágio. Ela também contém atalhos para áreas mais acessadas, como **vagas recomendadas**, **vagas por categorias** (ex.: Tecnologia, Gestão Empresarial), e as **empresas parceiras**.

* **Hierarquia Visual**: As seções são claramente divididas, utilizando **tamanhos de fonte diferenciados** para criar uma hierarquia de importância. Isso melhora a compreensão da interface e facilita a localização rápida das informações desejadas.
* Tela de Vagas Cadastradas



A tela de **Vagas Cadastradas** permite que os usuários, tanto empresas quanto administradores, visualizem e gerenciem as vagas disponíveis na plataforma. O layout foi projetado para ser eficiente, exibindo as vagas em um formato **tabular**, onde os campos principais são título da vaga, descrição, requisitos e status (ativa, fechada, em processo). Existem filtros disponíveis para refinar a busca, como **data de publicação**, **área de atuação**, e **número de candidatos inscritos**.

* **Interatividade**: A interação é dinâmica, com recursos de **edição in-line** que permitem a atualização rápida de informações, como a alteração da descrição ou requisitos das vagas, sem precisar recarregar a página.
* **Gestão de Vagas**: A interface também facilita ações rápidas, como **ativar** ou **desativar vagas**, além de oferecer a possibilidade de clonar uma vaga para publicação futura com pequenas modificações.

4.5 ACESSIBILIDADE

**Acessibilidade** é um conceito que envolve a criação de ambientes, produtos, serviços e tecnologias que sejam utilizáveis por todas as pessoas, independentemente de suas limitações físicas, sensoriais, intelectuais ou motoras. A ideia é garantir que qualquer pessoa, com ou sem deficiência, tenha acesso e consiga usufruir das mesmas oportunidades, sejam elas educacionais, profissionais ou culturais. Em um contexto mais amplo, a acessibilidade não se limita apenas a deficiências, mas inclui também aspectos como mobilidade reduzida, idade avançada ou mesmo condições temporárias que podem prejudicar o acesso a certos recursos.

**Richard J. L. Hodge**, especialista em políticas públicas, afirma: "A acessibilidade não é apenas sobre atender às necessidades básicas das pessoas com deficiência, mas sobre criar uma sociedade onde todos possam participar plenamente."

A **Lei Brasileira de Inclusão** (Lei nº 13.146/2015) reforça a importância da acessibilidade no Brasil, obrigando que espaços públicos e privados se adequem para atender a pessoas com deficiência. A acessibilidade está presente em várias áreas, como transporte, comunicação, internet e no próprio ambiente físico, sendo um direito fundamental para promover a igualdade de oportunidades.

# 5 BANCO DE DADOS

Um banco de dados é um sistema projetado para armazenar e organizar informações de maneira que facilite o acesso e a manipulação dos dados. O banco de dados permite que os usuarios gerenciem grandes volumes de informação de forma estruturada, assim possibilitando a realização de consultas, relatórios e analíse que sao fundamentais em diversos setores, alguns dos exemplo são: comércio, saúde e pesquisas. Existem variados tipos de bancos de dados tais como: relacionais, nao-relacionais e cada um com suas características e aplicações distintas.

Segundo o autor C. J. Date e com base no livro An Introduction to Database Systems “Um banco de dados é um conjunto de dados que é armazenado de maneira organizada e que pode ser acessado e manipulado com eficiência. A escolha do modelo de dados e das estruturas de armazenamento adequadas é crucial para o desempenho e a integridade do sistema" (DATE, C. J.2004).

Um banco de dados é uma estrutura essencial para a organização e a gestão de informações, permitindo que dados sejam armazenados de forma eficiente e acessados rapidamente. Segundo *Elmasri e Navathe*, "um sistema de banco de dados é um conjunto de programas que permitem a definição, construção e manipulação de bancos de dados para atender a uma variedade de aplicações" (ELMASRI, R.; NAVATHE, 2016). A capacidade de um banco de dados de lidar com grandes volumes de dados e garantir integridade e segurança é crucial em ambientes corporativos e acadêmicos, onde decisões importantes dependem da precisão e da acessibilidade das informações. Assim, o design e a implementação eficazes de bancos de dados se tornam uma prioridade, impactando diretamente o sucesso de projetos em diversas áreas.

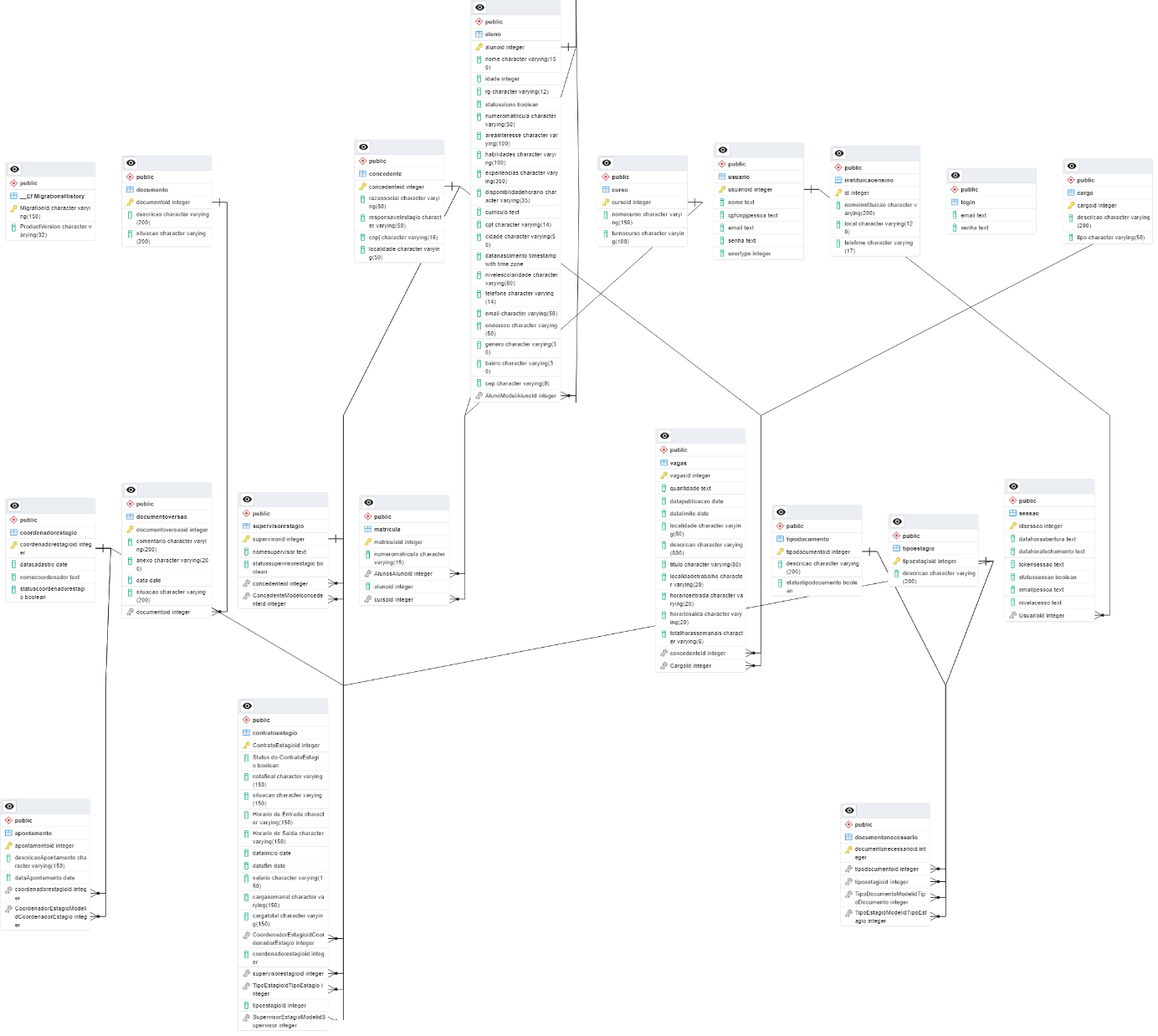
### 5.1 Modelo Entidade Relacionamento

O Modelo de Entidade-Relacionamento (MER) é uma técnica fundamental no design de bancos de dados que visa representar graficamente as entidades e os relacionamentos entre elas. Ele permite que os analistas identifiquem e organizem os dados de maneira que reflitam as necessidades do negócio, facilitando a comunicação entre as partes interessadas. Como destaca (MARTIN, J,1988) "O MER proporciona uma visão clara e concisa da estrutura do banco de dados, ajudando a garantir que todos os aspectos dos requisitos de informação sejam considerados"

Reforçando essa visão, Ramez Elmasri e Shamkant Navathe enfatizam que "O modelo de entidade-relacionamento é essencial para a representação do mundo real em um banco de dados, permitindo uma modelagem lógica que é tanto intuitiva quanto eficiente" (ELMASRI, R.; NAVATHE, 2016). Essa representação visual não apenas simplifica a compreensão dos dados, mas também orienta o desenvolvimento de um esquema que assegura a integridade e a coerência dos dados ao longo do tempo.

Com base na figura 29, representa-se um diagrama de entidade-relacionamento (DER), que mostra a estrutura de um banco de dados composto por várias tabelas interligadas. Cada retângulo no diagrama representa uma tabela, e cada linha dentro desses retângulos corresponde a um campo ou coluna da tabela, que armazena diferentes tipos de dados, como números inteiros (integer), textos (character varying), valores booleanos (true/false), ou datas e horas (timestamp). As chaves primárias (PK) são indicadas por ícones de chave e servem para identificar de maneira única cada registro dentro de uma tabela, enquanto as chaves estrangeiras (FK) são usadas para criar o relacionamento entre tabelas, conectando uma chave primária de uma tabela à outra. As setas que ligam essas tabelas indicam esses relacionamentos, mostrando como os dados de uma tabela se relacionam com os dados de outra, geralmente em uma estrutura de um-para-muitos. Isso significa que uma entrada em uma tabela pode estar relacionada a várias entradas em outra tabela, o que é comum em sistemas que lidam com grandes volumes de dados interconectados. Dessa forma, o diagrama facilita a visualização e o entendimento da organização dos dados e suas interdependências no sistema.

**Figura 29 –** Mapeamento do Objeto Relacional



Fonte:Elaborado pelos autores

5.2 SCRIPT DAS TABELAS

Um **script de tabelas** é um conjunto de instruções escritas em linguagem de banco de dados (como SQL) que define, cria e organiza as tabelas e seus respectivos campos (ou colunas) dentro de um banco de dados. Esses scripts especificam toda a estrutura que os dados devem seguir, incluindo tipos de dados, restrições, chaves primárias e estrangeiras. Essencialmente, um script de tabelas é como um "projeto estrutural" que detalha como o banco de dados armazenará e relacionará informações.

Em sua obra, Korth e Silberschatz (2005) afirmam que "o desenvolvimento de scripts de tabelas é essencial para garantir a integridade e a organização dos dados dentro de um sistema de banco de dados, permitindo uma estrutura consistente para suportar as operações e consultas" (Korth & Silberschatz, 2005, p. 124). Dessa forma, o uso de scripts facilita o gerenciamento e a manutenção dos dados, além de garantir que eles estejam sempre estruturados corretamente para atender aos requisitos do sistema.

**Quadro 32** – Script SQL – Tabela Aluno



CREATE TABLE aluno (

alunoid integer GE ERATED BY DEFAULT AS IDE TITY, nome character varying(150) OT ULL,

idade integer OT ULL,

rg character varying(12) OT ULL, statusaluno boolean OT ULL,

numeromatricula character varying(50) OT ULL, areainteresse character varying(100) OT ULL, habilidades character varying(100) OT ULL, experiencias character varying(350) OT ULL, disponibilidadehorario character varying(35) OT ULL, curriculo text OT ULL,

cpf character varying(14) OT ULL, cidade character varying(50) OT ULL,

datanascimento timestamp with time zone OT ULL, nivelescolaridade character varying(80) OT ULL, telefone character varying(14) OT ULL,

email character varying(50) OT ULL, endereco character varying(50) OT ULL, genero character varying(30) OT ULL, bairro character varying(30) OT ULL, cep character varying(9) OT ULL, “AlunoModelAlunoId” integer ULL,

CO STRAI T "flK\_aluno" flRIMARY KEY (alunoid)

CO STRAI T "FK\_aluno\_aluno\_AlunoModelAlunoId" FOREIG KEY (”AlunoModelAlunoId”) REFERE CES aluno (alunoid)

);

Fonte: Elaborado pelos autores.

**Quadro 33** – Script SQL – Tabela Concedente



CREATE TABLE concedente (

concedenteid integer GE ERATED BY DEFAULT AS IDE TITY, razaosocial character varying(80) OT ULL, responsavelestagio character varying(50) OT ULL, cnpj character varying(16) OT ULL,

localidade character varying(50) OT ULL,

CO STRAI T "flK\_concedente" flRIMARY KEY (concedenteid)

);

Fonte: Elaborado pelos autores.

**Quadro 34** – Script SQL – Tabela Curso



CREATE TABLE curso (

cursoid integer GE ERATED BY DEFAULT AS IDE TITY, nomecurso character varying(150) OT ULL, turnocurso character varying(100) OT ULL,

CO STRAI T "flK\_curso" flRIMARY KEY (cursoid)

);

Fonte: Elaborado pelos autores.

**Quadro 35** – Script SQL – Tabela Documento



CREATE TABLE documento (

documentoid integer GE ERATED BY DEFAULT AS IDE TITY, descricao character varying(200) OT ULL,

situacao character varying(200) OT ULL,

CO STRAI T "flK\_documento" flRIMARY KEY (documentoid)

);

Fonte: Elaborado pelos autores.

**Quadro 36** – Script SQL – Tabela TipoDocumento



CREATE TABLE tipodocumento (

tipodocumentoid integer GE ERATED BY DEFAULT AS IDE TITY, descricao character varying(200) OT ULL, statustipodocumento boolean OT ULL,

CO STRAI T "flK\_tipodocumento" flRIMARY KEY (tipodocumentoid)

);

Fonte: Elaborado pelos autores.

**Quadro 37** – Script SQL – Tabela TipoEstagio



CREATE TABLE tipoestagio (

tipoestagioid integer GE ERATED BY DEFAULT AS IDE TITY, descricao character varying(200) OT ULL,

CO STRAI T "flK\_tipoestagio" flRIMARY KEY (tipoestagioid)

);

Fonte: Elaborado pelos autores.

**Quadro 38** – Script SQL – Tabela Vagas



CREATE TABLE vagas (

vagasid integer GE ERATED BY DEFAULT AS IDE TITY, quantidade text OT ULL,

datapublicacao date OT ULL, datalimite date OT ULL,

localidade character varying(80) OT ULL, descricao character varying(800) OT ULL, titulo character varying(80) OT ULL,

localidadetrabalho character varying(20) OT ULL, horarioentrada character varying(20) OT ULL, horariosaida character varying(20) OT ULL, totalhorassemanais character varying(6) OT ULL, "concedenteId" integer OT ULL,

"CargoId" integer OT ULL,

CO STRAI T "flK\_vagas" flRIMARY KEY (vagasid),

CO STRAI T "FK\_vagas\_cargo\_cargoId" FOREIG KEY ("CargoId") REFERE CES cargo (cargoid) O DELETE RESTRICT,

CO STRAI T "FK\_vagas\_concedente\_concedenteId" FOREIG KEY ("concedenteId") REFERE CES concedente (concedenteid) O DELETE CASCADE

);

Fonte: Elaborado pelos autores.

**Quadro 39** – Script SQL – Tabela DocumentoVersao



CREATE TABLE documentoversao (

documentoversaoid integer GE ERATED BY DEFAULT AS IDE TITY, comentario character varying(200) OT ULL,

anexo character varying(200) OT ULL, date date OT ULL,

situacao character varying(200) OT ULL, documentoid integer OT ULL,

CO STRAI T "flK\_documentoversao" flRIMARY KEY (documentoversaoid),

CO STRAI T "FK\_documentoversao\_documento\_documentoid" FOREIG KEY (documentoid) REFERE CES documento (documentoid) O DELETE CASCADE

);

Fonte: Elaborado pelos autores.

**Quadro 40** – Script SQL – Tabela Cargo



CREATE TABLE cargo (

cargoid integer GE ERATED BY DEFAULT AS IDE TITY, descricao character varying(200) OT ULL,

tipo character varying(50) OT ULL, vagasid integer OT ULL,

CO STRAI T "flK\_cargo" flRIMARY KEY (cargoid),

);

Fonte: Elaborado pelos autores.

**Quadro 41** – Script SQL – Tabela CoordenadorEstagio



CREATE TABLE coordenadorestagio (

coordenadorestagioid integer GE ERATED BY DEFAULT AS IDE TITY, datacadastro date OT ULL,

nomecoordenador text OT ULL, statuscoordenadorestagio boolean OT ULL,

CO STRAI T "flK\_coordenadorestagio" flRIMARY KEY (coordenadorestagioid),

);

Fonte: Elaborado pelos autores.

**Quadro 42** – Script SQL – Tabela Login



CREATE TABLE login (

email text OT ULL, senha text OT ULL,

);

Fonte: Elaborado pelos autores.

**Quadro 43** – Script SQL – Tabela InstituicaoEnsino



CREATE TABLE instituicaoensino (

instituicaoensinoid integer GE ERATED BY DEFAULT AS IDE TITY, nomeinstituicao character varying(200) OT ULL, localinstituicao character varying(120) OT ULL, telefoneinstituicao character varying(17) OT ULL,

CO STRAI T "flK\_instituicaoensino" flRIMARY KEY (instituicaoensinoid),

);

Fonte: Elaborado pelos autores.

**Quadro 44** – Script SQL – Tabela Usuario



CREATE TABLE usuario (

usuarioid integer GE ERATED BY DEFAULT AS IDE TITY, nome text OT ULL,

cpfcnpjpessoa text OT ULL, email text OT ULL,

senha text OT ULL, usertype integer OT ULL,

CO STRAI T "flK\_usuario" flRIMARY KEY (usuarioid),

);

Fonte: Elaborado pelos autores.

**Quadro 45**– Script SQL – Tabela SupervisorEstagio



CREATE TABLE supervisorestagio (

supervisorid integer GE ERATED BY DEFAULT AS IDE TITY, nomesupervisor boolean OT ULL,

concedenteid integer OT ULL, “ConcedenteModelconcedenteId” integer OT ULL, senha text OT ULL,

usertype integer OT ULL,

CO STRAI T "flK\_supervisorestagio" flRIMARY KEY (supervisorid), CO STRAI T

"FK\_supervisorestagio\_concedente\_ConcedenteModelconcedenteId" FOREIG KEY ("ConcedenteModelconcedenteId") REFERE CES concedente (concedenteid),

CO STRAI T "FK\_supervisorestagio\_concedente\_concedenteid" FOREIG KEY (concedenteid) REFERE CES concedente (concedenteid) O DELETE CASCADE

);

Fonte: Elaborado pelos autores.

**Quadro 46**– Script SQL – Tabela Apontamento



CREATE TABLE apontamento (

apontamentoid integer GE ERATED BY DEFAULT AS IDE TITY, "descricaoApontamento" character varying(150) OT ULL, "dataApontamento" date OT ULL,

coordenadorestagioid integer OT ULL, "CoordenadorEstagioModelidCoordenadorEstagio" integer ULL, CO STRAI T "flK\_apontamento" flRIMARY KEY (apontamentoid),

CO STRAI T

"FK\_apontamento\_coordenadorestagio\_CoordenadorEstagioModelidCoo~" FOREIG KEY ("CoordenadorEstagioModelidCoordenadorEstagio") REFERE CES coordenadorestagio (coordenadorestagioid),

CO STRAI T "FK\_apontamento\_coordenadorestagio\_coordenadorestagioid" FOREIG KEY (coordenadorestagioid) REFERE CES coordenadorestagio (coordenadorestagioid) O DELETE CASCADE

);

Fonte: Elaborado pelos autores.

**Quadro 47**– Script SQL – Tabela Matricula



CREATE TABLE matricula (

matriculaid integer GE ERATED BY DEFAULT AS IDE TITY, numeromatricula character varying(15) OT ULL, "AlunosAlunoId" integer OT ULL,

alunoid integer OT ULL, cursoid integer OT ULL,

CO STRAI T "flK\_matricula" flRIMARY KEY (matriculaid),

CO STRAI T "FK\_matricula\_aluno\_AlunosAlunoId" FOREIG KEY ("AlunosAlunoId") REFERE CES aluno (alunoid),

CO STRAI T "FK\_matricula\_curso\_cursoid" FOREIG KEY (cursoid) REFERE CES curso (cursoid) O DELETE CASCADE

);

Fonte: Elaborado pelos autores.

**Quadro 48**– Script SQL – Tabela Matricula



CREATE TABLE matricula (

matriculaid integer GE ERATED BY DEFAULT AS IDE TITY, numeromatricula character varying(15) OT ULL, "AlunosAlunoId" integer OT ULL,

alunoid integer OT ULL, cursoid integer OT ULL,

CO STRAI T "flK\_matricula" flRIMARY KEY (matriculaid),

CO STRAI T "FK\_matricula\_aluno\_AlunosAlunoId" FOREIG KEY ("AlunosAlunoId") REFERE CES aluno (alunoid),

CO STRAI T "FK\_matricula\_curso\_cursoid" FOREIG KEY (cursoid) REFERE CES curso (cursoid) O DELETE CASCADE

);

Fonte: Elaborado pelos autores.

**Quadro 49**– Script SQL – Tabela DocumentoNecessario



CREATE TABLE documentonecessario (

documentonecessarioid integer GE ERATED BY DEFAULT AS IDE TITY, tipodocumentoid integer OT ULL,

tipoestagioid integer OT ULL, "TipoDocumentoModelidTipoDocumento" integer OT ULL, "TipoEstagioModelidTipoEstagio" integer OT ULL,

CO STRAI T "flK\_documentonecessario" flRIMARY KEY (documentonecessarioid),

CO STRAI T

"FK\_documentonecessario\_tipodocumento\_TipoDocumentoModelidTipoD~" FOREIG KEY ("TipoDocumentoModelidTipoDocumento") REFERE CES tipodocumento (tipodocumentoid),

CO STRAI T "FK\_documentonecessario\_tipodocumento\_tipodocumentoid" FOREIG KEY (tipodocumentoid) REFERE CES tipodocumento (tipodocumentoid) O DELETE CASCADE,

CO STRAI T

"FK\_documentonecessario\_tipoestagio\_TipoEstagioModelidTipoEstag~" FOREIG KEY ("TipoEstagioModelidTipoEstagio") REFERE CES tipoestagio (tipoestagioid),

CO STRAI T "FK\_documentonecessario\_tipoestagio\_tipoestagioid" FOREIG KEY (tipoestagioid) REFERE CES tipoestagio (tipoestagioid) O DELETE CASCADE

);

Fonte: Elaborado pelos autores.

**Quadro 50**– Script SQL – Tabela Sessão



CREATE TABLE sessao (

idsessao integer GE ERATED BY DEFAULT AS IDE TITY, datahoraabertura text OT ULL,

datahorafechamento text ULL, tokensessao text OT ULL, statussessao boolean OT ULL, emailpessoa text OT ULL, nivelacesso text OT ULL, "UsuarioId" integer OT ULL,

CO STRAI T "flK\_sessao" flRIMARY KEY (idsessao),

CO STRAI T "FK\_sessao\_usuario\_UsuarioId" FOREIG KEY ("UsuarioId") REFERE CES usuario (usuarioid) O DELETE CASCADE

);

Fonte: Elaborado pelos autores.

**Quadro 51**– Script SQL – Tabela ContratoEstagio

CREATE TABLE contratoestagio (

“ContratoEstagioId” integer GEERATED BY DEFAULT AS IDETITY, "Status do ContratoEstagio" boolean OT ULL,

notafinal character varying(150) OT ULL, situacao character varying(150) OT  ULL,

"orario de Entrada" character varying(150) OT ULL, "orario de Saida" character varying(150)  OT ULL, datainicio date OT  ULL,

datafim date OT ULL,

salario character varying(150)  OT ULL, cargasemanal character varying(150)  OT ULL, cargatotal character varying(150) OT ULL,

"CoordenadorEstagioidCoordenadorEstagio" integer ULL, coordenadorestagioid integer OT ULL, supervisorestagioid integer  OT ULL, "TipoEstagioidTipoEstagio" integer ULL,

tipoestagioid integer OT ULL, "SupervisorEstagioModelidSupervisor" integer  ULL,

COSTRAIT "flK\_contratoestagio" flRIMARY KEY ("ContratoEstagioid"), COSTRAIT

"FK\_contratoestagio\_coordenadorestagio\_CoordenadorEstagioidCoor~"

FOREIG KEY ("CoordenadorEstagioidCoordenadorEstagio") REFERECES coordenadorestagio (coordenadorestagioid),

COSTRAIT

"FK\_contratoestagio\_supervisorestagio\_SupervisorEstagioModelidS~" FOREIG KEY ("SupervisorEstagioModelidSupervisor") REFERECES supervisorestagio (supervisorid),

COSTRAIT "FK\_contratoestagio\_supervisorestagio\_supervisorestagioid" FOREIG KEY (supervisorestagioid) REFERECES supervisorestagio (supervisorid) O DELETE CASCADE,

COSTRAIT "FK\_contratoestagio\_tipoestagio\_TipoEstagioidTipoEstagio" FOREIG KEY ("TipoEstagioidTipoEstagio") REFERECES tipoestagio (tipoestagioid)

);

Fonte: Elaborado pelos autores.

5.3 MAPEAMENTO OBJETO RELACIONAL – ORM

O **mapeamento objeto-relacional (ORM)** é uma técnica que facilita a comunicação entre o código de uma aplicação e o banco de dados relacional. Em uma aplicação, trabalhamos com objetos — estruturas que possuem atributos e métodos. Por outro lado, o banco de dados relacional armazena informações em tabelas, com linhas e colunas. O processo de ORM atua como uma ponte entre esses dois universos, permitindo que dados sejam consultados, inseridos, atualizados e excluídos de forma direta, sem que o programador precise escrever código SQL diretamente.

Imagine que você tem uma classe "Cliente" no código, com atributos como "nome" e "email". O ORM cuida de transformar essa classe em uma tabela "Clientes" no banco de dados, onde cada instância da classe será uma linha da tabela, e os atributos "nome" e "email" se tornam colunas. Dessa forma, é possível trabalhar com o banco de dados de forma mais intuitiva, como se estivesse manipulando objetos do próprio código.

Segundo Larman (2004), o uso de ORM "minimiza a complexidade da conversão entre objetos e tabelas, permitindo ao desenvolvedor focar no código da aplicação sem se preocupar com a tradução dos dados" (Larman, 2004, p. 157). Em outras palavras, o ORM facilita o desenvolvimento, pois elimina a necessidade de escrever comandos SQL manualmente e automatiza a maioria das operações no banco de dados, tornando o código mais limpo e fácil de manter.

* TECNOLOGIA UTILIZADA

A tecnologia utilizada no projeto é o sistema de mapeamento objeto-relacional (ORM) SQLAlchemy, como detalhado no processo de desenvolvimento e configuração de classes e banco de dados dentro do código Python apresentado no projeto.

* EXEMPLO DE CÓDIGO UTILIZANDO ESSA TECNOLOGIA

### 1. Configuração do Banco de Dados e Definição de Modelos

Primeiro, configuramos o banco de dados e definimos as classes que representam as tabelas.

python

Copiar código

from sqlalchemy import create\_engine, Column, Integer, String

from sqlalchemy.ext.declarative import declarative\_base

from sqlalchemy.orm import sessionmaker

# Cria a conexão com o banco de dados SQLite (pode ser substituído por outro SGBD)

engine = create\_engine('sqlite:///meu\_banco.db')

Base = declarative\_base()

# Definindo a classe Cliente, que será mapeada para a tabela "clientes"

class Cliente(Base):

\_\_tablename\_\_ = 'clientes'

id = Column(Integer, primary\_key=True)

nome = Column(String)

email = Column(String)

def \_\_repr\_\_(self):

return f"<Cliente(nome='{self.nome}', email='{self.email}')>"

# Cria as tabelas no banco de dados (se não existirem)

Base.metadata.create\_all(engine)

Aqui, a classe Cliente representa a tabela clientes, onde id, nome e email são colunas da tabela.

### 2. Criando uma Sessão para Realizar Operações

Para interagir com o banco de dados, precisamos de uma sessão.

python

Copiar código

Session = sessionmaker(bind=engine)

session = Session()

### 3. Inserindo Dados na Tabela (Operação de Inserção)

Agora, vamos criar um objeto Cliente e adicioná-lo ao banco de dados.

python

Copiar código

# Cria um novo cliente

novo\_cliente = Cliente(nome="João Silva", email="joao.silva@email.com")

# Adiciona o cliente à sessão e confirma a transação no banco

session.add(novo\_cliente)

session.commit()

Aqui, o ORM traduz o código acima em um comando SQL INSERT INTO que insere um novo cliente na tabela.

### 4. Consultando Dados da Tabela (Operação de Consulta)

Para buscar dados, fazemos uma consulta usando a sessão.

python

Copiar código

# Consulta todos os clientes na tabela

clientes = session.query(Cliente).all()

# Exibe os clientes

for cliente in clientes:

print(cliente)

Esse código é convertido pelo ORM em um comando SELECT \* FROM clientes, retornando todos os clientes na tabela.

### 5. Atualizando Dados (Operação de Atualização)

Podemos atualizar registros específicos no banco.

python

Copiar código

# Consulta o cliente pelo nome

cliente\_para\_atualizar = session.query(Cliente).filter\_by(nome="João Silva").first()

cliente\_para\_atualizar.email = "novo.email@email.com"

# Confirma a atualização no banco

session.commit()

O ORM converte este código em um comando SQL UPDATE, atualizando o email do cliente com o nome "João Silva".

### 6. Excluindo Dados (Operação de Exclusão)

Para excluir um cliente:

python

Copiar código

# Consulta o cliente pelo nome e o exclui

cliente\_para\_excluir = session.query(Cliente).filter\_by(nome="João Silva").first()

session.delete(cliente\_para\_excluir)

session.commit()

Esse código é traduzido em um comando DELETE FROM, removendo o cliente do banco de dados.

Esses exemplos mostram como o ORM permite manipular o banco de dados como objetos Python, enquanto o SQLAlchemy cuida da tradução para comandos SQL, tornando o código mais limpo e fácil de manter.

# ARQUITETURA DE SOFTWARE

A arquitetura de software é um dos pilares fundamentais para a construção e evolução de sistemas complexos, sendo definida como o conjunto de decisões que determinam a organização, comportamento e qualidade de um sistema ao longo de seu ciclo de vida. Segundo Bass, Clements e Kazman (2003), a arquitetura de software abrange não apenas os componentes que constituem o sistema, mas também a maneira como esses componentes interagem e os padrões de design que estruturam a solução.

Componentes e conectores são elementos cruciais da arquitetura. Os componentes são as unidades modulares do sistema (como classes, bibliotecas e subsistemas), responsáveis por desempenhar funções específicas. Eles se conectam por meio de conectores — mecanismos que possibilitam a interação entre componentes, como APIs, chamadas de método e mensagens. Como destacam Shaw e Garlan (1996), a escolha de conectores adequados é determinante para a robustez e desempenho da comunicação no sistema, sendo um ponto chave na flexibilidade e escalabilidade da arquitetura.

Outro aspecto essencial são os padrões arquiteturais. Estes são soluções consagradas e reutilizáveis para problemas recorrentes em design de software. Gamma et al. (1995), no clássico Design Patterns: Elements of Reusable Object-Oriented Software, exploram como esses padrões organizam a estrutura interna dos componentes. O padrão Model-View-Controller (MVC), por exemplo, é amplamente usado para separar a interface do usuário da lógica de negócio, facilitando a manutenção e evolução do sistema. Já o padrão Microservices (Fowler, 2014) distribui o sistema em serviços independentes, otimizando escalabilidade e atualização.

De modo mais amplo, estilos arquiteturais fornecem estruturas gerais que orientam o design. A Arquitetura em Camadas, por exemplo, divide o sistema em níveis hierárquicos, como descrito por Buschmann et al. (1996) em Pattern-Oriented Software Architecture, promovendo separação de responsabilidades e maior organização estrutural. A Arquitetura de Microsserviços, um estilo mais recente, permite a construção de sistemas distribuídos, onde cada serviço é desenvolvido e mantido de forma independente, facilitando escalabilidade e resiliência.

A qualidade do software é outra dimensão fundamental. Segundo Pressman e Maxim (2015), uma arquitetura bem definida atende a requisitos de qualidade como desempenho, segurança e manutenibilidade, que influenciam diretamente a eficácia e a sustentabilidade do sistema ao longo do tempo. Boas práticas arquiteturais ajudam a assegurar que o sistema permaneça eficaz e adaptável, mesmo com mudanças e expansões futuras.

Por fim, a documentação arquitetural é essencial para preservar o entendimento das decisões de design e da estrutura do sistema. Esta documentação, como discutem Kruchten (1995) e Clements et al. (2002), orienta desenvolvedores, stakeholders e equipes de manutenção, garantindo que todos compreendam e acompanhem a evolução arquitetural do sistema.

* 1. ARQUITETURA DE DESENVOLVIMENTO

A arquitetura de desenvolvimento do TechVagas, é utilizada para definir a estrutura e a organização do código, bem como as interações entre os diferentes componentes do sistema. Como o TechVagas é um sistema em Web API e foi dividido entre back-end e front-end, essa arquitetura é importante para garantir que o sistema seja modular e de fácil manutenção. Conforme destacado por Pressman (2021), a arquitetura de desenvolvimento é importante em um software pois “Tomar boas decisões durante a definição da arquitetura é essencial para o sucesso de um produto de software. A arquitetura define a estrutura do sistema e determina a sua qualidade”.

Uma Web API (Application Programming Interface) é um conjunto de definições e protocolos usados para criar e integrar software de aplicações. Segundo Microsoft (2024):

Web API facilita a criação de serviços que alcancem uma ampla variedade de clientes, incluindo navegadores e dispositivos móveis. [...] Você usa o mesmo framework e os mesmos padrões para criar páginas da web e serviços lado a lado no mesmo projeto.

Cada parte do projeto desempenha um papel específico no fluxo de dados e na regra de negócio. A estrutura do back-end é desenvolvida em C# e é composta por várias camadas, cada uma responsável por uma parte distinta da aplicação. Já no front-end, o TechVagas foi desenvolvido utilizando React, uma biblioteca baseada em JavaScript muito popular para construção de interfaces de usuário dinâmicas e responsivas e utilizamos uma biblioteca de componentes de interface de usuário chamado de ShadCN UI, criada com o Tailwind CSS, voltada para o desenvolvimento de aplicações web. Ela oferece uma coleção de componentes prontos, como botões, tabelas e formulários, que podem ser facilmente copiados e integrados a projetos.

6.1.1 BACK-END

O back-end do Tech Vagas foi desenvolvido utilizando C#, a principal linguagem de programação da Microsoft para o desenvolvimento de aplicações .NET. C# é considerada uma das linguagens mais importantes do século XXI, especialmente por sua capacidade de suportar o desenvolvimento de sistemas robustos e de alto desempenho (Schildt, 2009). Segundo Skeet (2019), C# é uma linguagem moderna e orientada a objetos, amplamente utilizada no mercado para a criação de aplicações complexas, atendendo a diversas necessidades de desempenho e escalabilidade. Essa característica torna C# uma escolha apropriada para o desenvolvimento do Tech Vagas, que se destina a ser uma plataforma eficiente para a divulgação de vagas de estágio e gerenciamento de candidaturas.

O sistema Tech Vagas adota uma arquitetura em camadas, visando facilitar a manutenção e garantir a escalabilidade do projeto. Essa abordagem modular permite que diferentes responsabilidades sejam atribuídas a diferentes componentes da aplicação, tornando o código mais organizado e flexível para futuras alterações. A arquitetura é composta por várias camadas, cada uma com um papel específico.

A camada Controller é responsável por gerenciar as requisições recebidas da API. Ela atua como o ponto de entrada para todas as interações com o sistema, processando as solicitações, validando os dados e delegando as ações para as camadas apropriadas. A Controller garante que as respostas sejam retornadas de forma eficiente e organizada, permitindo que o sistema interaja corretamente com o usuário e com outros sistemas externos (Skeet, 2019).

Por sua vez, a camada Repository desempenha um papel fundamental na manipulação dos dados. Ela é responsável pela lógica de acesso e persistência de informações no banco de dados, funcionando como uma interface entre a camada de negócios e o armazenamento dos dados. A separação clara entre a lógica de acesso a dados e a lógica de negócios contribui para a manutenção e a escalabilidade do sistema, facilitando a evolução da aplicação à medida que novas funcionalidades são implementadas (Schildt, 2009).

A camada Services contém a lógica de negócios central do Tech Vagas. Nela são realizadas operações complexas, como a criação de novas vagas de estágio, a gestão de inscrições e o processamento de candidaturas. As regras de negócio são implementadas nesta camada, garantindo que a aplicação funcione de acordo com as necessidades do usuário final. A comunicação com o banco de dados é realizada de forma eficiente através da camada de DBContext, que assegura a leitura e escrita de dados de maneira segura e escalável (Skeet, 2019).

A camada Data é responsável pelo gerenciamento da conexão com o banco de dados. Ela inclui o TechVagasDBContext, que é o componente responsável pelo mapeamento dos dados e pela manutenção do estado da aplicação em relação ao banco de dados. Além disso, a camada Data também conta com a pasta Map, que define como os dados são mapeados entre as entidades do sistema e as tabelas do banco de dados, assegurando que as informações sejam persistidas de maneira correta e eficiente (Schildt, 2009).

A camada DTO (Data Transfer Object) desempenha um papel crucial na comunicação entre as diferentes camadas do sistema, especialmente na troca de dados entre a camada Controller e a camada Service. O DTO é um objeto simples que contém dados que são transferidos entre camadas, mas sem lógica de negócios. Sua principal função é otimizar a comunicação e evitar que objetos complexos, com muitas dependências ou informações desnecessárias, sejam passados entre as camadas. Isso melhora o desempenho da aplicação, uma vez que reduz a quantidade de dados trafegados e a complexidade das interações. Além disso, o uso de DTOs facilita a manutenção do sistema, pois permite que as camadas se comuniquem de forma mais clara e com maior desacoplamento, o que aumenta a flexibilidade do código e facilita a evolução da aplicação (Skeet, 2019).

Por fim, o Program.cs e o Startup.cs são responsáveis pela configuração e inicialização da aplicação. Esses arquivos definem as dependências e serviços necessários para o correto funcionamento do sistema. Eles garantem que a aplicação seja configurada corretamente desde o início, permitindo que o back-end seja iniciado de maneira eficiente e segura (Skeet, 2019).

A arquitetura modular do Tech Vagas, composta por essas camadas bem definidas, facilita a manutenção, a expansão e o desenvolvimento contínuo do sistema. Cada camada tem responsabilidades específicas, o que contribui para a criação de um sistema robusto e eficiente, capaz de lidar com a complexidade e as necessidades do mercado de estágios.

6.1.2 FRONT-END

O front-end é a parte do desenvolvimento de uma solução, cuja as principais responsabilidades são: se comunicar com o servidor e oferecer interações com o usuário através de uma interface. “O front-end é a interface do usuário — a parte de uma aplicação web que o usuário vê e interage. O front-end é essencial porque fornece a apresentação visual dos dados, garantindo que os usuários possam interagir com a aplicação de forma fácil e eficaz.” (Meloni, 2018).

O front-end foi desenvolvido utilizando o React, uma biblioteca JavaScript para criação de interfaces de usuário. Esta biblioteca permite a construção de componentes que gerenciam seus próprios estados e se combinam formar telas complexas (React, 2024). Como interface gráfica da aplicação, o front-end reúne elementos como menus, botões, imagens e layout, com o objetivo de proporcionar uma experiência fluída e intuitiva ao usuário, facilitando sua navegação e interação com a aplicação (TOTVS, 2021).

Além de facilitar a criação de interfaces complexas, o React contribui para uma organização de código mais modular e escalável. Cada componente é independente e atualizado de forma isolada, permitindo uma manutenção mais ágil e reduz o risco de falhas ao realizar alterações em partes específicas do código. Essa modularidade facilita o desenvolvimento incremental, em que novos recursos podem ser adicionados de forma contínua, sem impacto negativo no restante do sistema. Ademais, o React é amplamente suportado pela comunidade de desenvolvedores, com uma documentação abrangente de funcionalidades especificas, o que aumenta ainda mais a eficiência no desenvolvimento e na escalabilidade do projeto (React, 2024).

Para enriquecer ainda mais a experiência visual e simplificar o design, a biblioteca ShadCN UI foi adotada. Ela oferece componentes estilizados com Tailwind CSS, como botões, tabelas e formulários, que são facilmente personalizáveis e responsivos, facilitando a criação de uma interface consistente e atraente para os usuários. Com essa combinação, o front-end torna-se modular, escalável e esteticamente agradável, proporcionando uma navegação intuitiva e eficiente (React, 2024; TOTVS, 2021).

6.2 SEGURANÇA DA INFORMAÇÃO

A Segurança da Informação, de acordo com Fontes (2008), envolve um conjunto de políticas, procedimentos e medidas técnicas com o objetivo de proteger dados e informações sensíveis contra acessos não autorizados, divulgações indevidas, alterações e exclusões. Com o aumento da conectividade e a crescente dependência de sistemas digitais, a proteção das informações tornou-se uma necessidade fundamental para organizações de diversos setores. A violação de dados pode resultar em sérios danos financeiros, reputacionais e até legais. Dessa forma, a implementação de medidas de segurança eficientes que assegurem a confidencialidade, integridade e disponibilidade das informações é essencial para criar um ambiente confiável e seguro tanto para os usuários quanto para as empresas.

Dentro desse contexto de segurança, o sistema Tech Vagas adota o uso do JSON Web Token (JWT) para garantir uma autenticação e autorização seguras. O JWT é um padrão aberto (conforme a RFC 7519) que define um formato compacto e autossuficiente para a troca de informações entre partes, utilizando um objeto JSON. Sua utilização no sistema é justificada pela capacidade de oferecer uma forma eficiente e segura de autenticar e autorizar usuários, permitindo a transmissão de informações de maneira protegida entre o cliente e o servidor (JWT, 2024).

O processo de geração do token JWT inicia-se no momento em que o usuário faz login no sistema. Nesse momento, suas credenciais são verificadas e, uma vez validadas, um payload (carga útil) contendo as informações relevantes do usuário é criado. Esse payload, juntamente com o cabeçalho (header) do token, que descreve o tipo de token e o algoritmo de assinatura utilizado, é então assinado com uma chave secreta. O token JWT é, portanto, composto por três partes principais: o header, que contém as informações sobre o tipo do token e o algoritmo de assinatura; o payload, que carrega as informações que devem ser transmitidas; e a signature, que é a parte do token responsável por garantir a autenticidade da mensagem e verificar sua integridade (JWT, 2024).

Para cada requisição que exige autenticação, o token JWT é enviado pelo cliente no cabeçalho Authorization. O servidor, por sua vez, realiza a validação do token, conferindo sua assinatura e verificando se ele está dentro do prazo de validade. Para garantir a segurança adicional, o uso de algoritmos de assinatura robustos e a escolha de uma chave secreta forte são essenciais. Além disso, a implementação de políticas de expiração de tokens e a possibilidade de revogação dos mesmos são medidas importantes para manter a segurança ao longo do tempo. A utilização do JWT no Tech Vagas oferece uma camada adicional de proteção ao sistema, garantindo uma autenticação eficiente, além de proporcionar uma experiência de usuário mais fluida e segura.

Portanto, ao adotar o JWT como mecanismo de autenticação, o Tech Vagas assegura que a comunicação entre cliente e servidor seja protegida contra acessos não autorizados, minimizando riscos de segurança e garantindo a confiança dos usuários no sistema. O JWT se configura, assim, como uma escolha estratégica para a implementação de um processo de autenticação robusto e seguro em sistemas web.

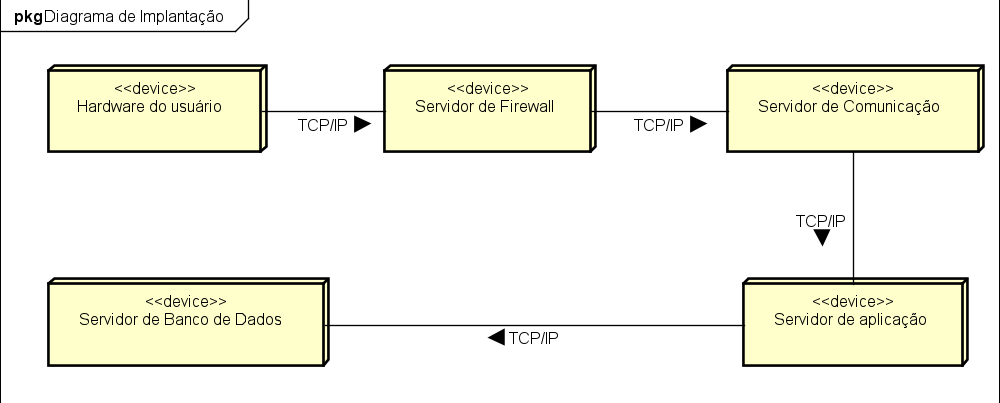
6.3 IMPLANTAÇÃO

O diagrama de implantação é uma ilustração visual que contribui para o planejamento e a estruturação da infraestrutura de hardware onde o software será implementado. Ele detalha os equipamentos que sustentarão o sistema, tais como servidores e computadores, e estabelece as configurações de conexão e os protocolos requeridos para a comunicação e transferência de informações entre esses aparelhos.

Adicionalmente, o diagrama possibilita a visualização da alocação dos módulos em diversos servidores, um elemento crucial em sistemas que funcionam em contextos distribuídos. Sua aplicação é particularmente pertinente para sistemas que requerem várias máquinas para a execução de módulos ou armazenamento de arquivos vitais (GUEDES, 2018).

A imagem a seguir ilustra graficamente o planejamento da infraestrutura de hardware, destacando a distribuição dos módulos do sistema e a comunicação entre os diversos servidores e dispositivos que sustentam o ambiente.

Figura 34 – Diagrama de Implantação



Fonte:Elaborado pelos autores

O Docker, uma plataforma de virtualização compacta, permite a empacotamento, distribuição e execução de aplicações em contêineres, criando ambientes uniformes e padronizados para desenvolvimento, teste e produção.

Cada contêiner engloba a aplicação e todas as suas dependências, incluindo bibliotecas e configurações, assegurando uma operação consistente em variados sistemas operacionais. Isso minimiza desavenças de compatibilidade e oferece eficácia e adaptabilidade, maximizando o uso dos recursos do sistema.

O Docker facilita a instalação e o escalonamento de aplicativos, seja na nuvem ou em servidores locais, sendo crucial para equipes de desenvolvimento que procuram rapidez e segurança na entrega de software (DOCKER).

A implementação de sistemas distribuídos requer instrumentos que simplifiquem a configuração, automação e administração de várias aplicações de maneira eficaz e segura. Neste contexto, o Docker emerge como uma plataforma sólida para a criação e administração de contêineres, unidades de software leves e independentes que envolvem o código e suas dependências.

Com o Docker, podemos estabelecer ambientes de desenvolvimento e produção que garantem uniformidade, autonomia de infraestrutura e compatibilidade entre diversos ambientes, tais como estações de trabalho e servidores na nuvem.

Este estudo descreve a implementação de um sistema integral usando Docker para conectar o front-end, o back-end e o banco de dados, ilustrando as configurações cruciais para que o sistema opere de forma unificada e eficaz.

É preciso criar um arquivo chamado Dockerfile para configurar o ambiente de back-end no Docker. Este arquivo deve ser colocado no mesmo diretório onde se encontra o arquivo.sln do projeto. Este Dockerfile incluirá as diretrizes fundamentais para a construção do contêiner que será utilizado para executar a aplicação.

Depois, é crucial compilar a aplicação e todas as suas dependências, algo que pode ser realizado através do comando `dotnet publish -c Release`, produzindo o build e preparando o projeto para ser executado.

No diretório do front-end, é preciso criar um arquivo `. env` contendo a URL da API, seguindo o formato `VITE\_API\_URL=https://localhost:7188/api/`. Essa URL deve corresponder à porta em que a API do back-end está rodando.

Além disso, é necessário incluir um Dockerfile no diretório do front-end, com instruções para compilar e executar a aplicação. Esse Dockerfile deve instalar as dependências, copiar os arquivos do projeto e definir as variáveis de ambiente.

A construção do front-end começa com a imagem `node:18`, e a aplicação compilada é servida usando `nginx:alpine`. Por fim, é preciso configurar o Nginx no arquivo `nginx.conf`, especificando o servidor para direcionar as requisições ao front-end e, quando necessário, ao contêiner da API.

A integração entre os serviços de back-end, front-end e banco de dados é gerenciada por meio do arquivo docker-compose.yml, que define as configurações de cada serviço, incluindo portas, variáveis de ambiente e as dependências entre os contêineres.

No docker-compose.yml, o serviço PostgreSQL é configurado para ser iniciado primeiro, com variáveis de ambiente para usuário, senha e nome do banco, além de um volume que garante a persistência dos dados.

A API é configurada para ser dependente do PostgreSQL, sendo iniciada somente quando o banco de dados estiver em condições adequadas. Finalmente, o serviço de front-end, que utiliza a API, começa após a configuração dos demais serviços. Este processo garante que todos os contêineres sejam iniciados na sequência correta e que as dependências estejam adequadamente definidas.

Para manter o ambiente Docker organizado e evitar resíduos de configurações anteriores, recomenda-se usar o comando docker-compose down --rmi all -v, que remove todos os contêineres e imagens criados anteriormente.

Em seguida, para construir e inicializar os contêineres, deve-se executar docker-compose up --build. Esse processo cria e inicia o ambiente completo, integrando o banco de dados, a API e o front-end, permitindo o acesso a cada serviço de acordo com as portas configuradas. Essa estruturação com Docker oferece um ambiente de desenvolvimento e produção coeso e modular, ideal para aplicações distribuídas.

# 7 CONCLUSÃO

A conclusão deste TCC deve abordar os principais aspectos analisados ao longo do trabalho e avaliar a viabilidade do desenvolvimento do sistema proposto. O projeto TECHVAGAS se concentrou na criação de um sistema eficiente para a gestão de vagas de estágio, buscando atender as necessidades específicas tanto dos candidatos quanto das empresas. Ao longo do trabalho, foi possível desenvolver uma plataforma robusta, baseada em uma arquitetura que permite o cadastro, a alteração e o controle de dados de maneira organizada e segura.

Primeiramente, a implementação de funcionalidades como cadastro de usuários, empresas e vagas foi fundamental para criar uma estrutura organizada e acessível. Além disso, o sistema incluiu requisitos funcionais e não funcionais essenciais, como a atualização ágil do banco de dados e o suporte ao usuário, aspectos que contribuem para a eficiência do sistema e a satisfação dos usuários.

Outro ponto relevante foi o foco em acessibilidade, garantindo que a plataforma fosse utilizável por todos os públicos, incluindo pessoas com diferentes limitações. A inclusão de funcionalidades de acessibilidade digital, como validações em tempo real e interfaces adaptáveis, reforçou o compromisso com a inclusão social e ampliou o potencial de uso do sistema por um público diversificadoise da viabilidade técnica e econômica do sistema indica que o desenvolvimento é favorável, dada a estrutura bem planejada e a adoção de práticas que facilitam a manutenção e a escalabilidade. Dessa forma, conclui-se que o projeto é viável e oferece uma solução inovadora e prática para conectar estudantes a oportunidades de estágio, contribuindo significativamente para o desenvolvimento acadêmico e profissional dos usuários.

# 8 REFERÊNCIAS

Colocar as referências utilizadas no projeto conforme os modelos abaixo:

ASFHAL, C. R. **Gestão de segurança do trabalho e de saúde ocupacional**. São Paulo: Reichmann & Autores, 2005.

BEAIRD, J. Princípios do Web Design Maravilhoso. Rio de Janeiro: Altabooks, 2008.

CHIAVENATO, I. **Gestão de pessoas**: o novo papel dos recursos humanos nas organizações.

3. ed. Rio de Janeiro: Elsevier, 2008.

COSTA, H. J. Acidentes do trabalho: teremos nova lei acidentária?. **Jus Navigandi**, Teresina,

v. 9, n. 664, 1 maio 2005. Disponível em:

<[http://jus2.uol.com.br/doutrina/texto.asp?id=6662>](http://jus2.uol.com.br/doutrina/texto.asp?id=6662). Acesso em: 1 jul. 2010.

DELIBERATO, P. C. P. **Fisioterapia preventiva**: fundamentos e aplicações. São Paulo: Manole, 2002.

### DIAS, E. C.; MELO, E. M. de. Políticas públicas em saúde e segurança no trabalho. In: MENDES, R. (Org.). **Patologia do trabalho**. 2. ed. atual. e ampl. São Paulo: Atheneu, 2007.

v. 2. p. 1683-1720.

GUEDES, Gilleanes T. A. **UML Uma Abordagem prática**, 3 ed. São Paulo: Novatec, 2008.

GUEDES, Gilleanes T. A. **UML 2 Uma Abordagem prática**, São Paulo: Novatec, 2009.

WATRALL, E; SIARTO, J. Use A Cabeça! **Web Design.** Alta Books, 2009.

SOMMERVILLE, Ian. Engenharia de Software. 10. ed. São Paulo: Pearson, 2011.

SOMMERVILLE, I. (2011). Engenharia de Software (9ª ed.). Pearson.

DATE, C. J. An Introduction to Database Systems. 8. ed. Boston: Pearson, 2004.

ELMASRI, R.; NAVATHE, S. B. Fundamentals of Database Systems. 7. ed. Boston: Pearson, 2016.

MARTIN, J. Principles of Data Base Management. New York: Prentice Hall, 1988.

BASS, L., CLEMENTS, P., & KAZMAN, R. (2003). Software Architecture in Practice. Addison-Wesley.

SHAW, M., & GARLAN, D. (1996). Software Architecture: Perspectives on an Emerging Discipline. Prentice Hall.

GAMMA, E., HELM, R., JOHNSON, R., & VLISSIDES, J. (1995). Design Patterns: Elements of Reusable Object-Oriented Software. Addison-Wesley.

FOWLER, M. (2014). *Microservices Resource Guide*. martinfowler.com.

BUSCHMANN, F., MEUNIER, R., ROHNERT, H., SOMMERLAD, P., & STAL, M. (1996). Pattern-Oriented Software Architecture: A System of Patterns. Wiley.

PRESSMAN, R. S., & MAXIM, B. R. (2015). Software Engineering: A Practitioner's Approach. McGraw-Hill.

KRUCHTEN, P. (1995). Architectural Blueprints - The “4+1” View Model of Software Architecture. IEEE Software, 12(6), 42-50.

CLEMENTS, P., BACHMANN, F., BASS, L., GARLAN, D., IVERS, J., LITTLE, R., NORD, R., & STAFFORD, J. (2002). Documenting Software Architectures: Views and Beyond. Addison-Wesley.

SCHILDT, Herbert. C# 3.0: A linguagem de programação. 5. ed. Porto Alegre: Bookman, 2009.

SKEET, Jon. C# in depth. 4. ed. Shelter Island: Manning Publications, 2019.

FONTES, Roberto. Segurança da Informação: Princípios e Práticas. São Paulo: Editora Atlas, 2008.

JWT. JSON Web Token - RFC 7519. Disponível em: https://tools.ietf.org/html/rfc7519. Acesso em: 07 nov. 2024.