

Curso Superior de Desenvolvimento de Software Multiplataforma

Carlos Eduardo da Silva, 3011392323015

Matheus Prusch, 3011392323025

Raphael Cremer Albiero, 3011392323028

Samuel Porcer Pregnolatto, 3011392323042

Projeto Interdisciplinar

Engenharia de Software II

Desenvolvimento Web II

Banco de Dados Relacional

Técnicas de Programação I

Ada Company

Orientador: Profa Cristiane Palomar Mercado

Profo Wladimir Zuanazzi

Profo Jones Artur Gonçalves

Profo Rodrigo de Paula Diver

Votorantim

Junho, 2024

**SUMÁRIO**

[1. DESCRIÇÃO DO PROJETO 6](#_Toc169882580)

[1.1. Introdução. 6](#_Toc169882581)

[1.2. Logomarca 7](#_Toc169882582)

[1.3. Diagrama de Caso de Uso 8](#_Toc169882583)

[1.4. Descrição do Caso de Uso 9](#_Toc169882584)

[1.5. Requisitos Não Funcionais 19](#_Toc169882585)

[2. PROJETO DO SOFTWARE 20](#_Toc169882586)

[2.1. Tecnologias Utilizadas 20](#_Toc169882587)

[2.2. Modelo de dados 23](#_Toc169882588)

[3.2.1 Modelo Conceitual 23](#_Toc169882589)

[3.2.2 Modelo Lógico 24](#_Toc169882590)

[2.3. Diagrama de Classe MVC 25](#_Toc169882591)

[2.4. Diagrama de Sequência Orçamento 26](#_Toc169882592)

[2.5. Interfaces com o usuário 27](#_Toc169882593)

[3.6.2 Interfaces Desktop 27](#_Toc169882594)

[3. IMPLEMENTAÇÃO NO PADRÃO MVC 33](#_Toc169882595)

[3.1. Model 33](#_Toc169882596)

[3.1. Controller 35](#_Toc169882597)

[3.2. View 35](#_Toc169882598)

[3.3. Outras classes 36](#_Toc169882599)

[3.4. Implementação dos métodos 37](#_Toc169882600)

[4. ESTRATÉGIA DE TESTES 40](#_Toc169882601)

[5. IMPLANTAÇÃO 40](#_Toc169882602)

[6. CONCLUSÃO 40](#_Toc169882603)

[7. REFERÊNCIAS 41](#_Toc169882604)

**LISTA DE FIGURAS**

[Figura 1 - Logotipo 7](#_Toc169869491)

[Figura 2 - Diagrama de Caso de Uso 10](#_Toc169869492)

[Figura 3 - Modelo Conceitual 25](#_Toc169869493)

[Figura 4 - Modelo Lógico 26](#_Toc169869494)

[Figura 5 - Diagrama de Classe 27](#_Toc169869495)

[Figura 6 - Diagrama de Sequência 28](#_Toc169869496)

[Figura 7 - Tela Principal (Administração) 29](#_Toc169869497)

[Figura 8 - Consultar Usuários 30](#_Toc169869498)

[Figura 9 - Cadastro de Usuário 30](#_Toc169869499)

[Figura 10 - Editar Perfil 31](#_Toc169869500)

[Figura 11 - Consultar Serviços 32](#_Toc169869501)

[Figura 12 - Gerenciar Serviços 33](#_Toc169869502)

[Figura 13 - Gerenciamento de Orçamentos 33](#_Toc169869503)

[Figura 14 - Cadastro de Orçamento 34](#_Toc169869504)

[Figura 15 - Padrão MVC Atual 41](#_Toc169869505)

**LISTA DE QUADROS**

[Tabela 1 - Requisitos Funcionais 9](#_Toc169869506)

[Tabela 2 - Caso de uso "Realizar Cadastro” 11](#_Toc169869507)

[Tabela 3 - Caso de uso "Realizar Login" 12](#_Toc169869508)

[Tabela 4 - Caso de uso "Acessar Perfil” 13](#_Toc169869509)

[Tabela 5 - Caso de uso "Bloquear Acesso” 14](#_Toc169869510)

[Tabela 6 - Caso de uso "Recuperar senha” 15](#_Toc169869511)

[Tabela 7 - Caso de uso "Solicitar Orçamento” 16](#_Toc169869512)

[Tabela 8 - Caso de uso "Acompanhar Orçamento” 17](#_Toc169869513)

[Tabela 9 - Caso de uso "Entrar em Contato” 18](#_Toc169869514)

[Tabela 10 - Caso de uso "Visualizar Serviços” 19](#_Toc169869515)

[Tabela 11 - Caso de uso "Gerenciar Orçamento” 20](#_Toc169869516)

[Tabela 12 - Requisitos não Funcionais 21](#_Toc169869517)

# DESCRIÇÃO DO PROJETO

## Introdução Este documento apresenta o projeto de um software em versão desktop capaz de gerar orçamentos de serviços, acompanhar o status dos pedidos, fornecer análises sobre os serviços prestados e criação e edição de usuários. A empresa utilizadora do software visa promover a acessibilidade digital, prestando serviços e consultoria em desenvolvimento web, com foco em públicos negligenciados, como idosos, deficientes em geral e pessoas que realizam pouco uso de tecnologias. O projeto em si, apresentará os detalhes e o processo do software desktop desenvolvido, explicando o funcionamento das classes utilizadas, métodos e como eles foram aplicados dentro do modelo MVC.

## Logomarca

Figura 1 - Logotipo

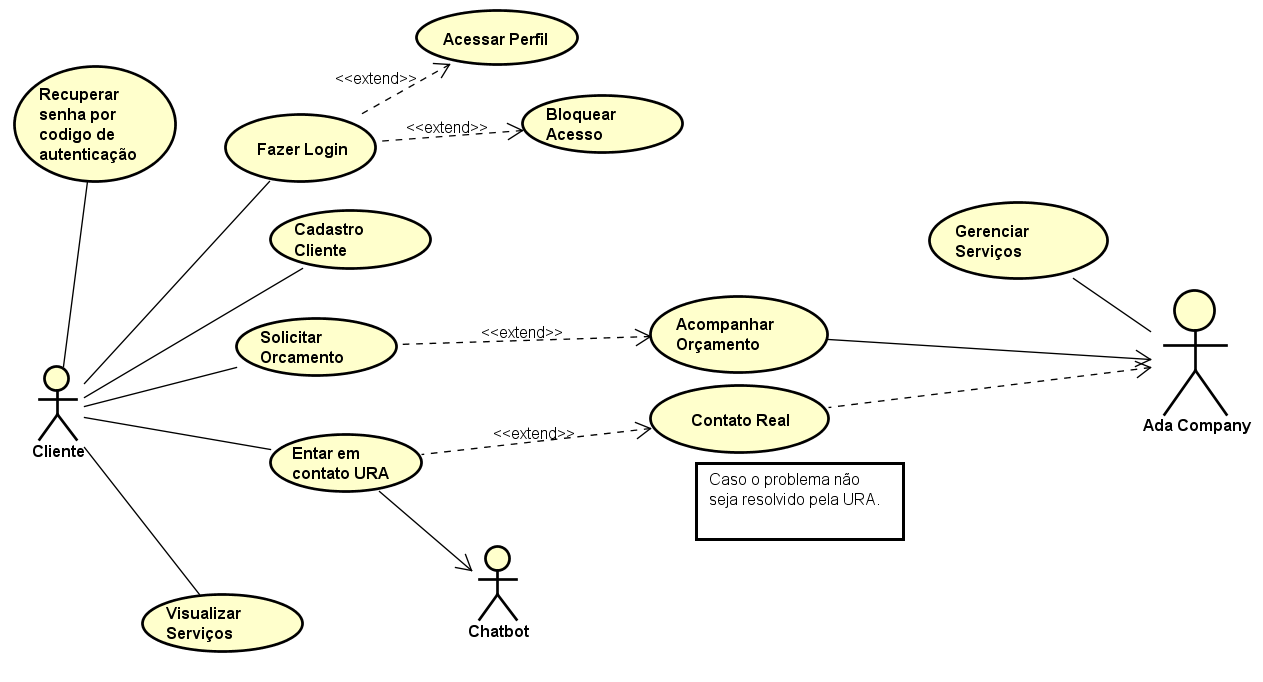
Forma

Descrição gerada automaticamente com confiança baixa

Fonte: elaboração própria com a ferramenta Looka (2023)

## Diagrama de Caso de Uso

Figura 2 - Diagrama de Caso de Uso



Fonte: elaboração própria (2024)

## Descrição do Caso de Uso

Tabela 2 - Caso de uso "Realizar Cadastro”

|  |  |  |
| --- | --- | --- |
| **Caso de uso** | RF001 - Realizar Cadastro | |
| **Ator Principal** | Cliente/Ada Company | |
| **Ator Secundário** | - | |
| **Pré-condições** | Fornecer dados pessoais necessários para o sistema criar um usuário | |
| **Pós-condições** | Utilizar o Usuário e senha criado para acessar o sistema | |
| **Fluxo Principal** | | |
| **Ações do Ator** | | **Ações do Sistema** |
| **Cadastrar** | | |
| 1. Usuário entra na tela de cadastro para preencher com seus dados | |  |
|  | | 1. Sistema solicita os dados do usuário para criar seu cadastro |
| 1. Usuário fornece seu nome, e-mail, telefone e senha para criar seu acesso | |  |
|  | | 1. Sistema pede confirmação de senha |
| 1. Usuário confirma senha e clica em cadastrar | |  |
|  | | 1. Sistema cria cadastro e dispara mensagem de que o cadastro foi realizado com sucesso |
| **Tratamento de exceções:** | |  |
| 3.a Usuário fornece dados inválidos | |  |
|  | | 2.a Sistema informa que os dados são inválidos e solicita os dados corretos novamente |
| 5.a Usuário confirma senha incorreta | |  |
|  | | 4.b Sistema pede que confirme a mesma senha digitada para realizar o cadastro corretamente |

Fonte: elaboração própria (2024).

Tabela 3 - Caso de uso "Realizar Login"

|  |  |  |
| --- | --- | --- |
| **Caso de uso** | RF002 - Realizar Login | |
| **Ator Principal** | Cliente/Ada Company | |
| **Ator Secundário** | - | |
| **Pré-condições** | Fornecer usuário e senha criado para acessar a plataforma | |
| **Pós-condições** | Pesquisar/Solicitar orçamento, atualizar ou excluir cadastro | |
| **Fluxo Principal** | | |
| **Ações do Ator** | | **Ações do Sistema** |
| **Logar** | | |
| 1. Usuário entra na tela de login | |  |
|  | | 1. Sistema solicita o usuário e senha |
| 1. Usuário fornece seu nome de usuário e senha e confirma | |  |
|  | | 1. Sistema valida o Login e atualiza plataforma para acesso do usuário |
| **Tratamento de exceções:** | |  |
| 3.a Usuário fornece dados inválidos | |  |
|  | | 4.a Sistema informa que os dados de login são inválidos e solicita os dados corretos novamente |
| 3.b Usuário confirma senha ou usuário incorreto novamente | |  |
|  | | 4.b Sistema Bloqueia o acesso |

Fonte: elaboração própria (2024).

Tabela 4 - Caso de uso "Acessar Perfil”

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Caso de uso** | | RF003 – Acessar Perfil | | |
| **Ator Principal** | | Cliente/Funcionário | | |
| **Ator Secundário** | | - | | |
| **Pré-condições** | | Fornecer usuário e senha criado para acessar a plataforma | | |
| **Pós-condições** | | Acessar o perfil após login | | |
| **Fluxo Principal** | | | | |
| **Ações do Ator** | | | **Ações do Sistema** | |
| **Acessar perfil** | | | | |
| 1. Usuário acessa o perfil | | |  | |
|  | | | 1. Sistema direciona o usuário ao seu perfil | |
| **Atualizar** | | | |
| 1. Usuário acessa o perfil | | |  |
|  | | | 1. Sistema exibe acesso pessoal |
| 1. Usuário clica em atualizar dados | | |  |
|  | | | 1. Sistema exibe os dados do usuário para alteração |
| 1. Usuário realiza ou não mudanças, e rola a tela até o final e clica em salvar | | |  |
|  | | | 1. Sistema retorna que dados foram atualizados |
| 1. Usuário preenche nova senha | | |  |

|  |  |
| --- | --- |
| **Excluir** | |
| 1. Usuário acessa o perfil |  |
|  | 1. Sistema exibe acesso pessoal |
| 1. Usuário clica em excluir perfil |  |
|  | 1. Sistema questiona se tem certeza |
| 1. Usuário clica em confirmar exclusão |  |
|  | 1. Sistema exclui o acesso e retorna mensagem “Exclusão efetuada com sucesso.” |

Fonte: elaboração própria (2024).

Tabela 5 - Caso de uso "Bloquear Acesso”

|  |  |  |
| --- | --- | --- |
| **Caso de uso** | RF004 - Bloquear Acesso | |
| **Ator Principal** | Cliente/Funcionário | |
| **Ator Secundário** | - | |
| **Pré-condições** | Bloquear o acesso do usuário caso tenha excedido as tentativas de acesso | |
| **Pós-condições** | Fornecer meios de recuperação de Login para o usuário | |
| **Fluxo Principal** | | |
| **Ações do Ator** | | **Ações do Sistema** |
| **Bloquear** | | |
| 1. Usuário fornece dados inválidos. | |  |
|  | | 1. Sistema informa que os dados de login são inválidos e solicita os dados corretos novamente. |
| 1. Usuário fornece dados incorretos por mais duas vezes seguidas. | |  |
|  | | 1. Sistema Bloqueia o acesso. |
| **Tratamento de exceções:** | |  |
| 3.a Usuário fornece dados inválidos | |  |
|  | | 4.a Sistema informa que os dados de login são inválidos e solicita os dados corretos novamente |
| 3.b Usuário confirma senha ou usuário incorreto novamente | |  |
|  | | 4.b Sistema Bloqueia o acesso |

Fonte: elaboração própria (2024).

Tabela 6 - Caso de uso "Recuperar senha”

|  |  |  |
| --- | --- | --- |
| **Caso de uso** | RF005 - Recuperar senha | |
| **Ator Principal** | Cliente/Ada Company | |
| **Ator Secundário** | - | |
| **Pré-condições** | Possuir Login de acesso | |
| **Pós-condições** | Solicitar recuperação de senha por e-mail | |
| **Fluxo Principal** | | |
| **Ações do Ator** | | **Ações do Sistema** |
| **Cadastrar** | | |
| 1. Usuário na página de login clica em recuperar senha | |  |
|  | | 1. Sistema envia recuperação de senha via e-mail |
| 1. Usuário abre seu e-mail pessoal e recebe um link para alteração de senha dentro da plataforma e clica | |  |
|  | | 1. Sistema direciona o agente para a página de Login e fornece a opção de alterar senha |
| 1. Usuário digita nova senha e confirma ela novamente | |  |
|  | | 1. Sistema confirma que senha foi alterada com sucesso |

Fonte: elaboração própria (2024).

Tabela 7 - Caso de uso "Solicitar Orçamento”

|  |  |  |
| --- | --- | --- |
| **Caso de uso** | RF006 – Solicitar Orçamento | |
| **Ator Principal** | Cliente | |
| **Ator Secundário** | - | |
| **Pré-condições** | Estar logado no sistema. | |
| **Pós-condições** | Registro de uma entrada no banco de dados na tabela Orçamentos. | |
| **Fluxo Principal** | | |
| **Ações do Ator** | | **Ações do Sistema** |
| **REALIZAR ORÇAMENTO** | | |
| 1. Cliente preenche o formulário de orçamento com dados do serviço e envia. | |  |
|  | | 1. Sistema pede para confirmar se todas as informações estão corretas. |
| 1. Usuário verifica e confirma as informações preenchidas. | |  |
|  | | 1. Sistema registra em uma tabela de orçamentos e informa que a solicitação foi encaminhada para a equipe de projetos e em breve retornaremos |

Fonte: elaboração própria (2024).

Tabela 8 - Caso de uso "Acompanhar Orçamento”

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Caso de uso** | | RF007 - Acompanhar Orçamento | | |
| **Ator Principal** | | Cliente/ Ada Company | | |
| **Ator Secundário** | | - | | |
| **Pré-condições** | | Ter feito Login e solicitação de orçamento | | |
| **Pós-condições** | | - | | |
| **Fluxo Principal** | | | | |
| **Ações do Ator** | | | **Ações do Sistema** | |
| **ACOMPANHAR ORÇAMENTO** | | | | |
| 1. Cliente visualiza uma tabela com todos os seus orçamentos pendentes. | | |  | |
|  | | | 1. Sistema demonstra o status dos orçamentos pendentes, em contato, negociação. | |
| 1. Cliente clica em “Ver Detalhes do orçamento”; | | |  | |
|  | | | 1. Sistema retorna com os dados mais detalhados do orçamento selecionado e meios de contato caso o cliente queira confirmação real do processo desenvolvido | |
| **ALTERAR** | | | |
| 1. Cliente altera dados do orçamento. | | |  |
|  | | | 1. Sistema solicita confirmação dos dados alterados. |
| 1. Cliente verifica e confirma os dados. | | |  |
|  | | | 1. Sistema atualiza os dados da tabela. |
| **EXCLUIR** | | | |
| 1. Cliente exclui orçamento gerado em até 5 dias corridos. | | |  |
|  | | | 1. Sistema solicita uma confirmação de exclusão |
| 1. Cliente confirma exclusão. | | |  |
|  | | | 1. Sistema deleta o registro do orçamento. |

Fonte: elaboração própria (2024).

Tabela 9 - Caso de uso "Entrar em Contato”

|  |  |  |
| --- | --- | --- |
| **Caso de uso** | RF008 - Entrar em contato URA (atendente virtual) | |
| **Ator Principal** | Cliente/Ada Company | |
| **Ator Secundário** | - | |
| **Pré-condições** | Ter um número com WhatsApp | |
| **Pós-condições** | Ter realizado o contato com nosso número | |
| **Fluxo Principal** | | |
| **Ações do Ator** | | **Ações do Sistema** |
| **CONTATAR** | | |
| 1. Cliente envia uma mensagem para sanar suas dúvidas. | |  |
|  | | 1. Sistema retorna uma mensagem automática com uma lista de opções para ajudar o cliente com suas dúvidas |
| 1. Cliente responde informando qual a sua dúvida. | |  |
|  | | 1. Sistema retorna uma mensagem automática com base em palavras-chave. Caso a interação não seja suficiente o cliente é redirecionamrnto automático. |

Fonte: elaboração própria (2024).

Tabela 10 - Caso de uso "Visualizar Serviços”

|  |  |  |
| --- | --- | --- |
| **Caso de uso** | RF009 - Visualizar Serviços | |
| **Ator Principal** | Cliente | |
| **Ator Secundário** | - | |
| **Pré-condições** | Acessar a aba serviços | |
| **Pós-condições** | - | |
| **Fluxo Principal** | | |
| **Ações do Ator** | | **Ações do Sistema** |
| **VISUALIZAR** | | |
| 1. Cliente visualiza os exemplos dos nossos serviços | |  |
|  | | 1. Sistema oferece a opção de “Ver mais” caso o cliente se interesse por algum serviço |
| 1. Cliente clica no campo “Ver mais” | |  |
|  | | 1. Sistema abre um texto mais detalhado sobre o serviço selecionado, mostrando imagens de sistemas que foram adaptados. |

Fonte: elaboração própria (2024).

Tabela 11 - Caso de uso "Gerenciar Orçamento”

|  |  |  |
| --- | --- | --- |
| **Caso de uso** | RF010 – Gerenciar Orçamentos | |
| **Ator Principal** | ADA company | |
| **Ator Secundário** | - | |
| **Pré-condições** | Precisa de acesso de administrador. | |
| **Pós-condições** | - | |
| **Fluxo Principal** | | |
| **Ações do Ator** | | **Ações do Sistema** |
| **GERENCIAR SERVIÇOS** | | |
| 1. Administrador gerencia os orçamentos pendentes e atualiza os dados caso necessário (status, tipo de serviço, prazo de atendimento, representante responsável). | |  |
|  | | 1. Sistema retorna a mensagem “dados atualizados com sucesso” e indica o código do orçamento alterado. |
| **ALTERAR** | | |
| 1. Administrador altera dados do orçamento. | |  |
|  | | 1. Sistema solicita confirmação dos dados alterados. |
| 1. Administrador verifica e confirma os dados. | |  |
|  | | 1. Sistema atualiza os dados da tabela. |
| **EXCLUIR** | | |
| 1. Administrador cancela orçamento gerado. | |  |
|  | | 1. Sistema solicita uma confirmação. |
| 1. Administrador confirma exclusão. | |  |
|  | | 1. Sistema salva na tabela Orçamentos Cancelado. |

Fonte: elaboração própr

## Requisitos Não Funcionais

Tabela 12 - Requisitos não Funcionais

|  |  |  |
| --- | --- | --- |
| **Nº** | **Nome** | **Descrição** |
| RNF001 | Segurança de acesso. | Deve haver controles de acesso apropriados de login para garantir que apenas usuários autorizados possam acessar determinadas áreas do site. |
| RNF002 | Manutenção do site. | Facilidade de Atualização: O site deve ser projetado para facilitar a implementação de atualizações de software. Monitoramento: Um sistema de monitoramento contínuo deve estar em vigor para identificar e corrigir problemas rapidamente. |
| RNF003 | Banco de Dados MySQL. | Banco de dados onde será armazenado as informações dos clientes, dos seus orçamentos e pedidos. |
| RNF004 | Regulamentação e Conformidade. | Conformidade Legal: O site deve cumprir as leis e regulamentações relevantes, como leis de privacidade e proteção de dados. |
| RNF005 | Ferramenta de desenvolvimento. | Visual Studio, HTML, CSS, Bootstrap, Javascript, MySQL, IDEs como Netbeans e Eclipse para Java. |
| RNF006 | Segurança da Informação: | O site deve ter medidas de segurança, como firewalls e detecção de intrusões, para prevenir ataques cibernéticos. |
| RNF007 | Tolerância a Falhas: | O sistema deve ser capaz de se recuperar de falhas automaticamente, minimizando o impacto nos usuários. |
| RNF008 | Navegação intuitiva | Apresentação muito simples de conteúdos, imagens e botões que facilitem a compreensão e navegação no site. |

Fonte: elaboração própria (2024).

# PROJETO DO SOFTWARE

## Tecnologias Utilizadas

Foi utilizado pesquisas em sites já desenvolvidos para verificar padrões e modelos corporativos, pesquisa de mercado, na qual, foi identificado que há essa necessidade no mercado atual.

JDBC:

[[1]](#footnote-2)A API Java Database Connectivity (JDBC) é o padrão do setor para conectividade independente de banco de dados entre a linguagem de programação Java e uma ampla variedade de bancos de dados (bancos de dados SQL e outras origens de dados tabulares, como planilhas ou arquivos simples). A API JDBC fornece uma API no nível da chamada para acesso ao banco de dados baseado em SQL.

Java Swing:

[[2]](#footnote-3)É uma ferramenta que implementa um conjunto de componentes para a construção de interfaces gráficas de usuário (GUIs) e adiciona funcionalidade gráfica rica e interatividade a aplicações Java. Os componentes do Swing são implementados inteiramente na linguagem de programação Java. A aparência plugável permite criar GUIs que podem ter a mesma aparência em todas as plataformas ou assumir a aparência do sistema operacional atual

GitHub:

[[3]](#footnote-4)GitHub é uma plataforma de hospedagem de código-fonte e colaboração baseada em Git. É amplamente utilizada por desenvolvedores para gerenciar projetos de software de forma colaborativa, permitindo o controle de versão, o rastreamento de problemas e a integração contínua.

Microsoft 365:

[[4]](#footnote-5)O Microsoft 365 é uma suíte de aplicativos e serviços de produtividade baseada em nuvem oferecida pela Microsoft. Inclui aplicativos familiares como Word, Excel e PowerPoint, além de serviços como o Outlook e o OneDrive, permitindo colaboração em tempo real e acesso remoto aos arquivos.

Netbeans:

[[5]](#footnote-6)O NetBeans IDE oferece ferramentas de primeira classe para o desenvolvimento de aplicações Java para web, corporativas, desktops e dispositivos móveis. É sempre o primeiro IDE a suportar as versões mais recentes de JDK, Java EE e JavaFX. Ele fornece visões gerais inteligentes para ajudar você a entender e gerenciar suas aplicações, incluindo suporte imediato para tecnologias populares, como Maven.

Astah:

[[6]](#footnote-7)Astah é uma ferramenta de modelagem e diagramação que permite aos usuários criar diagramas UML (Unified Modeling Language) e outros tipos de diagramas para visualizar e comunicar conceitos de design de software e sistemas.

brModelo:

[[7]](#footnote-8)O brModelo é uma ferramenta específica para modelagem de bancos de dados, permitindo aos usuários criar e visualizar diagramas que representam a estrutura e os relacionamentos dos dados em um banco de dados.

MySQL:

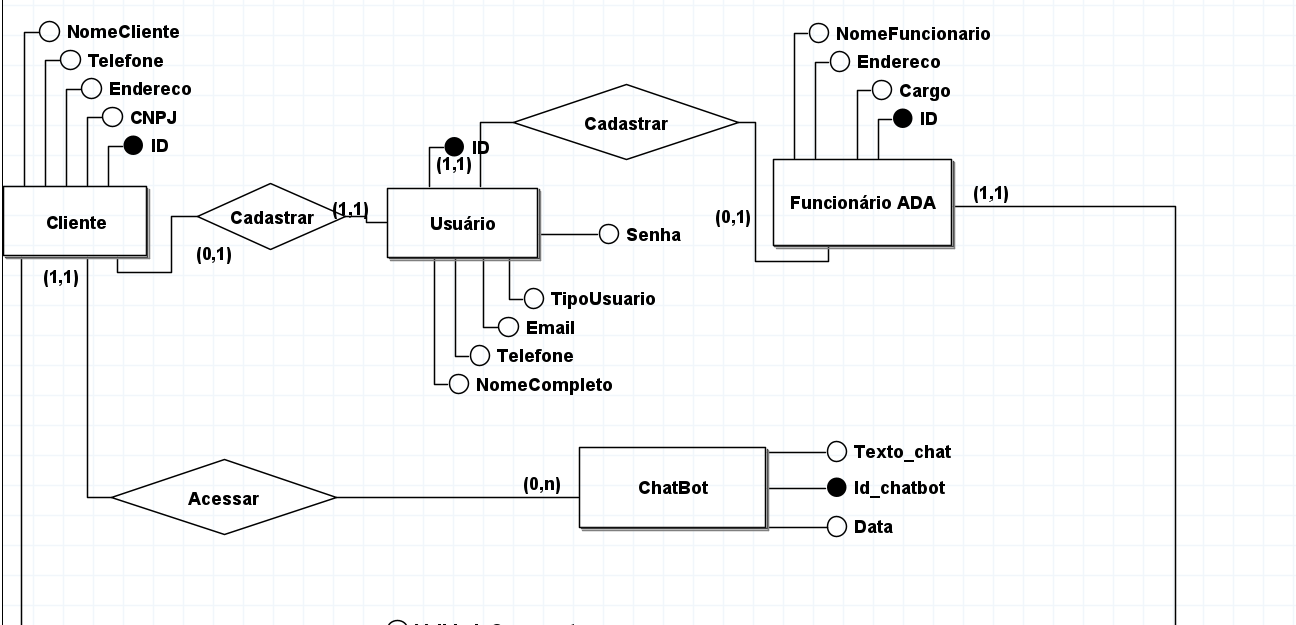
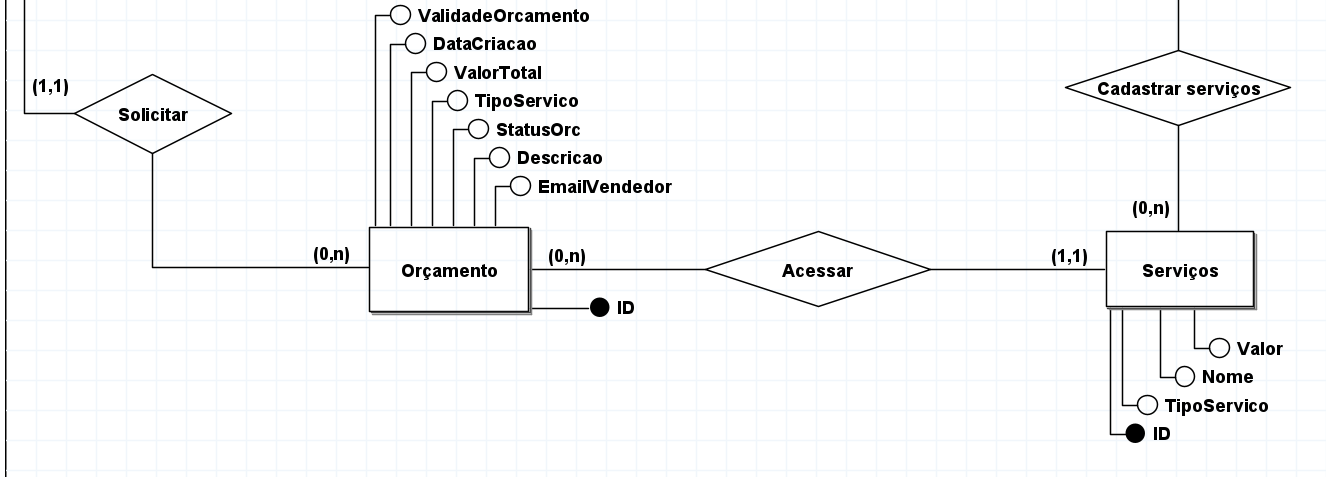
[[8]](#footnote-9)MySQL é um sistema de gerenciamento banco de dados usado para organizar grandes quantidades de dados. Ele é popular na web e em empresas por ser confiável e fácil de usar. Com ele, os desenvolvedores podem armazenar e acessar dados usando consultas de linguagem SQL.

## Modelo de dados

## 3.2.1 Modelo Conceitual

Foi utilizado a ferramenta brModelo para elaboração do modelo conceitual e lógico.

Figura 3 - Modelo Conceitual

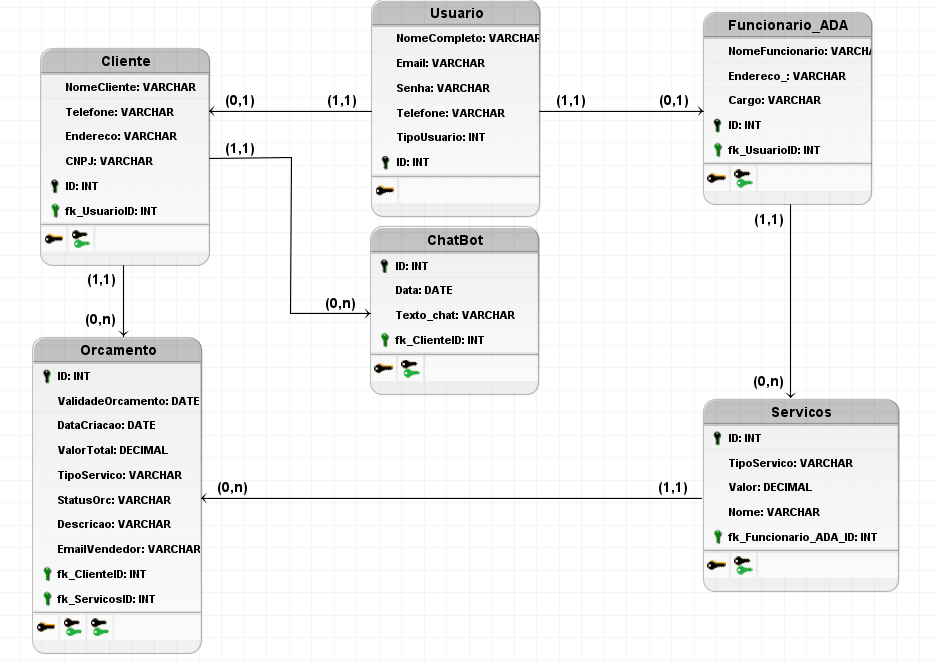


Fonte: elaboração própria com a ferramenta brModelo (2024)

## 3.2.2 Modelo Lógico

Fonte: elaboração própria com a ferramenta brModelo (2024)

Figura 4 - Modelo Lógico



## Diagrama Descrição gerada automaticamenteDiagrama de Classe MVC

Figura 5 - Diagrama de Classe

Diagrama

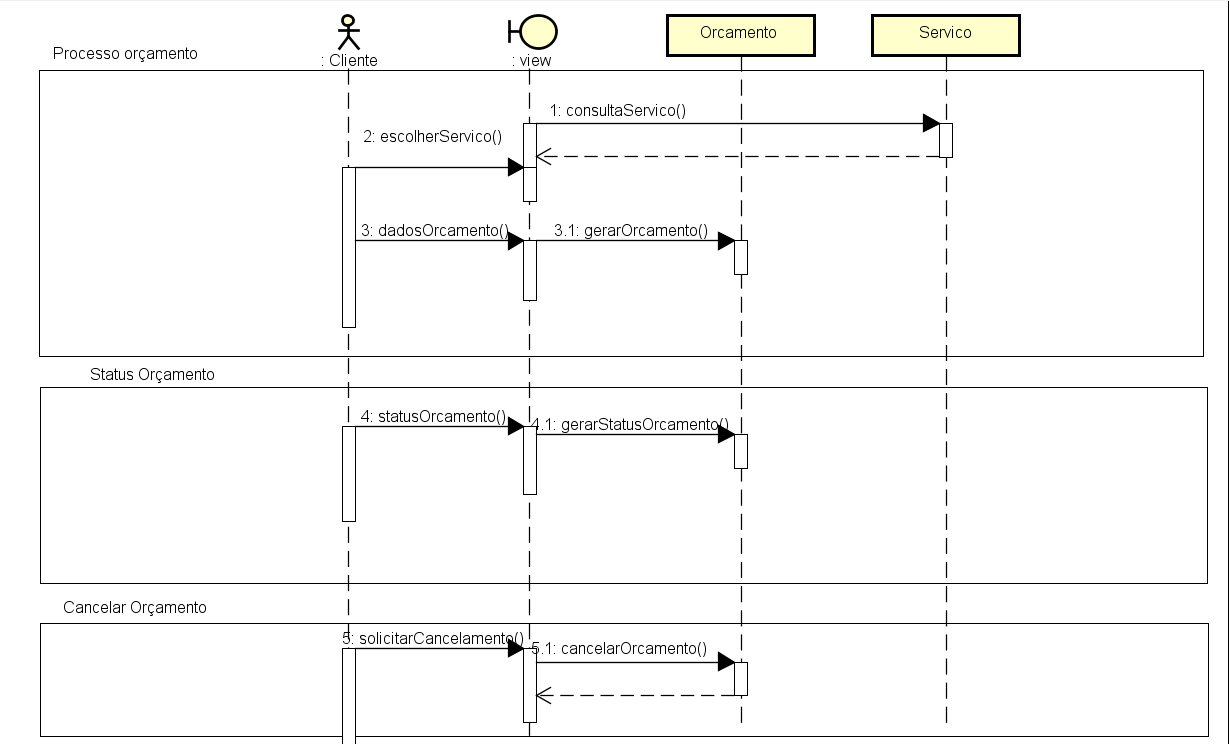
Descrição gerada automaticamente

Fonte: elaboração própria com a ferramenta Astah(2024)

Nas classes view: Login e Cadastro, o método exibeErros() refere-se a tratamento de erros(try e catch) e exibição de mensagens de erro pelo componente JOptionPane.

## Diagrama de Sequência Orçamento

Figura 6 - Diagrama de Sequência



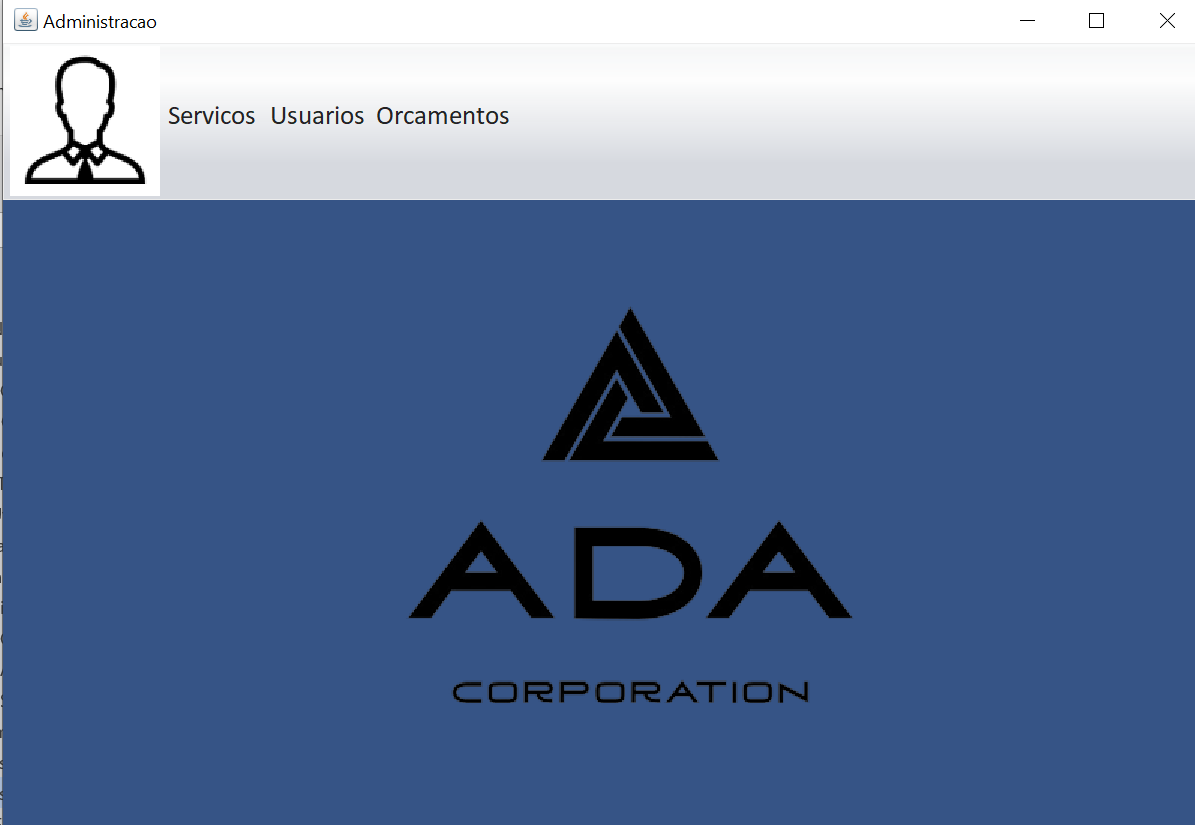
Fonte: elaboração própria com a ferramenta brModelo (2024)

## Interfaces com o usuário

## 3.6.2 Interfaces Desktop

Interfaces em Java Swing para aplicação desktop. Destinada apenas a usuários com privilégios de administração.

Figura 7 - Tela Principal (Administração)



Fonte: elaboração própria (2024)

Figura 8 - Consultar Usuários

Interface gráfica do usuário, Aplicativo

Descrição gerada automaticamente

Fonte: elaboração própria (2024)

Figura 9 - Cadastro de Usuário

Interface gráfica do usuário, Texto, Aplicativo, Email

Descrição gerada automaticamente

Fonte: elaboração própria (2024)

Figura 10 - Editar Perfil

Interface gráfica do usuário, Texto, Aplicativo, Email

Descrição gerada automaticamente

Fonte: elaboração própria (2024)

Figura 11 - Consultar Serviços

Interface gráfica do usuário, Aplicativo

Descrição gerada automaticamente

Fonte: elaboração própria (2024)

Figura 12 - Gerenciar Serviços

Interface gráfica do usuário

Descrição gerada automaticamente

Fonte: elaboração própria (2024)

Figura 13 - Gerenciamento de Orçamentos

Interface gráfica do usuário, Aplicativo

Descrição gerada automaticamente

Fonte: elaboração própria (2024)

Figura 14 - Cadastro de Orçamento

Interface gráfica do usuário, Aplicativo, Teams

Descrição gerada automaticamente

Fonte: elaboração própria (2024)

# IMPLEMENTAÇÃO NO PADRÃO MVC

O projeto foi desenvolvido seguindo o padrão de arquitetura MVC (Model-View-Controller), que divide a aplicação em três componentes principais: Modelo, Visão e Controle. Esta estrutura facilita a manutenção, a escalabilidade do software.

O modelo é responsável por representar a lógica de negócios e os dados da aplicação. Ele interage com o banco de dados e fornece os dados necessários para a visualização.

A Visão é responsável por exibir a interface gráfica ao usuário. Em nosso projeto, utilizamos Java Swing para as interfaces desktop e HTML/CSS/JavaScript para a interface web.

O Controle gerencia a interação entre o Modelo e a Visão. Ele recebe as entradas do usuário, processa a lógica de negócios, interage com os métodos da model e atualiza a Visão.

Um exemplo prático na implementação foi a questão de passar o valor do serviço para o orçamento, o qual em sua fase inicial, possuía seu valor em código. Como em determinado ponto foi implementado o método consultaServico(tipo) em ServicoDAO, o qual o retorna em Objeto Serviço, foi feito um simples aproveitamento desse método em getValorServico(tipo):

new ServicoDAO().consultaServico(tipo).getValor().

## Model

As Classes correspondentes ao model são:

* Usuário (Pessoa), classe abstrata;
* Cliente, extensão de Usuário;
* Funcionário, extensão de Usuário e implementa Administração;
* Administração, interface com os contratos dos métodos;
* Serviço, possui o enum TipoServiço;
* TipoServiço, enum com os valores Adaptação. Desenvolvimento e Consultoria;
* StatusOrçamento, enum com os valores de status de um orçamento.
* Orçamento, classe com os enums StatusOrçamento e TipoServiço;

Além disso, há também a classe Login, no pacote service. Um padrão Singleton, que tem como principal função “guardar” o usuário logado e passá-lo para outras partes do código que necessitem dessa informação: como em editar perfil e gerar orçamento, por exemplo, conforme orientação em sala de aula.

## Controller

As Classes correspondentes ao controller são:

* CadastroController: responsável pela chamada métodos de cadastrar usuários, consultar usuários e editar perfil.
* OrçamentoController: responsável pela chamada dos métodos de gerar orçamento, listar orçamentos, editar dados e alterar status.
* ServicoController: responsável pela chamada dos métodos de alterar serviço (requer administrador logado), visualizar os serviços em lista e consultar serviço de maneira única.
* LoginController: responsável pela chamada dos métodos de validar e realizar o login, logout e retornar o usuário logado para outras partes do sistema.

## View

As Classes correspondentes ao controller são:

* TelaAdm (tela principal): responsável por direcionar os usuários às outras telas;
* CadastroOrçamento: interface para cadastrar um novo orçamento, chama o OrçamentoController, para cadastro, e ServiçoController, para consultar o valor do serviço no banco de dados;
* ConsultarServiços: interface para visualizar os serviços cadastrados no banco de dados e seus dados, chama o ServiçoController;
* CadastroUsuários: interface para cadastrar novos clientes ou administradores, chama o CadastroController;
* ConsultarUsuários: interface para consultar todos os usuários cadastrados no banco de dados ou realizar consultas com base no nome ou e-mail. Chama o CadastroController consultarUsuarios();
* EditarPerfil (usuário logado): interface para edição de dados do perfil do usuário logado. Chama o CadastroController editarPerfil(user);
* GerenciarOrçamentos: interface para consultar todos os orçamentos cadastrados no banco de dados e realizar edições. Chama o OrçamentoController, primeiramente, em listarOrçamentos(). Assim, o administrador pode também utilizar os métodos de alterarStatusOrcamento() e editarOrcamento();
* GerenciarServiços: interface para alterar o valor atual dos tipos de serviços cadastrados (definido como 3 tipos). Chama o ServicoController em consultarServico(), para consultar o valor no BD, e caso for realizada uma alteração, chama o alterarServico();
* Login: interface para realizar login. Chama o LoginController, primeiramente, em validaLogin(), caso verdadeiro, chama o realizarLogin() que instancia o usuário com base nos dados do banco e retorna esse usuário ao Login, que agora pode criar a tela principal e passar o usuário logado ao sistema.

## Outras classes

Além disso, foi criado as classes DAO, isto é, Data Acess Object, responsável por realizar as consultas e edições no banco de dados e retornar, caso necessário, o resultado ou atualização ao sistema.

As classes são:

* ConexaoBD: realiza a conexão com o banco de dados com base no nome do banco, endereço e validação das credenciais.
* OrcamentoDAO;
* ServicoDAO;
* UsuarioDAO (PessoaDAO);

## Implementação dos métodos

O projeto foi totalmente baseado no levantamento de requisitos realizado no início do semestre, porém, foram necessárias algumas mudanças de alterar o retorno do método ou receber certos parâmetros.

Em relação à classe Usuário, ela possui os dados genéricos compartilhados e os getters e setters. Os métodos de acesso ao banco de dados são:

* boolean validaLogin(email, senha), o qual chama a classe DAO para consultar.
* static Usuário realizaLogin(email, senha), o qual chama a classe DAO para retornar um usuário.
* abstract void editarPerfil(usuário), o qual obriga todas as classes filhas a implementar o método.

Em relação à classe Cliente, herda os métodos de Usuário, possui os atributos extras de CNPJ e endereço e seus getters e setters. Implementa o método de editarPerfil(user), o qual chama a classe DAO().editarUsuario(user).

Em relação à classe Funcionário, herda os métodos de Usuário, possui o atributo extra cargo e implementa a interface Administração. Portanto, possui as lógicas de negócio dos métodos:

* ArrayList<Usuario> consultarUsuarios(String nome, int tipoConsulta), o qual chama a classe DAO para consultar com base no nome. O tipo é apenas para a DAO saber se irá consultar com base no NomeUsuario ou Email no banco de dados. Isso é feito na view;
* void alterarServico(tipoServico, valor), o qual chama a classe DAO para realizar um Update no valor, *WHERE* TipoServico for igual ao parâmetro recebido.
* void editarOrcamento(orcID), o qual recebe diversos parâmetros de alteração a classe DAO para realizar um Update nos dados, *WHERE* ID = orcID.
* void alterarStatusOrcamento(ID, int escolha), chama a classe DAO e realiza um Update no status do orçamento com base no ID. A escolha é 1 para ANÁLISE, 2 para CANCELADO, 3 para CONCLUIDO e 4 para DESENVOLVIMENTO (negociação). Exemplo abaixo:

if (escolha == 1)

new OrcamentoDAO().alterarStatusOrcamento("ANALISE", ID);

* void editarPerfil(usuário), chama a classe DAO e realiza um Update no banco de dados em duas tabelas (Usuario e Cliente ou Funcionário). Assim, precisa verificar a instancia do usuário (instanceof).
* void cadastraAdmin(usuario). Apenas um administrador pode cadastrar outro. Esse método chama a DAO.cadastrar(user).

Em relação à classe Serviço, seus métodos são de consultar e visualizar. Essa classe funciona principalmente para a classe de orçamento. Métodos com exceção de getters e setters:

* static Servico consultaServico(tipoServico), o qual chama ServicoDAO e realiza uma consulta no banco de dados para retornar um objeto da classe;
* static ArrayList<Servico> visualizarServicos(input), o qual consulta todos os serviços no banco de dados com base no *input*, caso esteja vazio, irá retornar tudo. Função utilizada na view de Consultar Serviços, a qual usa o componente jTable.

Em relação à classe Orçamento, seus métodos são de gerar novos orçamentos e retornar orçamentos cadastrados. Essa classe possui parâmetros como a data de criação, o enum TipoServiço, o Usuario, valor, o enum Status etc.

Além dos getters e setters, possui os métodos:

* void gerarOrcamento(Orcamento, Usuario), o qual passa para a classe DAO realizar um *insert* no banco de dados.
* ArrayList<Orcamento> listarOrcamentos(), o qual consulta todos os orçamentos cadastrados e retorna em formato de lista. Função utilizada na view de Gerenciar Orçamentos, a qual usa o componente jTable.

No caso da Visão, de maneira resumida, não possui métodos próprios, ela irá apenas processar as informações geradas ou consultar o Controle, que irá validar algumas informações caso necessário e realizar alguma ação chamando a camada Modelo, a camada modelo fica responsável por chamar a DAO e DAO de realizar as ações no banco de dados e/ou retornar o resultado. A depender do resultado, é exibido uma mensagem ao usuário na camada Visão, com o componente JOptionPane.showMessageDialog() ou JOptionPane.showConfirmDialog(). No caso do tratamento de erros, o *try catch* foi realizado apenas na camada de view, dessa forma, as antecessoras estão utilizando *throws Exception* para passar a responsabilidade.

Assim, cada camada do sistema ficou responsável por uma função específica e não tem conhecimento da lógica das outras.

Diagrama

Descrição gerada automaticamente

Fonte: Krasner e Pope (1988).

Figura 15 - Padrão MVC Atual

# ESTRATÉGIA DE TESTES

Na parte da interface Desktop foi utilizado testes unitários com JUnit na implementação de alguns métodos e classes, como instanciar usuário, realizar login, gerar orçamento, alterar dados dos orçamentos e do usuário.

# IMPLANTAÇÃO

O código fonte do website e da interface desktop estarão instanciados no GitHub do nosso projeto:

URL: <<https://github.com/ADACompany01/Projeto>>

# CONCLUSÃO

Este documento detalhou o desenvolvimento bem-sucedido de um software desktop destinado ao uso dos administradores. O software facilita a geração de orçamentos de serviços, o acompanhamento do status dos pedidos, a análise dos serviços prestados e a criação e edição de usuários. O desenvolvimento foi realizado com o objetivo de melhorar a eficiência administrativa do projeto em geral.

Aplicando o modelo MVC, explicamos o desenvolvimento dos diagramas do software e funcionamento das classes e métodos utilizados no projeto. A empresa, que oferece serviços e consultoria em desenvolvimento web, foca na inclusão digital, especialmente para públicos negligenciados, como idosos e pessoas com deficiência.

Em resumo, este projeto atendeu às expectativas, resultando em uma ferramenta eficiente que aprimora a gestão interna do site e reforça o compromisso da empresa com a acessibilidade digital.

# REFERÊNCIAS

SOMMERVILLE, Ian. **Engenharia de Software**. 8ª ed. Pearson, 2007

Guedes, Gilleanes T. A. UML Uma abordagem prática, São Paulo, Novatec Editora Ltda,2004

Paul Deitel, Harvey Deitel - Java como Programar - 10° Edição

**Anexos**

Texto, Carta

Descrição gerada automaticamente

Lousa branca com texto preto sobre fundo branco

Descrição gerada automaticamente com confiança média

**Apêndice – inserir script do banco**

O script do banco de dados e criação das tabelas está com a syntax do mySQL.

Scripts de criação do banco de dados e tabelas abaixo.

CREATE DATABASE IF NOT EXISTS ADACompany;

USE ADACompany;

CREATE TABLE Usuario (

    ID INT PRIMARY KEY AUTO\_INCREMENT,

    NomeCoIDmpleto VARCHAR(100),

    Email VARCHAR(50),

    Senha VARCHAR(50),

    Telefone VARCHAR(11),

    TipoUsuario INT

);

CREATE TABLE Cliente (

    ID INT PRIMARY KEY AUTO\_INCREMENT,

    Telefone VARCHAR(11),

    Endereco VARCHAR(100),

    NomeCliente VARCHAR(100),

    CNPJ VARCHAR(14),

    fk\_UsuarioID INT,

    FOREIGN KEY (fk\_UsuarioID) REFERENCES Usuario(ID)

);

CREATE TABLE Funcionario (

    ID INT PRIMARY KEY AUTO\_INCREMENT,

    NomeFuncionario VARCHAR(100),

    Cargo VARCHAR(50),

    fk\_UsuarioID INT,

    FOREIGN KEY (fk\_UsuarioID) REFERENCES Usuario(ID)

);

CREATE TABLE Orcamento (

    ID INT PRIMARY KEY AUTO\_INCREMENT,

    ValidadeOrcamento DATE,

    DataCriacao DATE,

    ValorTotal DECIMAL(10,2),

    TipoServico VARCHAR(20),

    StatusOrc VARCHAR(15),

    Descricao VARCHAR(500),

    EmailVendedor VARCHAR(100),

    fk\_IDCliente INT,

    fk\_IDCServico INT,

    FOREIGN KEY (fk\_IDServico) REFERENCES Servico(ID),

    FOREIGN KEY (fk\_IDCliente) REFERENCES Cliente(ID)

);

CREATE TABLE Servico (

    ID INT PRIMARY KEY AUTO\_INCREMENT,

    TipoServico VARCHAR(20) NOT NULL,

    Valor DECIMAL(10, 2),

    Nome VARCHAR(20)

);

SELECT \* FROM Orcamento;

/\* Mudanças 23/05: \*/

-- syntax do ID para AUTO\_INCREMENT (mySQL);

-- mudança tipo de CNPJ e Telefone para VARCHAR;

-- adequação datas para o tipo DATE (Java tipo sql.Date);

Mais detalhes no arquivo enviado ao professor de banco de dados.

1. Disponível em: <https://www.oracle.com/br/database/technologies/faq-jdbc.html/>. Acesso em 19 JUN. 2024. [↑](#footnote-ref-2)
2. Disponível em: <https://docs.oracle.com/javase/8/docs/technotes/guides/swing//>. Acesso em: 19 jun. 2023. [↑](#footnote-ref-3)
3. Disponível em: <https://github.com/about>. Acesso em: 09 mar. 2023. [↑](#footnote-ref-4)
4. Disponível em: <https://www.microsoft.com/pt-br/microsoft-365/>. Acesso em: 09 mar. 2023. [↑](#footnote-ref-5)
5. Disponível em: <https://www.oracle.com/br/tools/technologies/netbeans-ide.html/>. Acesso em: 19 jun. 2024. [↑](#footnote-ref-6)
6. Disponível em: <https://astah.net/pt/>. Acesso em: 09 mar. 2023. [↑](#footnote-ref-7)
7. Disponível em: <http://www.sis4.com/brmodelo/>. Acesso em: 09 mar. 2023. [↑](#footnote-ref-8)
8. Disponível em: <https://www.oracle.com/mysql/what-is-mysql/>. Acesso em: 14 abr. 2024. [↑](#footnote-ref-9)