**UNIVERSIDADE DE TAUBATÉ**

**NICKOLAS MARKUS DA SILVA COSTA**

**SISTEMA DE GERENCIAMENTO DE CLIENTES**

**TAUBATÉ - SP**

**2024**

**NICKOLAS MARKUS DA SILVA COSTA**

**SISTEMA DE GERENCIAMENTO DE CLIENTES**

Trabalho de Graduação, modalidade de Trabalho de Conclusão de Curso apresentado ao Departamento de Informática da Universidade de Taubaté para a obtenção do diploma de Tecnólogo em Análise e Desenvolvimento de Sistemas.

Orientador: Prof. Me. Luiz E. Souza Evangelista

**TAUBATÉ - SP**

**2024**

**NICKOLAS MARKUS DA SILVA COSTA**

**SISTEMA DE GERENCIAMENTO DE CLIENTES**

Trabalho de Graduação, modalidade de Trabalho de Conclusão de Curso apresentado ao Departamento de Informática da Universidade de Taubaté para a obtenção do diploma de Tecnólogo em Análise e Desenvolvimento de Sistemas.

Orientador: Prof. Me. Luiz E. Souza Evangelista

Data: \_\_\_\_\_ / \_\_\_\_\_ / \_\_\_\_\_\_

Resultado: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**BANCA EXAMINADORA**

Prof. Me. Luiz Eduardo Souza Evangelista Universidade de Taubaté

Assinatura: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Professor (a): \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Universidade de Taubaté

Assinatura: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Professor (a): \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Universidade de Taubaté

Assinatura: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**AGRADECIMENTOS**

Aos meus pais José Antônio Fernando Costa e Marlinete Lima da Silva pelo apoio incondicional e por terem proporcionado a melhor educação possível.

Ao meu irmão Luiz Henrique da Silva Costa por estar ao meu lado em qualquer situação e pelo incentivo constante.

Ao meu professor orientador Luiz Eduardo Souza Evangelista pelas influentes contribuições, apoio e paciência ao longo da elaboração deste trabalho.

Aos meus amigos Antônio Augusto da Cruz Henrique, Bernardo Marcondes Cesário da Silva, Camila de Paula Elias, Gabriel Oliveira Charleaux, Gabriel Oliveira Pires, Gabriel Pires de Aquino Santos, José Gabriel Alves da Silva, Maria Luiza Corrêa Baracho e Vinícius Leonardo dos Santos pela cooperação mútua na busca pelo conhecimento e pela amizade desenvolvida ao longo dos anos

Aos professores do Departamento de Informática da Universidade de Taubaté pelos conhecimentos compartilhados, pelo apoio e pelo incentivo no decorrer do curso

Aos meus colegas da Hermanos TI por terem acreditado em meu potencial e pela oportunidade para atuar profissionalmente na área de Tecnologia da Informação.

“A programação é a melhor alquimia da mente humana, pois transforma pensamentos em realidade.” (Alan Turing)

**RESUMO**

Na sociedade atual, as relações de consumo estão presentes no cotidiano dela, levando muitas pessoas a adquirir produtos ou serviços dos mais variados tipos, desde os essenciais, como roupas e alimentos, até os supérfluos, que exercem uma função redundante na vida do consumidor. Para ter condições de atender as demandas dos consumidores, as empresas utilizam sistemas em seus processos de atendimento para se organizarem. Porém, muitas delas acabam recorrendo a processos manuais ou utilizam sistemas pagos de forma ilegítima, por razões como falta de conhecimento ou de recursos financeiros. Tendo em vista esse problema, o presente trabalho trata do desenvolvimento de um sistema para gerenciamento dos atendimentos ao cliente, visando, principalmente, as necessidades das pequenas e médias empresas. Este trabalho tem por objetivo auxiliar na organização e gerenciamento de informações de atendimento ao cliente de pequenas e médias empresas por meio de um sistema *web*, além de contribuir para a comunidade de desenvolvimento de *softwares* livres. Este trabalho privilegia uma abordagem metodológica ~~voltada para a pesquisa bibliográfica~~, com foco na realização de uma solução prática para o problema proposto. Por meio dos resultados obtidos (quais?), foi desenvolvido um sistema onde os funcionários dos setores responsáveis pelo atendimento ao cliente da empresa poderão cadastrar novos clientes e consultar aqueles que já foram cadastrados, além de configurar o usuário que eles utilizam. Nesta linha: o que há por detrás da palavra gerenciamento? Com isso, pode-se concluir que o sistema atende os principais requisitos que foram definidos em seu desenvolvimento e, com isso, é capaz de atender as necessidades das organizações na busca pelo melhor gerenciamento dos atendimentos de seus clientes. Em termos de resumo, vai lá, mas “Com isso....” não se desenvolveu, mesmo que em resumo, o que ?

**Palavras-Chave:** ~~Sistemas~~ *~~web~~*~~. Programação~~. Gerenciamento de clientes.

**ABSTRACT**

In today society, consumer relations are present in everyday life, leading many people to purchase products or services of the most varied types, from essentials, such as clothes and food, to superfluous ones, which play a redundant role in the consumer's life. To be able to meet consumer demands, companies use systems in their service processes to organize themselves. However, many of them end up resorting to manual processes or using paid systems illegitimately, for reasons such as lack of knowledge or financial resources. In view of this problem, this work deals with the development of a system for managing customer service, mainly targeting the needs of small and medium-sized companies. This work aims to assist in the organization and management of customer service information for small and medium-sized companies through a web system, in addition to contributing to the free software development community. This work favors a methodological approach focused on bibliographical research, with a focus on achieving a practical solution to the proposed problem. Through the results obtained, a system was developed where employees in the sectors responsible for the company's customer service will be able to register new customers and consult those who have already been registered, in addition to configuring the user they use. With this, it can be concluded that the system meets the main requirements that were defined in its development and, as a result, can meet the needs of organizations in the search for better management of their customers' services.

**Key-words:** Websystems. Programming. Customer management

Depois de acertar o resumo,voltar aqui e corrigir o abstract, isso vale para palavras-chaves

**LISTA DE ILUSTRAÇÕES**

Figura 1 – Rede Cliente-Servidor ............................................................................. 17

Figura 2 – Composição das mensagens em uma transação HTTP ......................... 19

Figura 3 – Cabeçalho de uma mensagem HTTP ..................................................... 20

Figura 4 – Exemplo de algoritmo narrativo ............................................................... 22

Figura 5 – Exemplo de diagrama de fluxo ................................................................ 23

Figura 6 – Exemplo de pseudocódigo ...................................................................... 24

Figura 7 – Representação gráfica de um *array* ........................................................ 25

Figura 8 – Exemplo de lista encadeada ................................................................... 26

Figura 9 – Exemplo de árvore .................................................................................. 28

Figura 10 – Mecanismos de controle de fluxo .......................................................... 38

Figura 11 – Pilares da segurança da informação ..................................................... 41

Figura 12 – Representação de um ataque DDoS .................................................... 44

Figura 13 – Curva de defeito de *software* ................................................................ 45

Figura 14 – Modelo em cascata ............................................................................... 46

Figura 15 – Modelo incremental ............................................................................... 47

Figura 16 – Modelo de reutilização de componentes ............................................... 47

Figura 17 – Representação de um projeto de *software* ........................................... 48

Figura 18 – Arquitetura de chamada e retorno ......................................................... 50

Figura 19 – Arquitetura de fluxo de dados ............................................................... 50

Figura 20 – Arquitetura centralizada em dados ........................................................ 51

Figura 21 – Tipos de leitores de requisitos de *software* ........................................... 52

Figura 22 – Modelo de diagrama de caso de uso .................................................... 54

Figura 23 – Modelo de diagrama de sequência ....................................................... 55

Figura 24 – Modelo de diagrama de classe ............................................................. 55

Figura 25 – *Kanban* do projeto ................................................................................. 60

Figura 26 – Modelo Entidade Relacionamento do banco de dados ......................... 64

Figura 27 – Diagrama de Caso de Uso do sistema .................................................. 68

Figura 28 – Logo do sistema .................................................................................... 69

Figura 29 – Modelagem da interface no Figma ........................................................ 70

Figura 30 – Tela de *login* .......................................................................................... 72

Figura 31 – Tela principal ......................................................................................... 74

Figura 32 – Tela de cadastro .................................................................................... 75

Figura 33 – Formulário de cadastro do atendimento ................................................ 76

Figura 34 – Tela de consulta .................................................................................... 77

Figura 35 – Consulta do atendimento ...................................................................... 78

Figura 36 – Resultado da busca ............................................................................... 79

Figura 37 – Menu lateral ........................................................................................... 80

Figura 38 – Tela de consulta de usuários ................................................................ 81

Figura 39 – Tela de alteração de senha ................................................................... 82

Figura 40 – Tela de alteração de status ................................................................... 83

**LISTA DE TABELAS**

Tabela 1 – Classes de *status* ................................................................................... 20

Tabela 2 – Operadores aritméticos .......................................................................... 31

Tabela 3 – Operadores relacionais .......................................................................... 32

Tabela 4 – Operadores lógicos ................................................................................ 32

Tabela 5 – Requisitos funcionais do sistema ........................................................... 61

Tabela 6 – Requisitos não funcionais do sistema .................................................... 62

**LISTA DE ABREVIATURAS E SIGLAS**

**formatação**

**API** *Application Programming Interface*

**ASCII** *American Standard Code for Information Interchange*

**CERN** Organização Europeia para a Investigação Nuclear

**CPF** Cadastro de Pessoa Física

**CSS** *Cascading Style Sheets*

**DoS** *Denial of Service*

**DDoS** *Distributed Denial of Service*

**FIFO** *First In, First Out*

**FTP** *File Transfer Protocol*

**HTML** *HyperText Markup Language*

**HTTP** *HyperText Transfer Protocol*

**HTTPS** *Secure HyperText Transfer Protocol*

**IDE** *Integrated Development Environment*

**JIT** *Just-in-Time*

**JSON** *Javascript Object Notation*

**LIFO** *Last In, First Out*

**MER** Modelo Entidade Relacionamento

**MIME** *Multipurpose Internet Mail Extensions*

**MIT** *Massachusetts Institute of Technology*

**MVC** *Model-View-Controller*

**PHP** *Hypertext Preprocessor*

**SAC** Serviço de Atendimento ao Consumidor

**SGBD** Sistema de Gerenciamento de Banco de Dados

**SQL** *Structured Query Language*

**SMTP** *Simple Mail Transfer Protocol*

**TCP** *Transfer Control Protocol*

**UML** *Unified Modeling Language*

**URL** *Uniform Resource Locator*

**W3C** *World Wibe Web Consortium*

**XML** *Extensible Markup Language*

**SUMÁRIO**

**1 INTRODUÇÃO** .............................................................................................. 14

**2 FUNDAMENTAÇÃO TEÓRICA** .................................................................... 16

2.1 *WORLD WIDE WEB* ...................................................................................... 16

2.1.1 Protocolo HTTP ............................................................................................. 18

2.2 ALGORITMOS ............................................................................................... 21

2.2.1 Estrutura de Dados ........................................................................................ 24

2.2.1.1 *Arrays* .................................................................................................. 25

2.2.1.2 Lista Encadeada ................................................................................. 26

2.2.1.3 Pilha .................................................................................................... 26

2.2.1.4 Fila ...................................................................................................... 27

2.2.1.5 Árvores ................................................................................................ 27

2.3 LINGUAGENS DE PROGRAMAÇÃO ........................................................... 28

2.3.1 Sintaxe ........................................................................................................... 29

2.3.2 Elementos de uma Linguagem de Programação .......................................... 29

2.3.2.1 Tipos de Dados ................................................................................... 29

2.3.2.2 Operadores ......................................................................................... 30

2.3.2.3 Palavras Reservadas .......................................................................... 33

2.3.2.4 Variáveis ............................................................................................. 34

2.3.2.5 Estruturas de Decisão ......................................................................... 35

2.3.2.6 Estruturas de Repetição ..................................................................... 35

2.3.3 Paradigmas de Programação ........................................................................ 36

2.3.3.1 Programação Imperativa ..................................................................... 37

2.3.3.2 Programação Estruturada ................................................................... 37

2.3.3.3 Programação Funcional ...................................................................... 38

2.3.3.4 Programação Orientada a Objetos ..................................................... 39

2.4 SEGURANÇA DA INFORMAÇÃO ................................................................. 39

2.4.1 *Hacking* .......................................................................................................... 41

2.4.1.1 SQL *Injection* ...................................................................................... 42

2.4.1.2 *Session Hijacking* ................................................................................ 42

2.4.1.3 Negação de Serviço ............................................................................ 43

2.5 ENGENHARIA DE *SOFTWARE* .................................................................... 44

2.5.1 Processos de Desenvolvimento .................................................................... 46

2.5.2 Projeto de Arquitetura .................................................................................... 47

2.5.2.1 Arquitetura em Camadas .................................................................... 49

2.5.2.2 Arquitetura de Chamadas e Retornos ................................................ 49

2.5.2.3 Arquitetura Centralizada em Dados .................................................... 50

2.5.2.4 Arquitetura de Fluxo de Dados ........................................................... 51

2.5.3 Engenharia de Requisitos .............................................................................. 51

2.5.4 Modelagem de Sistemas ............................................................................... 53

2.5.4.1 *Unified Modeling Language* ................................................................ 53

**3 DESENVOLVIMENTO DO SISTEMA** ........................................................... 57

3.1 VISÃO GERAL DO PROJETO ...................................................................... 57

3.1.1 Escopo ........................................................................................................... 57

3.1.2 Planejamento ................................................................................................. 58

3.1.2.1 *Kanban ................................................................................................* 58

3.1.3 Requisitos Funcionais .................................................................................... 61

3.1.4 Requisitos Não Funcionais ............................................................................ 62

3.2 AMBIENTE DE DESENVOLVIMENTO ......................................................... 62

3.2.1 Banco de Dados ............................................................................................ 63

3.2.2 *Integrated Development Environment* ........................................................... 65

3.2.3 Versionamento de Código ............................................................................. 66

3.3 LÓGICA DO SISTEMA .................................................................................. 66

3.4 INTERFACE DE USUÁRIO ........................................................................... 68

**4 RESULTADOS OBTIDOS** ............................................................................. 72

4.1 TELA DE *LOGIN* ............................................................................................ 72

4.2 TELA PRINCIPAL .......................................................................................... 73

4.3 TELA DE CADASTRO ................................................................................... 74

4.4 TELA DE CONSULTA ................................................................................... 76

4.5 OUTRAS FUNCIONALIDADES ..................................................................... 79

**5 CONSIDERAÇÕES FINAIS** .......................................................................... 84

**REFERÊNCIAS BIBLIOGRÁFICAS** ............................................................. 85

**1 INTRODUÇÃO**

O consumo é uma das principais atividades econômicas presentes no modelo social em vigor atualmente, sendo muito presente em economias capitalistas, como os Estados Unidos, a Inglaterra e o Japão.

No Brasil, também considerado uma economia capitalista, as relações de consumo estão muito presentes na sociedade. De acordo com uma notícia divulgada pela Agência Brasil (2023), o consumo nos lares brasileiros encerrou o ano de 2022 com alta de 3,89%. Em datas comemorativas, os consumidores brasileiros tendem a gastar mais, como ocorreu na semana do Dia das Mães (considerada a data comemorativa mais lucrativa pelos comerciantes) do ano de 2023, onde as vendas no setor de varejo cresceram 3,7% em relação às semanas anteriores, segundo uma notícia divulgada pelo jornal Poder 360 (2023).

Por conta disso, as empresas, principais fornecedores dos produtos e serviços adquiridos pelos consumidores, precisam se organizar para atender as demandas dos clientes e para gerenciar seus recursos, a fim de evitar prejuízos financeiros e de imagem. Muitas delas delegam a tarefa de atender o cliente ao Serviço de Atendimento ao Consumidor (SAC) ou ao setor comercial da organização.

Para ter um melhor gerenciamento desses recursos e demandas, as empresas recorrem a sistemas de informação disponibilizados por meio de computadores e que podem ser acessados pela *Web* ou por meio de aplicativos *desktop*. Porém, muitas delas acabam recorrendo a soluções manuais, como anotações em papel, ou a pirataria de *softwares* pagos, por não terem recursos financeiros suficientes para adquiri-los ou pela falta de conhecimento dos gestores de soluções alternativas gratuitas.

Sendo assim, decidiu-se criar um sistema para gerenciar os atendimentos aos clientes, sendo ele voltado para pequenas e médias empresas, a fim de auxiliar na organização e gerenciamento de informações de atendimento ao cliente por meio de um sistema *web*, além de poder contribuir com a comunidade de desenvolvimento de *software open-source*.

A metodologia utilizada na elaboração deste trabalho consiste na pesquisa bibliográfica, com foco no desenvolvimento de uma solução prática. Segundo Gil (2002), as razões que determinam a execução de uma pesquisa podem ser divididas em razões de ordem intelectual, que são uma consequência do desejo de conhecimento, e razões de ordem prática, que decorrem de uma busca por fazer algo de maneira mais eficiente.

1.x Estrutura do trabalho

O presente trabalho está estruturado em capítulos, sendo o capítulo presente apresentando uma introdução ao tema e a organização do trabalho.

No capítulo 2, são abordados fundamentos teóricos que serviram como base para o estudo e desenvolvimento do sistema.

No capítulo 3, é demonstrado o processo de desenvolvimento do *software*, bem como as tecnologias utilizadas nesse processo.

No capítulo 4, com o desenvolvimento do sistema concluído, é apresentado sua versão final e orientações de como utilizá-lo.

No capítulo 5, são feitas as considerações finais do trabalho, bem como sugestões para futuras contribuições.

**2 FUNDAMENTAÇÃO TEÓRICA**

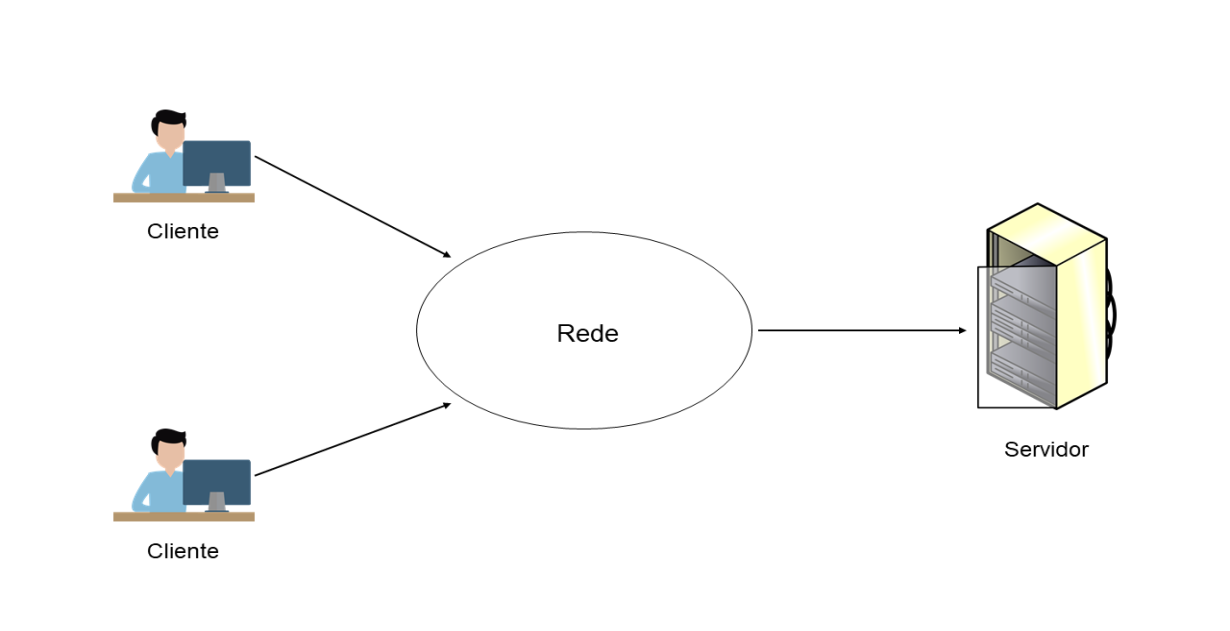
Para que o processo de desenvolvimento do sistema seja feito de maneira clara e objetiva, é necessário a compreensão de fundamentos teóricos que envolva a programação de sistemas *web* e o planejamento de projetos de produção de *softwares*.

~~2.1~~ *~~WORLD WIDE WEB~~*

~~A~~ *~~World Wide Web~~* ~~é um sistema que permite o acesso a documentos em hipermídia, que são vinculados entre si e sendo executados na~~ *~~Internet~~* ~~através de um programa chamado de navegador. Esses documentos, de acordo com Forouzan (2010), são distribuídos em diversos locais, chamados de~~ *~~sites~~*~~, onde os mesmos são responsáveis pelo gerenciamento desses documentos, conhecidos pelo nome de página~~ *~~Web~~*~~.~~

~~A~~ *~~Web,~~* ~~conforme Tanenbaum~~ *~~et al.~~* ~~(2021), teve seu início em 1989 a partir de um protótipo criado por Tim Berners-Lee na Organização Europeia para Investigação Nuclear (CERN). Após seu lançamento, em 1993, juntamente com o navegador~~ *~~Mosaic,~~* ~~esse sistema teve um aumento exponencial, tanto de quantidade de conteúdo disponível quanto de número de usuários, de acordo com Tanenbaum~~ *~~et al.~~* ~~(2021). Logo após a CERN, juntamente com o~~ *~~Massachusetts Institute of Technology~~* ~~(MIT), fundou uma organização voltada para o desenvolvimento e padronização da~~ *~~Web~~*~~, na qual foi nomeada como~~ *~~World Wide Web Consortium~~* ~~(W3C).~~

~~O modelo de rede utilizado pela~~ *~~Web~~* ~~para disponibilizar os conteúdos é o cliente-servidor que, segundo o Serviço Nacional de Aprendizagem Industrial (2012), consiste na função do servidor de fornecer serviços ou recursos da rede ao cliente, no qual solicita esses serviços por meio de requisições. A figura 1 demonstra o modelo de rede cliente-servidor.~~

~~Figura 1 – Rede Cliente-Servidor~~

~~Fonte: Adaptado de Tanenbaum~~ *~~et al.~~* ~~(2021)~~

~~Para que o cliente possa acessar uma página~~ *~~Web~~*~~, é necessário que ele informe o endereço dessa página por meio da~~ *~~Uniform Resource Locator~~* ~~(URL), que é um localizador utilizado como padrão para buscas de informações na~~ *~~Internet~~*~~. Conforme Forouzan (2010), a URL é composta por quatro partes:~~

* ~~Protocolo: é o programa utilizado para acessar os documentos na~~ *~~Web~~*~~, atuando, principalmente, na comunicação entre a máquina cliente e o servidor;~~
* *~~Host:~~* ~~é o computador onde as informações são armazenadas, podendo receber nomes alternativos que iniciam com os caracteres www;~~
* ~~Porta: segundo Galvão (2024), porta é o ponto final em uma conexão de rede, na qual representa uma conexão lógica para que ocorra a troca de informações. Em uma URL, a porta é localizada entre o~~ *~~host~~* ~~e o caminho, sendo separada por dois pontos;~~
* ~~Caminho (~~*~~path~~*~~): ele informa a localização do arquivo que armazena as informações disponibilizadas para o cliente.~~

~~As páginas~~ *~~Web~~* ~~precisam ser armazenadas em um servidor para que elas possam estar acessíveis ao cliente, que precisa enviar requisições ao servidor para possuir esse acesso. Segundo Forouzan (2010), para melhorar a eficiência, os servidores armazenam informações na área de cache da memória, que é mais rápida se comparada com a memória dos discos rígidos. Além disso, alguns servidores utilizam técnicas como multiprocessamento ou multitarefa baseada em~~ *~~threads~~* ~~para responder mais de uma requisição por vez.~~

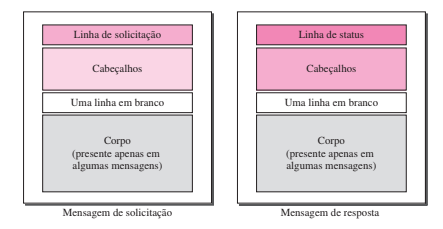
~~2.1.1 Protocolo HTTP~~

~~Segundo Tanenbaum~~ *~~et al.~~* ~~(2021), ao exibir uma página~~ *~~Web~~* ~~na máquina do cliente de acordo com o que foi solicitado ao servidor, é necessário de um protocolo nessa comunicação, sendo o HTTP (~~*~~HyperText Transfer Protocol~~*~~)~~~~o protocolo utilizado para o modelo cliente-servidor. Há ainda a versão segura desse protocolo, o HTTPS (~~*~~Secure HyperText Transfer Protocol~~*~~), sendo essa a mais utilizada atualmente em buscas na~~ *~~World Wide Web.~~*

~~De acordo com Forouzan (2010), o HTTP utiliza como base uma combinação de outros protocolos, que são:~~

* *~~File Transfer Protocol~~* ~~(FTP): permite a transferência de arquivos entre o cliente e o servidor e utiliza serviços do protocolo TCP (~~*~~Transmission Control Protocol~~*~~) por meio de uma única conexão;~~
* *~~Simple Mail Transfer Protocol~~* ~~(SMTP): os dados transferidos entre o cliente e o servidor são parecidos com mensagens SMTP, com a diferença que, no protocolo HTTP, as mensagens são lidas e interpretadas pelo servidor e pelo cliente. Além disso, elas são encapsuladas em cabeçalhos que se assemelham aos do formato MIME (~~*~~Multipurpose Internet Mail Extensions~~*~~).~~

~~Conforme Forouzan (2010), ao enviar uma requisição, por meio de uma mensagem de solicitação, ao servidor, o cliente inicia uma transação HTTP. Embora o HTTP seja essencialmente um protocolo sem estados (~~*~~stateless~~*~~), o servidor encaminha uma mensagem de resposta. Essas mensagens são compostas pela linha de solicitação ou de~~ *~~status~~*~~, pelo cabeçalho e pelo corpo da mensagem. A figura 2 demonstra como são compostas as mensagens em uma transação HTTP.~~

~~Figura 2 – Composição das mensagens na transação HTTP~~

~~Fonte: Forouzan (2010)~~

~~A linha de solicitação, como definido por Forouzan (2010), é separada por três partes, onde a primeira exibe o tipo de solicitação, a segunda é composta pela URL e a terceira demonstra a versão do HTTP, sendo a 1.1 a mais atual até o momento. Quanto ao tipo de solicitação, de acordo com Monari (2016), são compostos por métodos, sendo que os mais comuns são:~~

* ~~GET: é utilizado para receber informações;~~
* ~~POST: é utilizado quando se é necessário adicionar informações novas à URL;~~
* ~~PUT: é utilizado para atualizar as informações que estão presentes em uma URL;~~
* ~~DELETE: é utilizado para solicitar ao servidor que exclua a informação da URL.~~

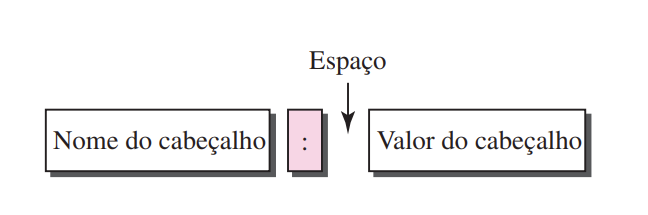
~~Já a linha de~~ *~~status~~*~~, conforme Forouzan (2010), também é dividida em 3 partes, porém a primeira é composta pela versão do HTTP, a segunda exibe o código do~~ *~~status~~* ~~da requisição e a terceira apresenta uma frase de~~ *~~status~~*~~. Segundo Monari (2016), os~~ *~~status~~* ~~é uma forma simples de verificar o que aconteceu com a requisição para o cliente que recebe a resposta, sendo separados por classes que contém o código e uma frase informativa. Existem diversas classes de~~ *~~status~~*~~, e as que estão mais presentes nas respostas são demonstradas na tabela 1.~~

~~Tabela 1 – Classes de Status~~

|  |  |  |
| --- | --- | --- |
| **~~Código~~** | **~~Frase~~** | **~~Descrição~~** |
| ~~200~~ | ~~OK~~ | ~~A solicitação obteve êxito~~ |
| ~~301~~ | *~~Moved Permanently~~* | ~~A URL solicitada não está mais disponível~~ |
| ~~304~~ | *~~Moved Temporarily~~* | ~~A URL solicitada foi movida de forma temporária~~ |
| ~~400~~ | *~~Bad Request~~* | ~~Ocorreu um erro de sintaxe na solicitação~~ |
| ~~401~~ | *~~Unauthorized~~* | ~~A solicitação não tem autorização para ser executada~~ |
| ~~403~~ | *~~Forbidden~~* | ~~A solicitação foi negada pelo servidor~~ |
| ~~404~~ | *~~Not Found~~* | ~~O documento solicitado não foi encontrado~~ |
| ~~500~~ | *~~Internal Server Error~~* | ~~Ocorreu um erro no servidor, impedindo o acesso ao documento solicitado~~ |

~~Fonte: Adaptado de Forouzan (2010)~~

~~O cabeçalho, de acordo com Forouzan (2010), permite que o cliente e o servidor troquem informações adicionais a respeito da solicitação e da resposta. Ele pode ser formado por uma ou mais linhas, que são constituídas pelo nome e valor do cabeçalho, separados por dois-pontos e um espaço em branco, conforme mostrado na figura 3.~~

~~Figura 3 – Cabeçalho de uma mensagem HTTP~~

~~Fonte: Forouzan (2010)~~

~~Forouzan (2010) afirma que em uma linha de cabeçalho, seja ela pertencendo a uma solicitação ou a uma resposta, é necessário que a mesma contenha as seguintes categorias:~~

* ~~Cabeçalho geral: esse tipo de cabeçalho fornece informações que demonstram uma visão geral do assunto da mensagem. Pode ser utilizado tanto para solicitação quanto para respostas;~~
* ~~Cabeçalho de solicitação: fornece informações específicas da configuração da máquina cliente e o formato de documentos que o mesmo possui preferência. Por conta disso, só pode ser utilizado por mensagens de solicitação;~~
* ~~Cabeçalho de resposta: fornece informações extras da solicitação, além de informações do servidor. Pode ser utilizado em mensagens de resposta;~~
* ~~Cabeçalho de entidade: fornece informações relacionadas ao corpo do documento. Embora seja utilizado, em sua maioria, nas mensagens de resposta, algumas mensagens de solicitação utilizam esse cabeçalho, principalmente aquelas que utilizam os métodos POST e PUT.~~

~~O corpo da mensagem de solicitação ou de resposta recebe os dados a serem transmitidos na mesma, podendo ser um documento ou um outro tipo de dado.~~

~~2.2 ALGORITMOS~~

~~Um algoritmo é um conjunto de regras que descrevem os passos necessários para a resolução de um determinado problema. Dias (2023) compara um algoritmo a uma receita, onde para se cozinhar um determinado alimento, é necessário que o cozinheiro siga um processo para que, dessa forma, o resultado final seja conforme descrito na receita.~~

~~O termo algoritmo é aplicado em duas áreas: na Matemática e na Ciência da Computação. Conforme explicado por Manzano e Oliveira (2019), o algoritmo está associado a um processo de cálculo que produz uma solução para um problema por meio de um número finito de etapas, enquanto na Ciência da Computação o algoritmo representa um conjunto de procedimentos lógicos perfeitamente definidos que levam para uma solução de um problema.~~

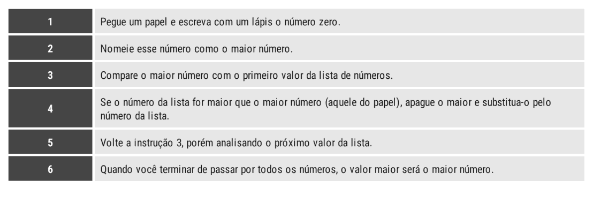
~~Na Ciência da Computação, os algoritmos precisam seguir alguns critérios para serem considerados legíveis e eficazes. Conforme listado por Dias (2023), são eles:~~

* ~~Objetividade: os algoritmos computacionais necessitam ser precisos e objetivos, evitando ambiguidade em sua escrita;~~
* ~~Entrada: um algoritmo precisa receber um valor de entrada;~~
* ~~Saída: após o processamento do algoritmo, o mesmo deve produzir um valor de saída;~~
* ~~Finito: o algoritmo deve possuir uma série finita de etapas e terminar o seu processamento após sua conclusão;~~
* ~~Eficácia: um algoritmo computacional deve ter a capacidade de processamento em um tempo finito utilizando ferramentas básicas.~~

~~Dias (2023) afirma que um algoritmo computacional pode ser escrito de diversas maneiras, e os mais utilizados pela Ciência da Computação são o narrativo, o diagrama de fluxo e o pseudocódigo.~~

~~O algoritmo narrativo é escrito seguindo uma narrativa, podendo utilizar uma tabela ou uma lista para construir essa narrativa, conforme ilustra a figura 4. O problema de utilizar esse tipo de escrita, de acordo com Dias (2023), é que ela pode se tornar ambígua, o que contradiz com o critério de objetividade de um algoritmo.~~

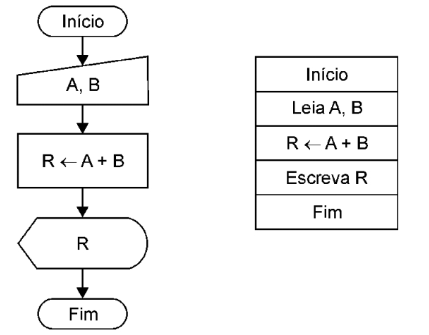
~~Figura 4 – Exemplo de algoritmo narrativo~~

~~~~

~~Fonte: Dias (2023)~~

~~Para resolver o problema da objetividade no algoritmo narrativo, é comumente utilizado outras duas formas, que são o diagrama de fluxo e o pseudocódigo. Manzano e Oliveira (2019) define um diagrama de fluxo como uma representação gráfica de um algoritmo onde essa representação é composta por formas geométricas. No entanto, Manzano e Oliveira (2019) afirmam que essa forma de representar o algoritmo é utilizada apenas na esfera de análise de sistemas, enquanto a esfera da programação utiliza o pseudocódigo como forma de escrever um algoritmo. A figura 5 mostra um exemplo de diagrama de fluxo.~~

~~Figura 5 – Exemplo de diagrama de fluxo~~

~~~~

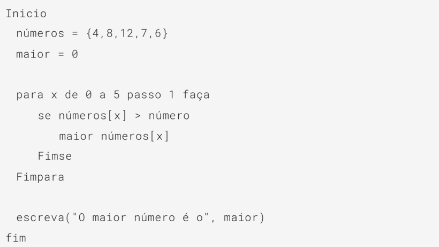
~~Fonte: Manzano e Oliveira (2019)~~

~~No processo de desenvolvimento, o pseudocódigo é utilizado para representar textualmente um algoritmo. O pseudocódigo, também chamada de Linguagem de Projeto de Programação, possui comandos escritos no idioma local e uma forma de representação de caracteres mais simplificadas, conforme explicado por Manzano e Oliveira (2019). Além disso, o pseudocódigo não pode ter o mesmo rigor sintático de uma Linguagem de Programação Formal, tendo em vista que ele não é considerado uma linguagem de programação, e sim uma forma de representar um algoritmo.~~

~~Como os comandos do pseudocódigo são escritos em idioma local, uma das características dele é o regionalismo, conforme dito por Manzano e Oliveira (2019). No Brasil, o pseudocódigo é conhecido por Portugol ou Português Estruturado.~~

~~A técnica de utilizar pseudocódigos, segundo Manzano e Oliveira (2019), é herança de uma outra técnica de programação, chamada~~ *~~Program Design Language~~*~~. Essa técnica, proposta por Caine e Gordon em 1975, tem como objetivo escrever um programa de computador sem se preocupar com comandos específicos e sintaxe da linguagem de programação a ser utilizada no projeto. A figura 6 mostra um exemplo de pseudocódigo.~~

~~Figura 6 – Exemplo de pseudocódigo~~

~~~~

~~Fonte: Dias (2023)~~

~~2.2.1 Estrutura de Dados~~

~~Em um programa de computador, a existência de apenas um algoritmo não é suficiente. É necessário que se utilizem vários algoritmos, cada um desempenhando uma determinada função no programa, sendo o gerenciamento de recursos do computador a mais essencial de todas. Porém, para que essa função seja feita da melhor maneira possível, é utilizado um conceito muito importante na Ciência da Computação, que é a Estrutura de Dados.~~

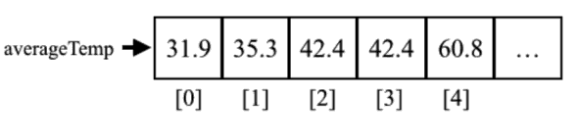
~~Cormen~~ *~~et al.~~* ~~(2002) define Estrutura de Dados como um recurso para armazenar e organizar dados na memória do computador com o objetivo de facilitar o acesso e as modificações desse armazenamento.~~

~~Conforme Ascencio e Araújo (2010), a escolha de um algoritmo para resolver um problema computacional também irá depender do tipo de estrutura que será utilizada para armazenar os dados, pois não é possível separar a estrutura para armazenar os dados do algoritmo. Há diversos tipos de estrutura de dados, cada uma com sua peculiaridade, sendo as comumente utilizadas no desenvolvimento de sistemas de computador:~~ *~~Arrays~~*~~, Listas Encadeadas, Filas, Pilhas e Árvores Binárias.~~

~~2.2.1.1~~ *~~Arrays~~*

~~Um~~ *~~array~~* ~~é o tipo de estrutura de dados mais simples para armazenar dados em memória, conforme definição por Groner (2018). Ele armazena de forma sequencial dados de um mesmo tipo, conforme ilustrado na figura 7. Por ser uma estrutura de dados simples, todas as linguagens de programação possuem um tipo de dado~~ *~~array~~* ~~incluído em sua sintaxe, possuindo apenas diferenças técnicas particulares da linguagem utilizada.~~

~~Figura 7 – Representação gráfica de um~~ *~~array~~*

~~~~

~~Fonte: Groner (2018)~~

~~2.2.1.2 Lista Encadeada~~

~~Segundo Goodrich e Tamassia (2007), uma lista encadeada é uma coleção de nós que formam uma ordem linear quando estão ligados. Um nó, no contexto da Ciência da Computação, seria um elemento que representa a interconexão da estrutura. O primeiro e o último nó de uma lista são comumente conhecidos como cabeça (~~*~~head~~*~~) e cauda (~~*~~tail~~*~~). A figura 8 representa uma lista encadeada de forma gráfica.~~

~~Figura 8 – Exemplo de lista encadeada~~

~~~~

~~Fonte: Goodrich e Tamassia (2007)~~

~~Ascencio e Araújo (2010) afirmam que as listas encadeadas podem ser classificadas em dois tipos principais, sendo eles:~~

* ~~Lista simplesmente encadeada: nessa estrutura, cada nó armazena um ou mais dados e um ponteiro para o próximo nó;~~
* ~~Lista duplamente encadeada: já nesse tipo, cada nó armazena um ou mais dados e dois ponteiros, sendo um apontando para o nó anterior e o outro para o próximo.~~

~~2.2.1.3 Pilha~~

~~Conforme definido por Goodrich e Tamassia (2007), uma pilha é uma coleção de objetos onde a manipulação dos dados segue o princípio de que o último que entra é o primeiro que irá sair (do inglês~~ *~~Last In, First Out~~* ~~– LIFO).~~

~~Ainda de acordo com Goodrich e Tamassia (2007), essa estrutura permite que objetos possam ser inseridos na pilha a qualquer momento, porém somente o mais recente sairá da pilha a qualquer momento.~~

~~2.2.1.4 Fila~~

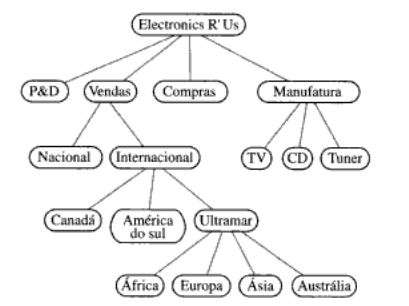
~~Segundo a definição de Goodrich e Tamassia (2007), a fila é uma estrutura de dados similar à pilha, pois também é uma coleção de objetos. Porém, diferente de sua “prima”, a fila segue o princípio de que o primeiro objeto que entra será o primeiro a sair (do inglês~~ *~~First In, First Out~~* ~~– FIFO). Nesse caso, o acesso aos elementos e a remoção dos mesmos fica restrito ao primeiro objeto da estrutura.~~

~~2.2.1.5 Árvores~~

~~No contexto da Ciência da Computação, uma árvore é um tipo de estrutura de dados que organiza os objetos de maneira não-linear. Segundo a explicação de Goodrich e Tamassia (2007), isso permite a implementação de mais algoritmos, ocasionando um processamento mais rápido do que as listas encadeadas.~~

~~Uma árvore é organizada de forma hierárquica, onde é construindo partindo do elemento do topo, denominado de raiz, até chegar no último elemento da estrutura. Cada elemento da árvore, exceto o nó raiz, possui um elemento pai e zero ou mais elementos filhos, sendo denominados como nós folhas, conforme ilustrado na figura 9.~~

~~Figura 9 – Exemplo de árvore~~

~~~~

~~Fonte: Goodrich e Tamassia (2007)~~

~~2.3 LINGUAGENS DE PROGRAMAÇÃO~~

~~Para que o computador possa executar um~~ *~~software~~*~~, é necessário que haja comunicação entre a aplicação e o~~ *~~hardware~~*~~. Para isso, são utilizadas as linguagens de programação, nas quais irão enviar comandos para a máquina através de um código-fonte.~~

~~Para Melo e Silva (2014), uma linguagem de programação é composta por um conjunto de recursos disponíveis e na maneira que serão utilizados para construir máquinas abstratas, que são os programas de computador. Além disso, elas possuem um conjunto de regras de composição que garantem a implementação do programa nos computadores com uma qualidade apropriada para aquela determinada máquina.~~

~~Melo e Silva (2014) listam alguns itens que precisam ser levados em consideração no momento de projetar uma determinada linguagem de programação. São eles:~~

* ~~Requisitos: deve ser analisado o conjunto de problemas que a linguagem pretende resolver;~~
* ~~Expressividade: momento onde é definida a forma que os elementos da linguagem serão representados da maneira mais natural possível;~~
* ~~Paradigma: será analisada a melhor maneira de abstrair os problemas que a linguagem pretende resolver;~~
* ~~Implementação: os requisitos da linguagem devem ser passíveis de implementação;~~
* ~~Eficiência: a linguagem, bem como os seus requisitos, deve ter um patamar de eficiência aceitável.~~

~~2.3.1 Sintaxe~~

~~Em uma linguagem de programação, a sintaxe é a principal característica de sua estrutura. Melo e Silva (2014) definem a sintaxe de uma linguagem de programação como a maneira que a mesma tem de manipular os programas de computador. Em outras palavras, a sintaxe da linguagem é a forma como ela é projetada para ser escrita, de maneira que possibilite a interpretação pela máquina.~~

~~Melo e Silva (2014) ainda acrescentam a semântica da linguagem de programação, na qual determina um significado para os elementos da linguagem, onde a junção delas irá compor o conjunto de regras para o funcionamento da linguagem e o entendimento da mesma pelo computador.~~

~~2.3.2 Elementos de uma Linguagem de Programação~~

~~2.3.2.1 Tipos de Dados~~

~~Todo dado manipulado pela linguagem de programação é armazenado em um pedaço da memória do computador. Para determinar a quantidade necessária de~~ *~~bytes~~* ~~que devem ser armazenados, os dados são classificados em tipos que, segundo Alves (2014), representam características da informação armazenada. São eles:~~

* ~~Caractere: esse tipo de dado permite o armazenamento de caracteres e ocupa 1 byte na memória. Utilizam a tabela ASCII como referência;~~
* ~~Inteiro: trabalha com valores numéricos inteiros. Geralmente, armazenam números inteiros em uma faixa que vai de -32768 a 32767 (inteiros curtos) ou de -2.147.483.648 a 2.147.483.647 (inteiros longos);~~
* ~~Ponto flutuante: também trabalha com valores numéricos, com a diferença que esse tipo de dado também abrange números decimais. O armazenamento pode ser feito de visando uma precisão simples, na qual irá ocupar até 32~~ *~~bits~~* ~~de memória, ou uma dupla precisão, onde irá ocupar até 64~~ *~~bits~~* ~~de memória;~~
* ~~Lógico (~~*~~boolean~~*~~): nesse tipo de dado são trabalhos valores lógicos, no qual sempre vão retornar verdadeiro ou falso. Assim como o caractere, ocupa 1 byte na memória.~~

~~Esses tipos de dados, de acordo com Alves (2014), são denominados de tipos de dados primitivos, pois eles vêm definidos nas linguagens de programação e são a base das declarações de variáveis nos programas.~~

~~Existe também dados do tipo~~ *~~string~~*~~, nas quais armazenam uma sequência de caracteres com tamanho variável, podendo ocupar mais ou menos memória. Porém, segundo o W3Schools (2024), nem todas as linguagens consideram~~ *~~string~~* ~~como um tipo de dado. Linguagens como Java consideram-na como um objeto, na qual é definido em uma classe.~~

~~2.3.2.2 Operadores~~

~~Os operadores são conectores que realizam a ligação entre duas ou mais expressões. As linguagens de programação possuem operadores que permitem trabalhar com os dados armazenados na memória através de cálculos, comparações, entre outros.~~

~~Conforme Alves (2014), os operadores são organizados nos seguintes tipos:~~

* ~~Aritméticos: são operadores que permitem realizar cálculos matemáticos, sendo possível a utilização deste por valores numéricos. A tabela 2 demonstra os operadores utilizados;~~

~~Tabela 2 – Operadores Aritméticos~~

|  |  |  |
| --- | --- | --- |
| **~~Operador~~** | **~~Nome~~** | **~~Descrição~~** |
| ~~+~~ | ~~Adição~~ | ~~Efetua a soma de valores numéricos~~ |
| ~~-~~ | ~~Subtração~~ | ~~Realiza a subtração de valores numéricos~~ |
| ~~\*~~ | ~~Multiplicação~~ | ~~Efetua a multiplicação de valores numéricos~~ |
| ~~/~~ | ~~Divisão~~ | ~~Efetua a divisão de valores numéricos e retorna um valor decimal~~ |
| ~~\~~ | ~~Divisão inteira~~ | ~~Realiza a mesma ação do operador de divisão, porém retorna um valor inteiro~~ |
| ~~%~~ | ~~Resto~~ | ~~Retorna o resto de uma divisão~~ |

~~Fonte: Adaptado de Alves (2014).~~

* ~~Relacionais: são operadores que permitem a comparação de duas variáveis ou de dois valores. A tabela 3 demonstra esses tipos de operadores;~~

~~Tabela 3 – Operadores relacionais~~

|  |  |
| --- | --- |
| **~~Operador~~** | **~~Descrição~~** |
| ~~<~~ | ~~Menor que~~ |
| ~~>~~ | ~~Maior que~~ |
| ~~<=~~ | ~~Menor ou igual a~~ |
| ~~>=~~ | ~~Maior ou igual a~~ |
| ~~==~~ | ~~Igual~~ |
| ~~!=~~ | ~~Diferente (dependendo da linguagem, pode ser representado por <>)~~ |

~~Fonte: Adaptado de Alves (2014).~~

* ~~Lógicos: esses operadores são utilizados na construção de expressões lógicas, retornando um valor verdadeiro ou falso após a avaliação da expressão, conforme ilustrado na tabela 4.~~

~~Tabela 4 – Operadores lógicos~~

|  |  |  |
| --- | --- | --- |
| **~~Operador~~** | **~~Nome~~** | **~~Resultado~~** |
| ~~&&~~ | ~~E~~ | ~~Retorna verdadeiro se ambas as expressões forem verdadeiras~~ |
| ~~| |~~ | ~~OU~~ | ~~Retorna verdadeiro se uma das expressões (ou ambas) forem verdadeiras~~ |
| ~~!~~ | ~~NÃO~~ | ~~Efetua a negação de uma expressão~~ |

~~Fonte: Adaptado de Alves (2014).~~

~~Há ainda o operador de atribuição (=), utilizado pelas linguagens de programação para atribuir valores para as variáveis. Alves (2014) ainda cita alguns operadores especiais utilizados por algumas linguagens de programação para manipular os~~ *~~bits~~* ~~de um dado armazenado, sendo os principais:~~

* ~~Deslocamento de~~ *~~bits~~*~~;~~
* ~~Rotação de~~ *~~bits~~*~~.~~

~~2.3.2.3 Palavras Reservadas~~

~~Em linguagem de programação, as palavras reservadas são aquelas que fazem parte de sua estrutura técnica para executar certos comandos. Por conta disso, não podem ser utilizadas na construção do programa para identificar variáveis ou qualquer outra forma de identificação.~~

~~Conforme listado pela IBM (2024), as principais palavras reservadas da linguagem de programação C são:~~

* *~~char:~~* ~~palavra destinada para identificar variáveis do tipo caractere~~*~~;~~*
* *~~int~~*~~: palavra reservada utilizada para identificar as variáveis do tipo inteiro;~~
* *~~float~~*~~: assim como a palavra~~ *~~int~~*~~,~~ *~~float~~* ~~tem a função de identificar as variáveis do tipo flutuante~~*~~;~~*
* *~~void~~*~~: palavra utilizada para identificar métodos ou funções que não irão retornar nenhum valor~~*~~;~~*
* *~~static~~*~~: palavra utilizada para identificar métodos ou classes que não precisam ser instanciadas~~*~~;~~*
* *~~return~~*~~: palavra utilizada para indicar o retorno de uma determinada função;~~
* *~~if~~*~~: palavra utilizada para construir um bloco de estrutura de decisão~~*~~;~~*
* *~~else~~*~~: palavra utilizada em uma estrutura de decisão para indicar o que deve ser feito caso a primeira parte do bloco retorne um valor Falso~~*~~;~~*
* *~~switch~~*~~: assim como a palavra~~ *~~if~~*~~, é utilizada em uma estrutura de decisão~~*~~;~~*
* *~~case~~*~~: utilizada dentro de um comando~~ *~~switch~~*~~, servindo para indicar os casos de condição desse bloco~~*~~;~~*
* *~~break~~*~~: palavra utilizada para indicar uma pausa do comando~~ *~~switch~~*~~. Em algumas linguagens, essa palavra também é utilizada na instrução~~ *~~while;~~*
* *~~for~~*~~: palavra utilizada na construção de laços de repetição;~~
* *~~while:~~* ~~assim como a palavra~~ *~~for~~*~~, também é utilizada na construção de estruturas de repetição.~~

~~2.3.2.4 Variáveis~~

~~Segundo Sebesta (2018), uma variável é a abstração de uma parte da memória na qual está sendo utilizada. Essa abstração é feita por programadores para identificar os locais da memória através de nomes, tornando a codificação de um~~ *~~software~~* ~~muito mais legível.~~

~~As variáveis podem ser caracterizadas por meio de seis atributos, conforme listado por Sebesta (2018):~~

* ~~Nome: é a identificação da variável no programa. As nomeações das variáveis são estabelecidas por regras da própria linguagem de programação, como a proibição do uso de palavras reservadas, o fato da linguagem ser~~ *~~case sensitive~~* ~~(quando letras maiúsculas e minúsculas são consideradas caracteres distintos), entre outros;~~
* ~~Endereço: é o local da memória da máquina onde a variável está sendo armazenada;~~
* ~~Valor: é o conteúdo armazenado na memória e no qual está associado a variável;~~
* ~~Tipo: é o tipo de dado que está associado a variável. Durante a codificação do programa, algumas linguagens exigem que o tipo seja identificado junto com a variável (fortemente tipada) enquanto outras não obrigam o programador a declarar o tipo da variável (fracamente tipada)~~
* ~~Vinculação: é a associação entre a variável e ao seu tipo ou valor, podendo o tempo de vinculação entre eles variar de acordo com a linguagem de programação;~~
* ~~Escopo: Rossi (2024) define um escopo de variável como o contexto no qual a variável é definida. Com isso, o acesso a ela será impactado dependendo do local em que ela for definida, podendo ser declarada fora de uma função (escopo global) ou dentro de uma função (escopo local).~~

~~Além das variáveis, existe o conceito de constantes que, de acordo com Lopes (2023), desempenha a mesma função das variáveis, porém, ao contrário das mesmas, seu valor nunca é alterado durante a execução do programa.~~

~~2.3.2.5 Estruturas de Decisão~~

~~Souza~~ *~~et al.~~* ~~(2019) define as estruturas de decisão como uma estrutura da codificação que permite a tomada de decisão por meio de uma condição. Elas são classificadas de acordo com a palavra reservada utilizada, podendo ser:~~

* ~~SE-ENTÃO: essa estrutura utiliza apenas a palavra~~ *~~if~~* ~~em seu bloco de construção;~~
* ~~SE-ENTÃO-SENÃO: além da palavra~~ *~~if~~*~~, utiliza o~~ *~~else~~* ~~em seu bloco de construção. Essa estrutura é utilizada para indicar ao programa o que deve ser feito caso a primeira expressão lógica retornar o valor Falso;~~
* ~~CASO: estrutura utilizada para apresentar mais de duas expressões lógicas, possibilitando a utilização de mais testes de lógica sem sobrecarregar o programa. Utiliza as palavras reservadas~~ *~~switch~~* ~~e~~ *~~case~~* ~~em seu bloco de construção.~~

~~2.3.2.6 Estruturas de Repetição~~

~~Conforme Souza~~ *~~et al.~~* ~~(2019), uma estrutura de repetição permite a repetição de blocos de código de forma controlada. São classificadas nos seguintes tipos:~~

* ~~ENQUANTO-FAÇA: essa estrutura permite a repetição de comando enquanto a condição da estrutura for verdadeira. Utiliza a palavra reservada~~ *~~while~~*~~;~~
* ~~REPITA-ATÉ: essa estrutura executa os comandos repetidamente até a condição da estrutura retornar o valor Verdadeiro. A palavra reservada utiliza para a construção dessa estrutura são~~ *~~do~~* ~~e~~ *~~while~~*~~;~~
* ~~PARA-ATÉ-FAÇA: essa estrutura executa os comandos repetidamente até atingir um determinado valor parametrizado em uma espécie de contador. A palavra reservada~~ *~~for~~* ~~é utilizada para construir essa estrutura de repetição.~~

~~2.3.3 Paradigmas de Programação~~

~~Para possuir a capacidade de processar um programa, o computador executa o mesmo por uma linguagem que ele é capaz de entender. Essa linguagem, denominada linguagem de máquina ou linguagem de baixo nível, possui suas instruções construídas por meio de~~ *~~bits~~*~~, nos quais utilizam números binários em sua composição. Porém, esse tipo de linguagem é de difícil entendimento tanto para os programadores quanto para os usuários comuns.~~

~~Para tornar a escrita de~~ *~~softwares~~* ~~mais acessíveis, foram criadas as linguagens de alto nível, nas quais utilizam palavras comuns do vocabulário nas instruções dos programas. Com isso, o programador, após a escrita do código-fonte, deve transformar esse código em linguagem de máquina por meio de um processo chamada de compilação.~~

~~Com a evolução e criação de mais linguagens de alto nível, cada uma delas possui sintaxes e um conjunto de parâmetros bem definidos para que possam ser executadas e solucionar problemas a nível computacional. Com isso, cada linguagem utiliza um ou mais paradigmas de programação na abordagem de problemas e na elaboração de soluções.~~

~~Eldeweiss e Livi (2014) define um paradigma de programação como a forma que a solução de um problema está estruturada e será executada em um programa utilizando as técnicas e recursos disponíveis daquela linguagem. Os principais paradigmas utilizados pelas linguagens de programação são: imperativo, estruturado, funcional e orientado a objetos.~~

~~2.3.3.1 Programação Imperativa~~

~~Segundo Eldeweiss e Livi (2014), a programação imperativa consiste na construção dos comandos de um~~ *~~software~~* ~~em sequência, nos quais o computador os executará na ordem que forem definidos.~~

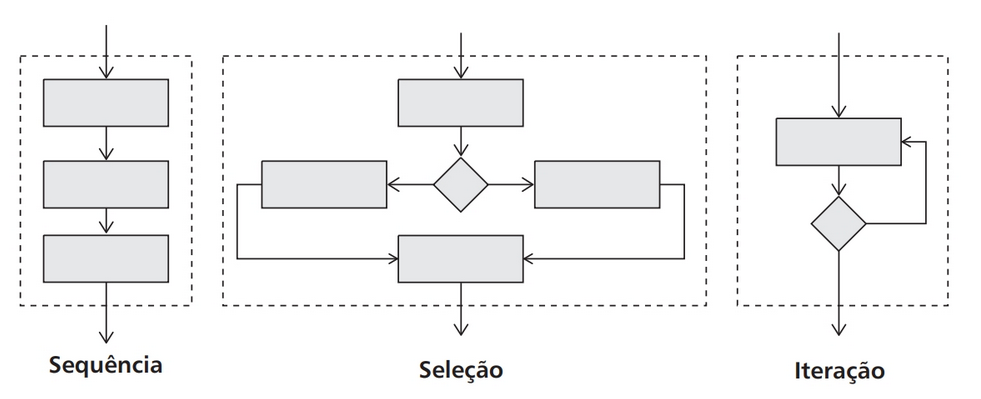
~~Esse tipo de paradigma foi uma das primeiras técnicas de programação utilizada no desenvolvimento de sistemas e se caracteriza por representar os problemas de forma intuitiva, seguindo uma lógica de ações pré-definidas. Por conta disso, conforme explicação de Eldeweiss e Livi (2014), possui uma grande importância até os dias atuais e são muito utilizados não apenas por sistemas legados, mas muitos sistemas modernos utilizam algoritmos escritos dessa forma como referência.~~

~~As linguagens de programação que utilizam esse paradigma são: Pascal, Fortran, Cobol, Ada, C, entre outras.~~

~~2.3.3.2 Programação Estruturada~~

~~A programação estruturada, conforme definida por Eldeweiss e Livi (2014), baseia-se que o fluxo de execução do programa deve possuir uma estrutura sintática evidente, e que a mesma deve estar presente em dois níveis: controle e unidade.~~

~~No nível de comando, o programa deve possuir apenas dois pontos: um de entrada e um de saída, e os “n” caminhos que fazem a ligação entre esses dois pontos já são definidos na construção do programa por meio de mecanismos de controles de fluxo, que podem ser de sequência, de seleção e de iteração. A figura 10 exemplifica o funcionamento desse controle.~~

~~Figura 10 – Mecanismos de controle de fluxo ~~

~~Fonte: Eldeweiss e Livi (2014).~~

~~No nível de unidade, conforme explicação de Eldeweiss e Livi (2014), consiste na utilização de unidades separadas de programas, chamadas de módulos. Essa ênfase tem origem em uma ideia proposta pelo cientista da computação Edsger Wybe Dijkstra em 1972.~~

~~As linguagens de programação que utilizam esse paradigma em sua estrutura são: ALGOL 58 e ALGOL 60.~~

~~2.3.3.3 Programação Funcional~~

~~Manzano (2020) define a programação funcional como uma atividade de ação lógica que tem como foco os resultados. Em outras palavras, esse paradigma possui uma preocupação maior no que deve ser feito, e não como ser feito. Já Wampler (2012) traz a definição de que a programação funcional na forma como os elementos da programação (como variáveis e valores) se relacionam com a matemática.~~

~~A programação funcional, de acordo com Wampler (2012), segue alguns princípios fundamentais, que são:~~

* ~~Estado imutável: os valores atribuídos a uma determinada variável não podem ser modificados;~~
* ~~Prioridades para funções: as estruturas de dados são separadas das funções, onde as mesmas possuem prioridade na ordem de execução;~~
* ~~Recursão: a programação funcional não utiliza laços de repetição, utilizando a técnica de recursividade para executar comandos repetidos.~~

~~As linguagens de programação que utilizam o paradigma funcional, conforme listado por Manzano (2020), são: Elm, Haskell, Miranda, Lambdascript, entre outras.~~

~~2.3.3.4 Programação Orientada a Objetos~~

~~Conforme definição por Dall’Oglio (2007), a programação orientada a objetos é representada por uma filosofia de construção de sistemas, que consiste na utilização de objetos para representar elementos do mundo real. Esses objetos são representados por classes, que possuem atributos (variáveis) e métodos.~~

~~O desenvolvimento de~~ *~~software~~* ~~utilizando o paradigma de orientação a objetos traz muitas vantagens para o produto desenvolvido. Franco (2014) lista algumas dessas vantagens:~~

* ~~Natural: permite que o programador defina os componentes do sistema com base em objetos e comportamentos do mundo real;~~
* ~~Confiável: os objetos isolam o comportamento e a responsabilidade do código no local de onde eles pertencem, permitindo a manutenção do sistema e trazendo mais confiabilidade;~~
* ~~Reutilizável: esse paradigma permite a reutilização de código a partir de conceitos como herança e polimorfismo, trazendo um ciclo de desenvolvimento mais curto e menos custoso.~~

~~Algumas linguagens de programação que tem esse paradigma como base são: Python, Java, C#, Dart, entre outras.~~

~~2.4 SEGURANÇA DA INFORMAÇÃO~~

~~Durante o processo de desenvolvimento de~~ *~~software,~~* ~~uma das características que se deve levar em consideração é a segurança do sistema. Muitos sistemas lidam com dados sensíveis, como o CPF e números do cartão de crédito, e, por conta disso, é dever dos desenvolvedores tornar o ambiente seguro, a fim de evitar vazamentos de dados e prejuízos tanto para a equipe de programação quanto para o usuário final.~~

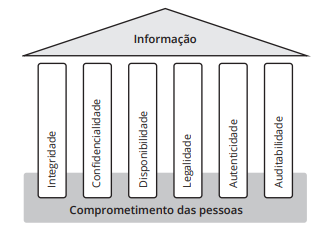
~~Para Fontes (2015), a segurança da informação tem como objetivo proteger a informação e seus recursos para que a empresa tenha capacidade de atingir seus propósitos institucionais. Com isso, a segurança da informação tem como base os processos organizacionais definidos pela gestão do projeto de desenvolvimento ou da organização.~~

~~Fontes (2015) separa o processo organizacional da segurança da informação nos seguintes pilares:~~

* ~~Confidencialidade: a informação deve ser acessada apenas por um usuário previamente autorizado e com objetivos relacionados a sua atividade profissional na organização;~~
* ~~Integridade: a informação deve manter seu estado original, não podendo haver manipulações que levem a adulterações ou que provoque a sua corrupção;~~
* ~~Disponibilidade: a informação deve estar disponível para as operações da organização;~~
* ~~Autenticidade: a veracidade das fontes da informação deve ser garantida;~~
* ~~Auditabilidade: a informação e as ações sobre elas devem estar registradas e as mesmas deve ter condições de serem auditadas;~~
* ~~Legalidade: o uso das informações deve estar em conformidade com a lei;~~

~~A figura 11 demonstra os pilares da segurança da informação.~~

~~Figura 11 – Pilares da segurança da informação~~

~~~~

~~Fonte: Fontes (2015)~~

~~2.4.1~~ *~~Hacking~~*

*~~Hacking~~* ~~é uma ação que consiste em explorar falhas e vulnerabilidades de um determinado sistema, obtendo acesso não autorizado a ele, conforme definição da Kaspersky (2024). Esse tipo de prática pode não ser criminosa, pois há profissionais da área de segurança cibernética que exercem essa atividade com o intuito de melhorar a segurança dos sistemas. Porém, o~~ *~~hacking~~* ~~ainda possui conotação negativa devido a sua ligação com atividades criminosas.~~

~~A prática criminosa do~~ *~~hacking~~* ~~causa prejuízos não apenas para o financeiro das empresas afetadas, mas também para sua imagem. Segundo uma pesquisa realizada pela empresa de segurança Proofpoint em 2022 e divulgada pelo site de notícias G1 em 2023, 25% das empresas brasileiras tiveram prejuízos relacionados com crimes cibernéticos, com a varejista Americanas sendo a mais afetada, tendo perdas em vendas de cerca de 1 bilhão de reais. Para evitar perdas, muitas empresas têm investido na segurança digital de seus sistemas e de sua infraestrutura tecnológica a fim de mitigar ataques cibernéticos, conforme um levantamento realizado pela Confederação Nacional das Seguradoras e divulgado pelo jornalista Thássius Veloso no portal de notícias da Globo em 2024.~~

~~Apesar das pesquisas e da Kaspersky (2024) relatarem que os métodos mais utilizados pelos cibercriminosos serem a Engenharia Social e o~~ *~~phishing,~~* ~~eles estão relacionados com a interação do~~ *~~hacker~~* ~~com o usuário final. Nesses casos, o sistema, por mais sofisticado e bem programado que seja, não é capaz de impedir uma ação errônea do usuário fora do âmbito de programação do sistema, sendo dever da organização orientar e treinar o usuário do sistema para uma utilização segura do~~ *~~software.~~*

~~Durante o processo de desenvolvimento do sistema, a equipe de desenvolvedores deve evitar vulnerabilidades que possibilitem um ataque~~ *~~hacker~~* ~~utilizando a programação do sistema. Os principais ataques~~ *~~hackers~~* ~~que exploram falhas de~~ *~~software~~* ~~são: SQL~~ *~~Injection~~*~~,~~ *~~Hijacking~~* ~~e ataques de negação de serviços.~~

~~2.4.2.1 SQL~~ *~~Injection~~*

~~Conforme definição de Fraga (2019), o SQL~~ *~~Injection~~* ~~é um tipo de ataque que se aproveita em falhas do~~ *~~software~~* ~~que interage com banco de dados por meio da linguagem SQL (~~*~~Structured Query Language~~*~~). Esse ataque ocorre quando o~~ *~~hacker~~* ~~insere instruções SQL nos campos de formulário do sistema. Dessa forma, ele irá concatenar essas instruções dentro de uma consulta ao banco de dados, provocando a execução do comando que foi inserido.~~

~~Essa falha do sistema, como afirma Ferreira (2021), geralmente ocorre por causa que as informações digitadas pelo usuário no formulário são concatenadas nos comandos SQL do banco de dados sem haver um tratamento dessas informações. A maioria das linguagens de programação, como o Java e o PHP, possui comandos que preparam as instruções SQL e inibe o ataque de SQL~~ *~~Injection~~* ~~por meio da classe~~ *~~PreparedStatement~~*~~. Dessa forma, o programador atribui o comando SQL por meio das funções dessa classe, sem precisar concatenar de maneira direta.~~

~~2.4.2.2~~ *~~Session~~**~~Hijacking~~*

~~Uma das características de aplicações~~ *~~web~~* ~~é a utilização de~~ *~~cookies~~* ~~de autenticação, de maneira que faça com que o servidor, através das requisições, identifique se o usuário está logado ou não no sistema e se o~~ *~~cookie~~* ~~utilizado na requisição é válido.~~

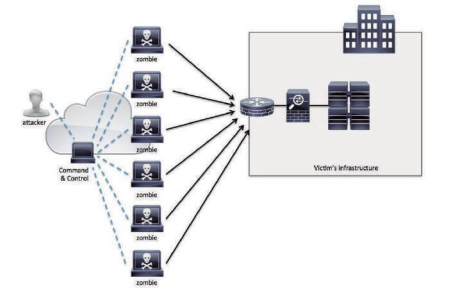
~~De acordo com Ferreira (2021),~~ *~~Session Hijacking~~* ~~é um tipo de ataque onde o atacante rouba o~~ *~~cookie~~* ~~de autenticação único e sequestra a sessão de usuário dele através de uma vulnerabilidade do sistema. Essa falha ocorre justamente no~~ *~~cookie~~* ~~de autenticação, pois o servidor não atrela ele ao dispositivo, mas ao usuário. Ou seja, se o usuário acessar o sistema em três dispositivos distintos, o servidor vai atrelar o mesmo identificador para cada dispositivo logado. Para roubar o identificador único do navegador do usuário, o~~ *~~hacker~~* ~~utiliza uma combinação de outros tipos de ataques, como o~~ *~~phising~~* ~~e ataques~~ *~~Cross Site Scripting.~~*

~~Para que o sistema fique seguro desse tipo de ataque, Ferreira (2021) afirma que é necessário a utilização do protocolo HTTPS nas aplicações que estarem disponíveis na~~ *~~web~~*~~. Além disso, o sistema deve possuir um certificado digital, garantindo uma segurança maior nas operações da aplicação. Outra maneira também seria restringir o acesso de scripts aos cookies por meio da programação do sistema.~~

~~2.4.2.3 Negação de Serviço~~

~~Os ataques de negação de serviço (do inglês~~ *~~Denial of Service~~*~~) são caracterizados como uma técnica de~~ *~~hacking~~* ~~que tem como objetivo tornar o acesso ao sistema~~ *~~web~~* ~~e a seus recursos indisponíveis para seus usuários, conforme definição de Fraga (2019).~~

~~Esse tipo de ataque não se trata de uma invasão ao sistema, mas da sua indisponibilidade por sobrecarga de requisições ao servidor onde o mesmo está hospedado, podendo ocorrer de uma forma comum (DoS) ou de maneira distribuída (DDoS), sendo esse último podendo utilizar~~ *~~bots~~* ~~como ferramenta para propagar o ataque. A figura 12 demonstra graficamente como ocorre um ataque DDoS.~~

~~Figura 12 – Representação de um ataque DDoS~~

~~Fonte: Fraga (2019)~~

~~De acordo com a IBM (2024), para mitigar esse tipo de ataque ao~~ *~~software~~*~~, é necessário adotar tecnologias que possam desviar e redistribuir o tráfego de rede, as quais podem ser:~~ *~~firewalls~~*~~, redes de servidores distribuídos e sistemas de gerenciamento de eventos.~~

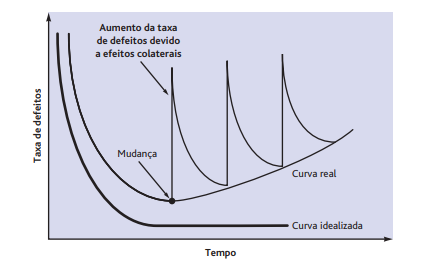
~~2.5 ENGENHARIA DE~~ *~~SOFTWARE~~*

~~A engenharia de~~ *~~software~~* ~~é um campo de estudo da engenharia que se preocupa com questões relacionadas ao~~ *~~software~~*~~, desde o seu planejamento inicial até a disponibilização dele para o usuário. Porém, antes de discorrer sobre essa área, é necessário entender o conceito de~~ *~~software~~*~~.~~

~~Pressman e Maxim (2016) definem um~~ *~~software~~* ~~como um conjunto de instruções que, ao serem executadas, desempenham funções e características desejadas. Além disso, afirma que o~~ *~~software~~* ~~possui um papel duplo, pois ele pode ser utilizado como um produto ou como uma plataforma para oferecer outros produtos.~~

~~Diferentemente do~~ *~~hardware~~*~~, o~~ *~~software~~* ~~possui como característica principal o fato dele ser próximo de um elemento lógico do que de um elemento físico, conforme afirmação de Pressman e Maxim (2016). Isso faz com que ele não seja suscetível a fatores maléficos do meio ambiente e, consequentemente, não sofrendo efeitos de desgaste, o que é característico do~~ *~~hardware~~*~~. Porém, apesar de não sofrer desgaste, o~~ *~~software~~* ~~sofre consequências de deterioração à medida que alterações sejam realizadas nele, conforme demonstrada na figura 13.~~

~~Figura 13 – Curva de defeito de~~ *~~software~~*

~~~~

~~Fonte: Pressman e Maxim (2016)~~

~~Além disso, quando um componente do~~ *~~hardware~~* ~~se desgasta, há a possibilidade de substituição do mesmo, algo que não se aplica ao~~ *~~software~~*~~. Nesse caso, quando há um defeito, o mesmo foi ocasionado por falhas no projeto ou no processo de execução das tarefas.~~

~~Para evitar essas falhas que podem causar defeitos no~~ *~~software~~* ~~justifica a importância da existência da engenharia de~~ *~~software~~*~~. Para Sommerville (2011), há dois motivos principais que tornam a engenharia de~~ *~~software~~* ~~importante:~~

* ~~Dependência da sociedade por~~ *~~softwares~~* ~~avançados;~~
* ~~Custo menor na utilização de abordagens sistemáticas na produção de um~~ *~~software~~*~~.~~

~~2.5.1 Processos de Desenvolvimento~~

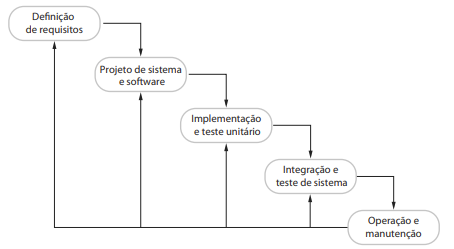
~~Conforme definição de Sommerville (2011), um processo de~~ *~~software~~* ~~é um conjunto de atividades que tem como resultado a produção de um produto de~~ *~~software~~*~~. Esse conjunto pode envolver diversas tarefas definidas pela equipe do projeto, porém deve-se envolvê-las em quatro atividades principais listadas por Sommerville (2011):~~

* ~~Especificação: consiste na definição das funcionalidades e das restrições do~~ *~~software~~*~~;~~
* ~~Implementação: consiste na etapa de produção do~~ *~~software~~*~~, conforme as especificações levantadas;~~
* ~~Validação: consiste na avaliação do~~ *~~software~~* ~~para que o mesmo esteja de acordo com as demandas do usuário;~~
* ~~Evolução: consiste na aplicação de mudanças e melhorias no~~ *~~software~~*~~.~~

~~Os processos podem ser representados por modelos, nos quais apresentam informações genéricas sobre as atividades. Sommerville (2011) divide os processos de desenvolvimento em três modelos principais:~~

* ~~Modelo em Cascata: esse modelo apresenta as fases do processo de desenvolvimento de forma separada, ou seja, ele considera as atividades como fases distintas, conforme mostrado na figura 14;~~

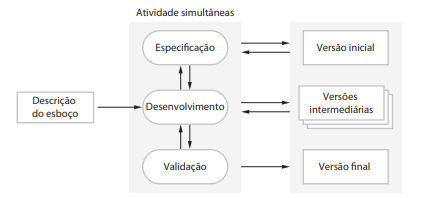
~~Figura 14 – Modelo em cascata~~

~~~~

~~Fonte: Sommerville (2011)~~

* ~~Desenvolvimento incremental: nesse modelo, o~~ *~~software~~* ~~é desenvolvido como uma série de versões, na qual cada versão adiciona uma funcionalidade ou melhoria que não estava presente na anterior, como demonstrado na figura 15;~~

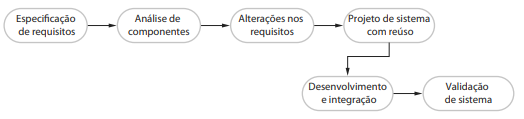
~~Figura 15 – Modelo incremental.~~

~~~~

~~Fonte: Sommerville (2011)~~

* ~~Engenharia orientada a reuso: esse modelo consiste na reutilização de componentes já existentes no sistema, se preocupando apenas na integração desses componentes com outras partes da aplicação em vez de planejar o desenvolvimento de um~~ *~~software~~* ~~do zero, conforme demonstrado na figura 16.~~

~~Figura 16 – Modelo de reutilização de componentes~~

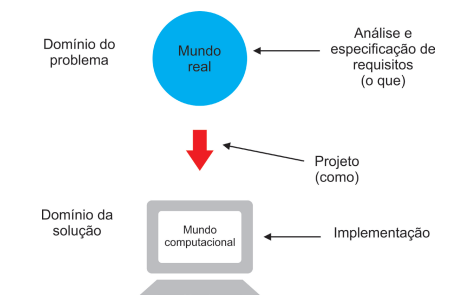
~~~~

~~Fonte: Sommerville (2011)~~

~~2.5.2 Projeto de Arquitetura~~

~~Para Ribeiro~~ *~~et al.~~* ~~(2015), o projeto do~~ *~~software~~* ~~é o passo inicial no processo de desenvolvimento do mesmo. É nesse momento que a equipe deve utilizar da abstração para converter os problemas do mundo real em uma solução computacional. Ao abstrair o problema, a equipe de desenvolvimento é capaz de elaborar uma solução para o mesmo por meio de ferramentas computacionais. A figura 17 demonstra um projeto de~~ *~~software~~*~~.~~

~~Figura 17 – Representação de um projeto de~~ *~~software~~*~~.~~

~~~~

~~Fonte: Ribeiro~~ *~~et al.~~* ~~(2015)~~

~~Para que a implementação do~~ *~~software~~* ~~seja feita de maneira correta, e que o mesmo seja capaz de trabalhar em seu pleno funcionamento, é necessário analisar a melhor arquitetura para atender a essa solução. Para Brooks (2018), a arquitetura do~~ *~~software~~* ~~é composta pela especificação completa e detalhada de seu funcionamento, de acordo com a pessoa que está utilizando.~~

~~Para Pressman e Maxim (2016), cada envolvido possui uma visão diferente do conceito de arquitetura. Por conta disso, para evitar que haja desalinhamento do objetivo do projeto por conta dessas diferenças, existem padrões para que a equipe possa seguir para desenvolver o~~ *~~software~~*~~. Os principais padrões de arquitetura utilizados nos processos de desenvolvimento de~~ *~~software~~*~~, conforme listado por Pressman e Maxim (2016),~~~~são: arquitetura em camadas, arquitetura de chamadas e retornos, arquitetura de fluxo de dados e arquitetura centralizada em dados.~~

~~2.5.2.1 Arquitetura em Camadas~~

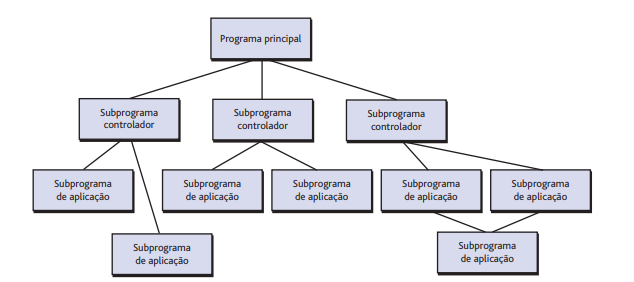
~~A arquitetura em camadas, também conhecida como arquitetura MVC (~~*~~Model-View-Controller~~*~~), é~~~~um conceito para desenvolvimento de~~ *~~software~~* ~~que separa a aplicação em três camadas, cada uma desempenhando uma função única. Segundo Bento (2021), essas camadas são:~~

* *~~Model~~*~~: essa camada é responsável pela lógica do~~ *~~software~~* ~~e da integração com o banco de dados ou com aplicações de terceiros através de API;~~
* *~~View~~*~~: camada responsável por interagir com o usuário final, exibindo as páginas ou interagindo com outros sistemas através de linguagens como XML ou JSON;~~
* *~~Controller~~*~~: essa camada é responsável pelo gerenciamento das requisições.~~

~~2.5.2.2 Arquitetura de Chamadas e Retornos~~

~~Esse padrão de arquitetura consiste em decompor as funções do sistema em uma estrutura hierárquica de controle, onde a função principal do programa invoca os componentes do mesmo, e esses componentes podem invocar outros componentes abaixo deles. Outra forma de aplicar essa arquitetura é por meio remoto, onde os componentes do sistema são distribuídos através de uma rede de computadores. A figura 18 demonstra a estrutura desse tipo de arquitetura.~~

~~Figura 18 – Arquitetura de chamada e retorno~~

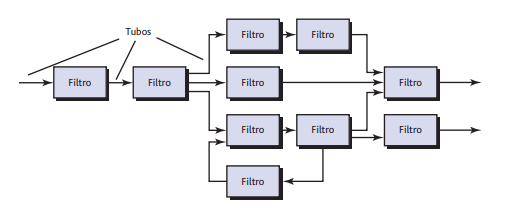
~~~~

~~Fonte: Pressman e Maxim (2016)~~

~~2.5.2.3 Arquitetura de Fluxo de Dados~~

~~Esse padrão de arquitetura se aplica quando um~~ *~~software~~* ~~precisa trabalhar com o tratamento de dados de entrada utilizando componentes computacionais ou manipulando os dados de saída. Essa manipulação ocorre através de uma técnica conhecida como tubos-e-filtros, onde ocorre a transmissão dos dados pela aplicação. A figura 19 demonstra visualmente a estrutura da arquitetura de fluxo de dados.~~

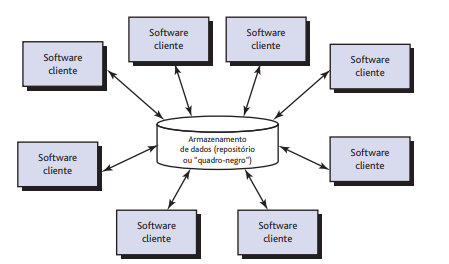
~~Figura 19 – Arquitetura de fluxo de dados~~

~~~~

~~Fonte: Pressman e Maxim (2016)~~

~~2.5.2.4 Arquitetura Centralizada em Dados~~

~~Essa arquitetura consiste na centralização de uma base de dados, onde ela poderá ser acessada pelos componentes do sistema, nos quais trazem alterações nos dados contidos nessa base. Esse tipo de arquitetura prioriza a característica da integrabilidade, ou seja, cada componente trabalha de forma independente, sem que a sua operação na base afete o outro. A figura 20 demonstra a estrutura da arquitetura centralizada em dados.~~

~~Figura 20 – Arquitetura centralizada em dados~~

~~Fonte: Pressman e Maxim (2016)~~

~~Esse padrão de arquitetura é frequentemente utilizado em~~ *~~softwares~~* ~~desenvolvidos para a~~ *~~web~~*~~, tendo em vista que esse tipo de aplicação depende da comunicação entre a máquina do cliente e o servidor. Também é utilizado em~~ *~~softwares~~* ~~voltados para versionamento de código, muito utilizado por equipes de programação durante o processo de desenvolvimento de uma aplicação para gerenciar as alterações realizadas na aplicação pelos colaboradores.~~

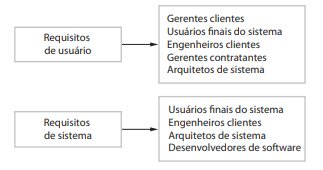
~~2.5.3 Engenharia de Requisitos~~

~~Os requisitos de um sistema são especificações de seu funcionamento, bem como do que ele deve fazer e quais são as suas restrições. A etapa de levantamento dos requisitos é de extrema importância no processo de produção do~~ *~~software,~~* ~~pois é a partir dela que a equipe entende as necessidades do cliente para com o sistema e, dessa forma, entregar um produto que tenha como finalidade oferecer alguma solução no âmbito computacional. Essa etapa de levantamento é chamada de engenharia de requisitos (do inglês,~~ *~~requirements engineering~~*~~).~~

~~Sommerville (2011) afirma que esse termo, no contexto da indústria de~~ *~~software~~*~~, é utilizado como abstração do que o sistema deve fazer em vez da descrição formal e planejada de seu funcionamento. Por isso, os requisitos são separados em dois níveis de descrição:~~

* ~~Requisitos de usuário: conforme definido por Sommerville (2011), os requisitos de usuário são compostos por descrições dos serviços que o~~ *~~software~~* ~~deve oferecer a seus usuários;~~
* ~~Requisitos de sistema: de acordo com Sommerville (2011), requisitos de sistema são declarações mais detalhadas das funções e restrições operacionais do~~ *~~software~~*~~, definindo o que deve ser implementado. A figura 21 mostra os tipos de leitores de um requisito de~~ *~~software~~*~~.~~

~~Figura 21 – Tipos de leitores de requisitos de~~ *~~software~~*

~~~~

~~Fonte: Sommerville (2011).~~

~~Outra classificação dos requisitos de~~ *~~softwares~~* ~~frequentemente utilizada, e citada por Sommerville (2011), é composta pelos seguintes tipos:~~

* ~~Requisitos funcionais: são declarações do comportamento do sistema perante a determinada situação ou entradas específicas, listando as obrigações (o que o~~ *~~software~~* ~~deve fazer) e as proibições (o que o~~ *~~software~~* ~~não dever fazer);~~
* ~~Requisitos não funcionais: são funções ou restrições oferecidas pelo sistema. Não estão diretamente relacionados com os serviços aos usuários, mas a aspectos técnicos que afetam o funcionamento do~~ *~~software~~*~~, como segurança, desempenho, usabilidade, entre outros.~~

~~2.5.4 Modelagem de Sistemas~~

~~A modelagem de sistemas é a etapa de desenvolvimento de modelos que representam diferentes visões do sistema.~~

~~Os modelos são usados durante o processo de engenharia de requisitos para ajudar a extrair os requisitos do sistema; durante o processo de projeto, são usados para descrever o sistema para os engenheiros que o implementam; e, após isso, são usados para documentar a estrutura e a operação do sistema (SOMMERVILLE, 2011).~~

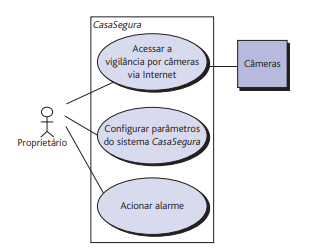
~~Geralmente, esses modelos são feitos utilizando a linguagem UML (~~*~~Unified Modeling Language~~*~~), porém outras ferramentas podem ser utilizadas na construção desses modelos.~~

~~2.5.4.1~~ *~~Unified Modeling Language~~*

~~A linguagem UML é uma linguagem utilizada para a construção de modelos visuais do funcionamento de um~~ *~~software~~*~~, sendo parte integrante no processo de desenvolvimento. Embora não seja a única ferramenta existente na construção desses modelos, ela se tornou padrão quando se trata de modelagem de sistemas.~~

~~Para construir os modelos, são utilizados diagramas que podem conter figuras ou formas geométricas. A UML possui cinco tipos de diagramas, conforme listado por Sommerville (2011), sendo eles:~~

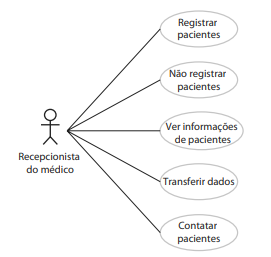
* ~~Diagramas de atividades: esse tipo de diagrama exibe as atividades envolvidas em um processo;~~
* ~~Diagramas de caso de uso: são utilizados para demonstrar uma visão completa do funcionamento do sistema a partir das interações do usuário. A figura 22 mostra um exemplo desse tipo de diagrama;~~

~~Figura 22 – Modelo de diagrama de caso de uso ~~

~~Fonte: Pressman e Maxim (2016)~~

* ~~Diagrama de sequência: esses diagramas são utilizados para modelar a sequência de ações e interações do usuário com o sistema. A figura 23 mostra um exemplo desse tipo de diagrama;~~

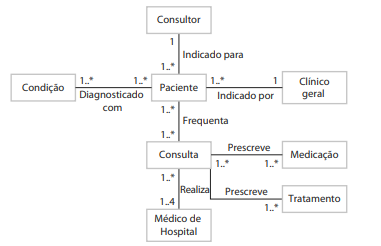
~~Figura 23 – Modelo de diagrama de sequência~~

~~~~

~~Fonte: Sommerville (2011)~~

* ~~Diagrama de classe: são utilizados no desenvolvimento de projetos onde há presença do paradigma de orientação a objetos, auxiliando na visualização das classes e objetos do sistema e no relacionamento entre eles. A figura 24 mostra um exemplo desse tipo de diagrama;~~

~~Figura 24 – Modelo de diagrama de classe~~

~~~~

~~Fonte: Sommerville (2011)~~

* ~~Diagrama de estado: esse diagrama exibe o comportamento do sistema perante a determinados eventos.~~

~~Em um processo de desenvolvimento de~~ *~~software,~~* ~~há muitas outras etapas a serem seguidas para que o sistema esteja de acordo com os requisitos levantados e que o mesmo seja capaz de solucionar problemas reais utilizando ferramentas computacionais, como realização de testes, análise de qualidade, manutenção do~~ *~~software~~*~~, entre outras. Todas essas etapas, quando aplicadas de maneira correta e de forma conjunta e sequencial, são capazes de resultar em um produto de~~ *~~software~~* ~~que atenda ao cliente, e ao mesmo tempo, traz boa reputação e sensação de segurança para a equipe de programação.~~

Nickolas, o que deve colocar aqui é uma seção de cada uma das palavras-chaves. De outra forma de contribuir: se vc for no buscador da net e colocar suas palavras-chaves encontrará trabalhos próximos do seu. Não isso aqui. Entende?

Por isso, acabei rabiscando duas das palavras-chaves. Sistemas gerenciamento de clientes sim, soluções web (?) e o que mais colocaríamos?

Vc está propondo um sistema de controle de clientes, mas não apresentou nada sobre controle de clientes.

**3 DESENVOLVIMENTO DO SISTEMA**

Dados os fundamentos teóricos necessários para o entendimento do funcionamento das ferramentas tecnológicas e de práticas de desenvolvimento de *software* aplicadas no projeto, nesse capítulo é apresentado o processo de desenvolvimento do sistema, desde as definições iniciais do projeto até a execução das etapas de desenvolvimento.

3.1 VISÃO GERAL DO PROJETO

3.1.1 Escopo do Projeto

O projeto tem como objetivo a construção de um *software web de* cadastro de atendimentos aos clientes de uma determinada empresa. Esse desenvolvimento se fez necessário devido ao fato das empresas, especialmente as de pequeno porte, necessitarem de ferramentas tecnológicas que não prejudiquem o orçamento delas e para organizarem os atendimentos e, dessa forma, construir um bom relacionamento com seus clientes. Além disso, com o desenvolvimento desse sistema, espera-se contribuir com a comunidade de desenvolvimento de *softwares open-source* na elaboração de soluções computacionais para a sociedade.

Esse projeto possui como público-alvo os seguintes *stakeholders*:

* Microempresas;
* Empresas de pequeno porte que priorizam o atendimento presencial em seu modelo de negócio;

O sistema estará disponível para utilização através da *web*, sendo necessário um navegador, como o Google Chrome ou Microsoft Edge, para acessá-lo. Nesse sistema, o usuário, representado pelo funcionário da empresa, será capaz de cadastrar os vendedores que realizam as vendas dos produtos ou dos serviços da organização, os clientes que adquirem esses produtos ou serviços e os usuários dos funcionários da empresa que irão utilizar o sistema. Além disso, o usuário será capaz de consultar os atendimentos que foram cadastrados, bem como os vendedores e os clientes. Todas as ações dos usuários serão realizadas por meio de uma interface intuitiva.

3.1.2 Planejamento

Inicialmente, foi necessário estabelecer uma identificação para o *software*, no qual recebeu “Sistema de Gerenciamento de Clientes” como seu nome oficial.

Após esse passo inicial, o desenvolvimento do sistema foi separado em três principais etapas, que são:

* Levantamento das necessidades; requisitos – não faltou isso?
* Configuração do banco de dados;
* Desenvolvimento da lógica do sistema e da conexão com o banco de dados (*backend*);
* Desenvolvimento da interface de usuário (*frontend*);

Definida as etapas de desenvolvimento, o próximo estágio de planejamento compreendeu-se em fazer o levantamento dos requisitos do sistema e a definição das ferramentas computacionais que seriam utilizadas no processo de construção do sistema. Depois de definidos, foram criadas tarefas para cada etapa do desenvolvimento.

Materiais e métodos

3.1.2.1 *Kanban*

~~O~~ *~~Kanban~~* ~~é uma ferramenta utilizada no gerenciamento de projetos que tem como origem o~~ *~~Just-in-Time~~* ~~(JIT). De acordo com Paoleschi (2019), o JIT é uma filosofia de administração surgida no Japão na metade do século XX nas fábricas da Toyota Motor Company para gerenciar e controlar o estoque, visando na redução de desperdícios e, consequentemente, dos custos. Para isso, essa filosofia utiliza diversas ferramentas, e uma delas é o~~ *~~kanban~~*~~.~~

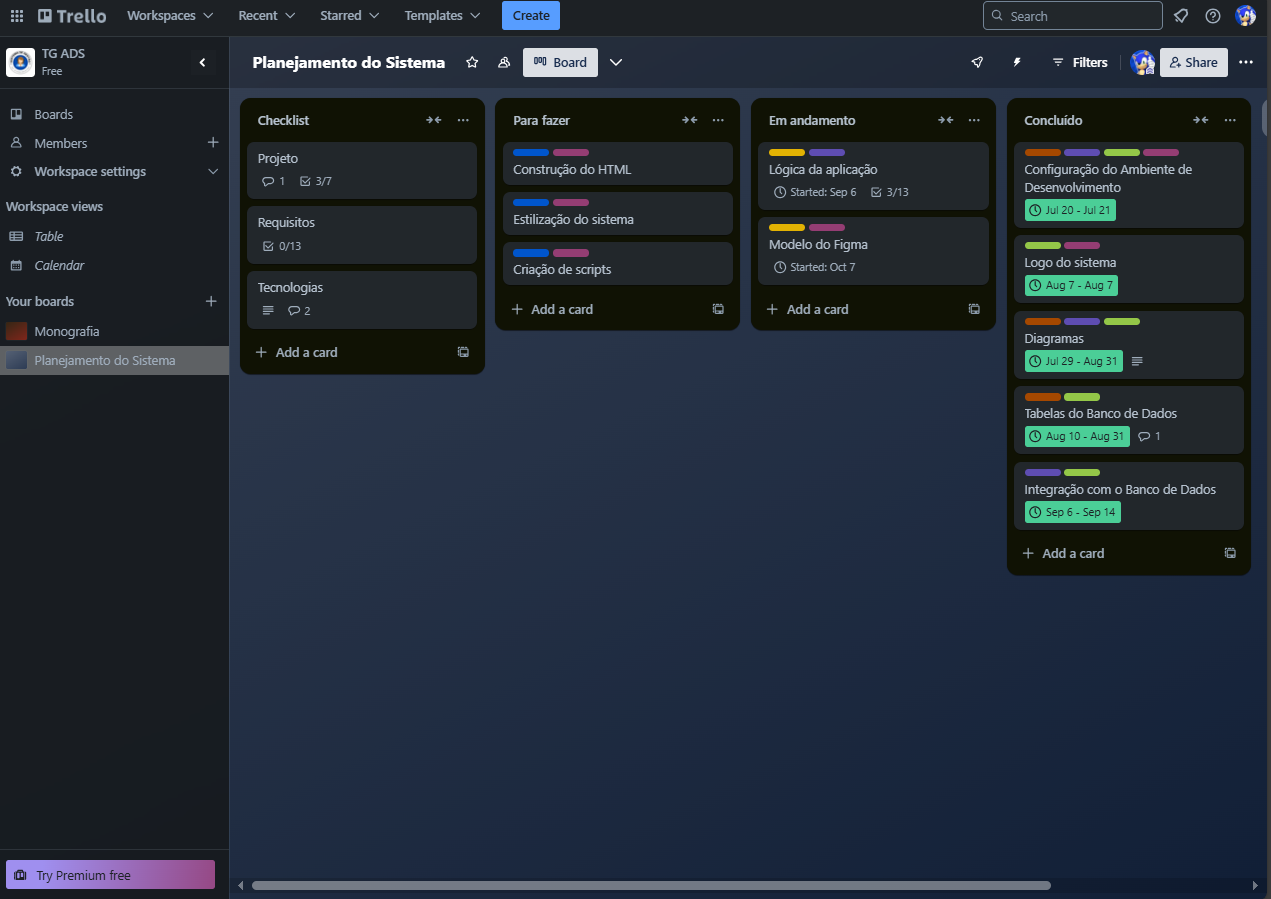
~~O~~ *~~kanban~~* ~~consiste em um quadro de controle de atividades, onde se pode ter uma visão geral do andamento do projeto. Cada atividade é representada por um cartão, que está localizado em uma etapa do projeto (geralmente representada por uma coluna). Quando a etapa daquele determinado cartão é finalizada, o mesmo é transferido para a próxima, seguindo um fluxo de planejamento até a finalização do projeto.~~

Atualmente, existem diversos *softwares* que representam digitalmente esse quadro de atividades. Para o desenvolvimento do sistema, foi escolhido o Trello para a organização das atividades do projeto.

~~O Trello é um~~ *~~software~~* ~~de gestão de projetos inicialmente desenvolvido pela Fog Creek Software e posteriormente adquirido pela Atlassian. De acordo com o guia elaborado pela Atlassian (2024), esse~~ *~~software~~* ~~pode ser adaptado a qualquer tipo de projeto, além de sua capacidade de ser utilizado em qualquer área de atuação, desde setores de produção até para projetos voltados para tecnologia da informação. Ele também é capaz de simplificar os projetos de uma forma intuitiva e lidar com projetos mais robustos e complexos sem prejudicar a organização dos mesmos e das equipes participantes. O Trello possui um modelo de serviço~~ *~~freemium~~*~~, onde a utilização da ferramenta é gratuita, porém o acesso a recursos mais avançados é pago.~~

Para montar um projeto no Trello, é necessário a criação de uma área de trabalho. Nessa área de trabalho, o usuário pode criar quadros de atividades, onde estarão as listas, que representam as etapas do projeto, e, dentro das listas, estão os cartões, que representam as atividades do projeto. O usuário tem a capacidade de criar um quadro por ele mesmo ou pode utilizar um modelo que o *software* disponibiliza.

No caso do projeto de desenvolvimento do Sistema de Gerenciamento de Clientes, foi criado uma área de trabalho específica para esse projeto. Nessa área de trabalho, foram criados dois quadros: um voltado para o gerenciamento da escrita da monografia e o outro específico para a gestão das atividades do projeto. A figura 25 permite visualizar o *kanban* do projeto feito no Trello.

Figura 25 – *Kanban* do projeto

Fonte: Desenvolvimento próprio (2024)

O quadro de atividades do desenvolvimento do sistema é organizado da seguinte forma:

* *Checklist*: essa lista é composta pelas informações gerais do projeto, que apresentam os passos que devem ser seguidos, os requisitos que devem ser cumpridos e as tecnologias utilizadas no projeto;
* Para fazer: essa lista é composta por atividades que não foram iniciadas no projeto;
* Em andamento: essa lista é composta por atividades que estão em desenvolvimento no momento atual;
* Concluído: lista que armazena as atividades que já foram finalizadas no projeto.

Nos cartões, são inseridas etiquetas para identificação da atividade, nas quais são divididas em dois tipos:

* Identificação do estágio: essa etiqueta tem como finalidade a identificação do estágio em que a atividade se encontra, representado pelas seguintes cores: azul (para fazer), amarelo (em andamento) e verde (concluído);
* Identificação da etapa: essa etiqueta tem como finalidade a identificação da etapa que a atividade pertence, representada pelas seguintes cores: laranja (banco de dados), roxo (*backend*) e rosa (*frontend*).

Cada cartão pode ter apenas uma etiqueta que identifica o estágio daquela atividade, mas para identificar a etapa, dependendo da atividade, ele pode ter mais de uma etiqueta de identificação.

Além disso, o quadro possui uma automação, funcionalidade disponibilizada pelo Trello, das datas de início e término da atividade e da mudança da etiqueta de identificação do estágio de desenvolvimento. Apenas as etiquetas de identificação da etapa são necessárias a manipulação manual.

Não é a seção para ficar “ensinando” Para todas as ferramentas, dizer como foi utilizada, com informações efetivas do seu projeto.

3.1.3 Requisitos Funcionais

Para estabelecer as funções que o sistema deve executar quando o usuário interage com ele, bem como suas restrições, foram levantados os seguintes requisitos, demonstrados na tabela 5:

Tabela 5 – Requisitos funcionais do sistema

|  |  |
| --- | --- |
| **Requisito** | **Descrição** |
| Cadastro de clientes | O sistema deve permitir que o usuário cadastro cadastre clientes |
| Outros ???? cadastros (quebrar) | O sistema deve permitir que o usuário cadastre atendimentos, vendedores e outros usuários |
| Consultas no sistema (idem ao anterior) | O sistema deve permitir que o usuário consulte os clientes, vendedores e atendimentos cadastrados anteriormente |
| Visualização dos usuários | O sistema deve permitir que o usuário visualize os usuários que foram cadastrados. Usuários e usuário: estas palavras confundem, quem é quem? Critérios da visualização deve, desejável, ser dita em sua particularidade, como, benefícios, valor ganho para o utilizador do sistema. |
| Cadastro existente | O sistema não deve permitir cadastrar um cliente ou vendedor que já tem cadastro Leia de novo! Requisito manutenção de dados de clientes |
| Exclusão de atendimento | O sistema não deve permitir que um atendimento seja excluído |
| Usuário inativo | O sistema não deve permitir que um usuário com *status* inativo acesse o sistema. O que é considerado inativo? Condição/restrição para RnF-Req, não-funcional segurança? A desativação é registrada em algum log, sugestão. |

Fonte: Desenvolvimento próprio (2024)

Revendo novamente – resumo em um requisito: Manutenção de dados de clientes

3.1.4 Requisitos Não Funcionais

Em relação aos aspectos técnicos do sistema, foram estabelecidos os seguintes requisitos, nos quais estão demonstrados na tabela 6:

Tabela 6 – Requisitos não funcionais do sistema

|  |  |
| --- | --- |
| **Requisito** | **Descrição** |
| Segurança | Acesso aos dados do sistema por meio de autenticação por *login* e senha: isso e Req Func. Enqto estratégia de segurança é a restrição do login é controle de acesso e autorização. |
| Usabilidade | Interface simples e intuitiva para o usuário |
| Compatibilidade | O sistema deve ser compatível com versões atuais dos navegadores *web* |

Fonte: Desenvolvimento próprio (2024)

Uma dica que colocamos: revise no produto final o que foi dito aqui de requisitos, mostre e comente lá em resultados o alcance disso.

3.2 AMBIENTE DE DESENVOLVIMENTO

Definida a organização do projeto, são necessárias a criação e a configuração do ambiente de desenvolvimento do sistema. Como o sistema é uma aplicação *web* e sua estrutura não é complexa ao ponto de consumir grandes volumes de recursos de *hardware* e do sistema operacional, não é necessário um computador que disponha de *hardware* com capacidade de grande processamento e de gerenciamento de memória.

3.2.1 Banco de Dados

~~Para o armazenamento dos dados do sistema, é necessário a construção de um banco de dados. Um banco de dados, segundo Date (2003), é uma coleção de dados persistentes usados pelos sistemas de uma determinada organização. O termo “persistente”, segundo o autor, refere-se ao fato de que, uma vez que o dado é inserido no sistema, ele só pode ser removido por alguma requisição explícita do sistema, podendo ser uma instrução SQL ou qualquer outra funcionalidade.~~

~~Para acessar esses dados, conforme explicado por Date (2003), é necessário a utilização de um~~ *~~software~~* ~~denominado Sistema de Gerenciamento de Banco de Dados (SGBD). No mercado, existem diversos sistemas que gerenciam o banco de dados e que permitem o acesso aos dados.~~

Para o projeto, foi escolhido o MySQL, um SGBD criado com a licença de *software* livre e gratuito para utilização para fins educacionais (para uso comercial, é necessário adquirir uma licença). Atualmente, o *software* pertence a Oracle.

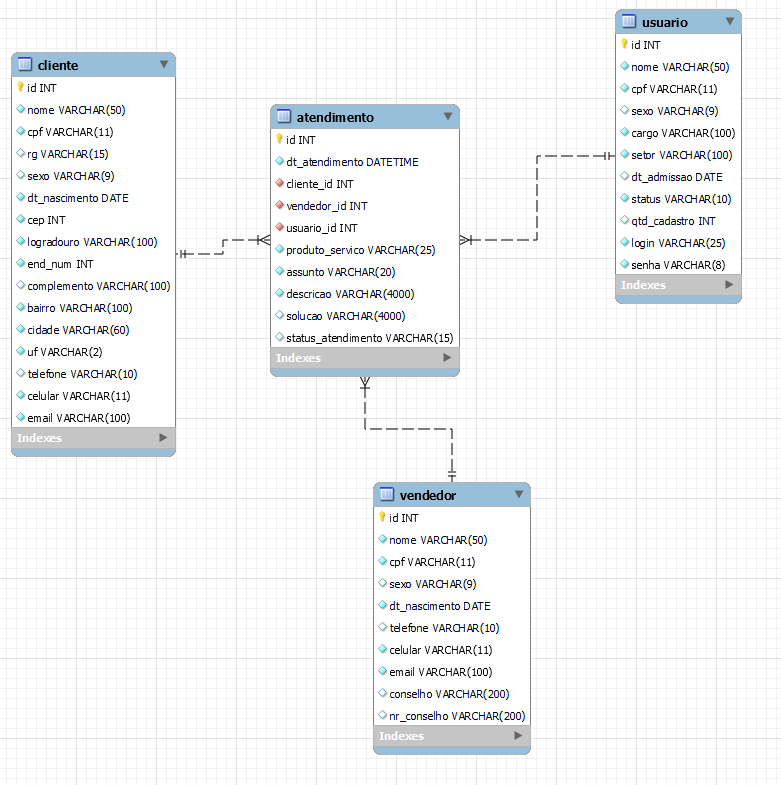
De acordo com Carvalho (2015), o MySQL possui características que estão alinhadas com os objetivos do projeto, entre elas:

* O *software* possui uma interface simples, facilitando o uso pelo desenvolvedor;
* Apesar da simplicidade da interface, o *software* é robusto, possuindo várias funcionalidades encontradas nos sistemas de gerenciamento de banco de dados;
* Possui alta estabilidade e desempenho, livrando a preocupação do desenvolvedor da indisponibilidade de sua aplicação por causa de instabilidades no banco de dados ou da demora no processamento;
* Compatibilidade com diversas linguagens de programação, permitindo a construção de diversos tipos de aplicação.

O banco de dados do sistema é composto por quatro tabelas, que são: atendimento, cliente, vendedor e usuário. Cada registro feito na tabela é representado por um número de identificação, que é gerado automaticamente assim que o registro é gravado no banco de dados. Esse número de identificação é a chave primária da tabela, na qual tem como função a criação dos relacionamentos com as outras tabelas.

A tabela de atendimento, sendo a principal, possui um relacionamento de um para muitos (representada pela notação 1:N) com as outras tabelas. Para conseguir referenciar as outras tabelas e, consequentemente, criar os relacionamentos, essa tabela possui três chaves estrangeiras, que são as chaves primárias das outras tabelas. Esse relacionamento é representado pelo Modelo Entidade Relacionamento (MER) da figura 26:

Figura 26 – Modelo Entidade Relacionamento do sistema



Fonte: Desenvolvimento próprio (2024)

O Modelo Entidade Relacionamento do banco de dados foi construído utilizando uma funcionalidade do MySQL de engenharia reversa, onde o SGBD converte as tabelas do banco de dados em um Modelo Entidade Relacionamento.

3.2.2 *Integrated Development Environment*

Para que o sistema seja desenvolvido, é necessário a utilização de um *software* chamado IDE (Ambiente de Desenvolvimento Integrado, do inglês *Integrated Development Environment*). Conforme a organização Red Hat (2023), uma IDE é um *software* que permite criar aplicações que reúne as ferramentas necessárias para o desenvolvimento de um sistema em um único ambiente, juntamente com uma interface gráfica.

Para o projeto, foi escolhida a IDE desenvolvida pela Microsoft, o Visual Studio Code. Embora muitos desenvolvedores o considerem como uma IDE, a empresa desenvolvedora do *software* o divulga como um editor de código-fonte. Apesar disso, o Visual Studio Code possui diversas funcionalidades que o compara a uma IDE, e que trazem facilidades no desenvolvimento do sistema. Entre elas são:

* Custo: a IDE é *open-source* e sua utilização é gratuita;
* Configuração da IDE: tanto a instalação quanto a configuração da IDE para o desenvolvimento são simples de serem realizados;
* Ferramentas de depuração: o Visual Studio Code possui ferramentas de depuração de código, o que facilita na busca de erros e *bugs* no código;
* Ferramentas de controle de alterações: o Visual Studio Code possui ferramentas integradas que auxiliam no gerenciamento de alterações no código;
* Extensões: uma das principais funcionalidades da IDE, o Visual Studio Code permite a instalação de extensões no *software*, nas quais podem possuir diversas finalidades, desde a configuração para desenvolver em uma determinada linguagem de programação, até a mudança de cores no tema de fundo do *software*.

3.2.3 Versionamento de Código

O versionamento de código é uma técnica que utiliza sistemas de controle de versão em sua aplicação, e consiste em gerenciar as alterações realizadas no código, auxiliando no desenvolvimento de um sistema. Segundo Aquiles e Ferreira (2014), esses sistemas utilizam um repositório, onde os membros do projeto podem acessar todas as versões do mesmo.

Existem diversos sistemas de controle de versão no mercado e, para esse projeto, foi selecionado o Git, um dos sistemas de controle de versão mais utilizados no mercado, e desenvolvido por Linus Torvalds, criador do Linux, conforme apresentado por Aquiles e Ferreira (2014).

Para gerenciar as versões do código do projeto, foi necessário a criação de um repositório. Para o Git, há dois tipos:

* Repositório local: os arquivos do projeto são armazenados em repositório localizado na máquina do desenvolvedor;
* Repositório remoto: é um repositório disponível na rede, podendo estar em uma rede interna da organização ou na *web*.

O modo mais comum de armazenamento de repositórios remotos é através do Github, uma aplicação *web* que hospeda repositórios Git. Além disso, ele também serve como referência social e profissional para desenvolvedores. Porém, no contexto do projeto, ele é utilizado para armazenar os arquivos do projeto de forma pública, dessa maneira, outros desenvolvedores podem contribuir com melhorias ou utilizar essa solução em outras aplicações.

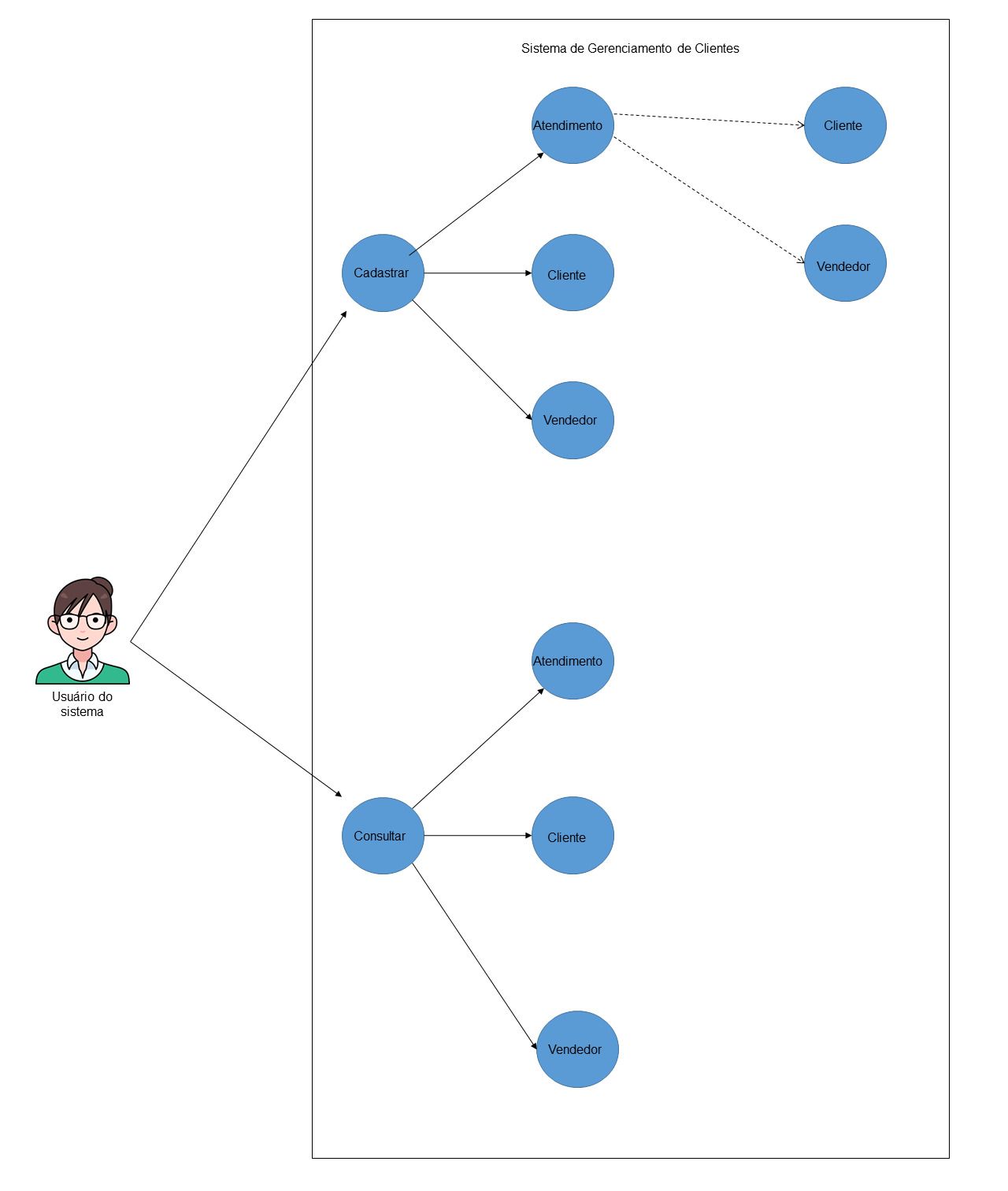
3.3 LÓGICA DO SISTEMA

Após a configuração do ambiente de desenvolvimento e a construção do banco de dados, foi iniciado o desenvolvimento da lógica de funcionamento do sistema, o que Taylor (2023) denomina de *backend* do sistema.

O *backend* foi construído utilizando a linguagem de programação PHP, uma tecnologia *open-source* que permite o pré-processamento de páginas HTML. A escolha da linguagem para a construção do sistema está relacionada com a compatibilidade das necessidades da aplicação com as vantagens listadas por Bento (2021), que são:

* Possui licença gratuita e seu código é aberto à comunidade;
* Ligação entre a interface visual e a lógica do sistema é feita de forma simples, sem a necessidade de bibliotecas ou *frameworks* de terceiros;
* Fácil integração com o banco de dados utilizado no projeto, que no caso é o MySQL;
* Bastante oferta de serviços de hospedagem de páginas PHP, bem como seu baixo custo;
* A versão 8.3, utilizada no projeto, possui servidor integrado, o que facilita o processo de desenvolvimento e absolve o desenvolvedor da dependência de *softwares* de servidores *web*.

Quanto ao desenvolvimento da lógica, o sistema permite que o usuário cadastre e consulte atendimentos, clientes e vendedores. Porém, para que ele possa cadastrar o atendimento, é necessário que exista um cliente e um vendedor já cadastrado. Através do diagrama de caso de uso representado na figura 27, é possível visualizar como funciona a lógica do sistema.

Figura 27 – Diagrama de Caso de Uso do sistema

Fonte: Desenvolvimento próprio (2024)

Sua proposta – sistema de gerenciamento – a palavra gerenciamento não lhe traz mais atribuições funcionais que consulta e cadastrar apenas

3.4 INTERFACE DE USUÁRIO

Finalizado o desenvolvimento do *backend* do sistema, iniciou-se o processo para desenvolver a interface visual do *software*, chamada de *frontend* por Taylor (2023). Tudo que está relacionado com o que o usuário pode visualizar e interagir no sistema foi elaborado nessa última etapa.

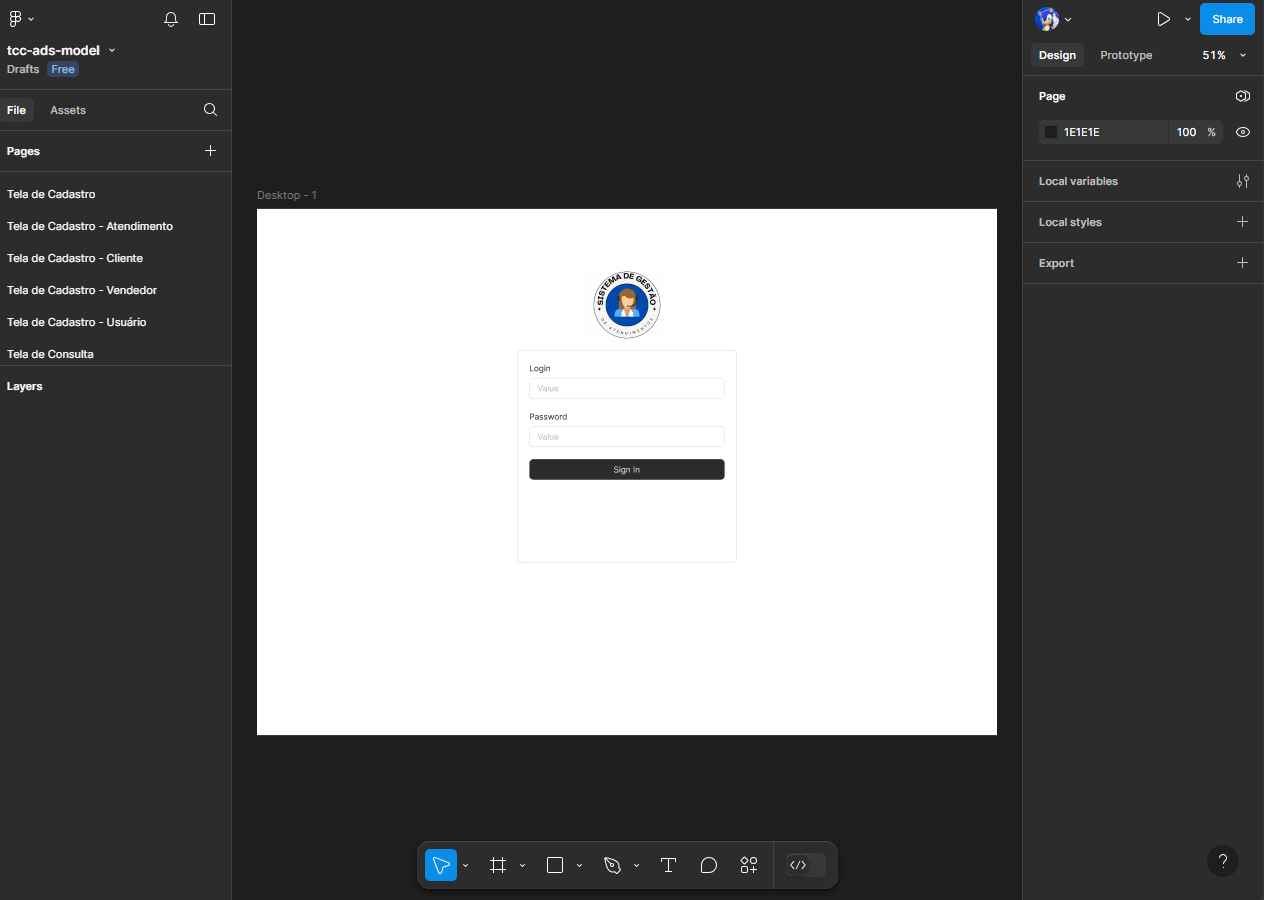
Inicialmente, foi elaborada uma logo para o sistema. Essa logo tem como objetivo principal servir de ícone para a página HTML, mas também serve como identificação visual do sistema, como uma espécie de marca. A figura 28 demonstra o resultado final da elaboração da logo do Sistema de Gerenciamento de Clientes.

Figura 28 – Logo do sistema



Fonte: Desenvolvimento próprio (2024)

Após a construção da logo, o próximo passo para a montagem da interface do usuário foi a elaboração de um protótipo visual. Esse protótipo tem como objetivo a aplicação da ideia para a interface em um modelo, antes de implementá-la de forma oficial no projeto. Para a construção desse protótipo, foi utilizado o Figma, ferramenta de design e desenvolvimento de modelos visuais, conforme definição de Staiano (2023). A escolha do Figma para desenvolver o protótipo se deve pela sua popularidade e pela sua acessibilidade. A figura 29 permite visualizar o processo de desenvolvimento do protótipo da interface visual do projeto pelo Figma.

Figura 29 – Modelagem da interface no Figma

Fonte: Desenvolvimento próprio (2024)

Para o desenvolvimento da interface de usuário, foram utilizadas três tecnologias que, geralmente, são usadas em conjunto por grande parte dos projetos de sistemas *web*: HTML, CSS e Javascript.

HTML, sigla de Linguagem de Marcação de Hipertexto (do inglês, *HyperText Markup Language*), é uma linguagem de marcação utilizada para converter documentos de textos em páginas *web* e aplicações, segundo Robbins (2009). Sua utilização no projeto se deve ao fato da linguagem PHP ter a capacidade de pré-processar as páginas escritas em HTML, o que traz compatibilidade para a tecnologia, facilitando o desenvolvimento do sistema.

CSS é uma linguagem de estilo usada para definir a apresentação de documentos escritos em linguagem de marcação, conforme definição de Junior e Patrício (2015). Uma de suas principais características é sua capacidade de prover a separação entre o formato e o conteúdo de um documento, ajudando na confecção de documentos mais bem estruturados e fáceis de manter (JUNIOR; PATRICIO, 2015). O CSS foi utilizado no projeto para estilizar e permitir que o sistema fique visualmente mais bonito. Além disso, sua integração com o HTML permite que a compatibilidade entre essas tecnologias seja feita de forma natural, o que traz facilidade no processo de desenvolvimento.

Javascript é uma linguagem de programação que foi desenvolvida para executar *scripts* na *web*, trabalhando em conjunto com o HTML e o CSS, segundo Flanagan (2013). A linguagem é comumente utilizada para executar comandos no lado do cliente, sem a necessidade de execução de *scripts* no servidor, tornando o processamento da página mais rápida. Por esse motivo, a linguagem foi escolhida para executar *scripts* simples e que não precisem de integração com o banco de dados ou de serem processados no servidor, deixando o PHP responsável apenas pela lógica do sistema e comunicação com o servidor.

**4 RESULTADOS OBTIDOS**

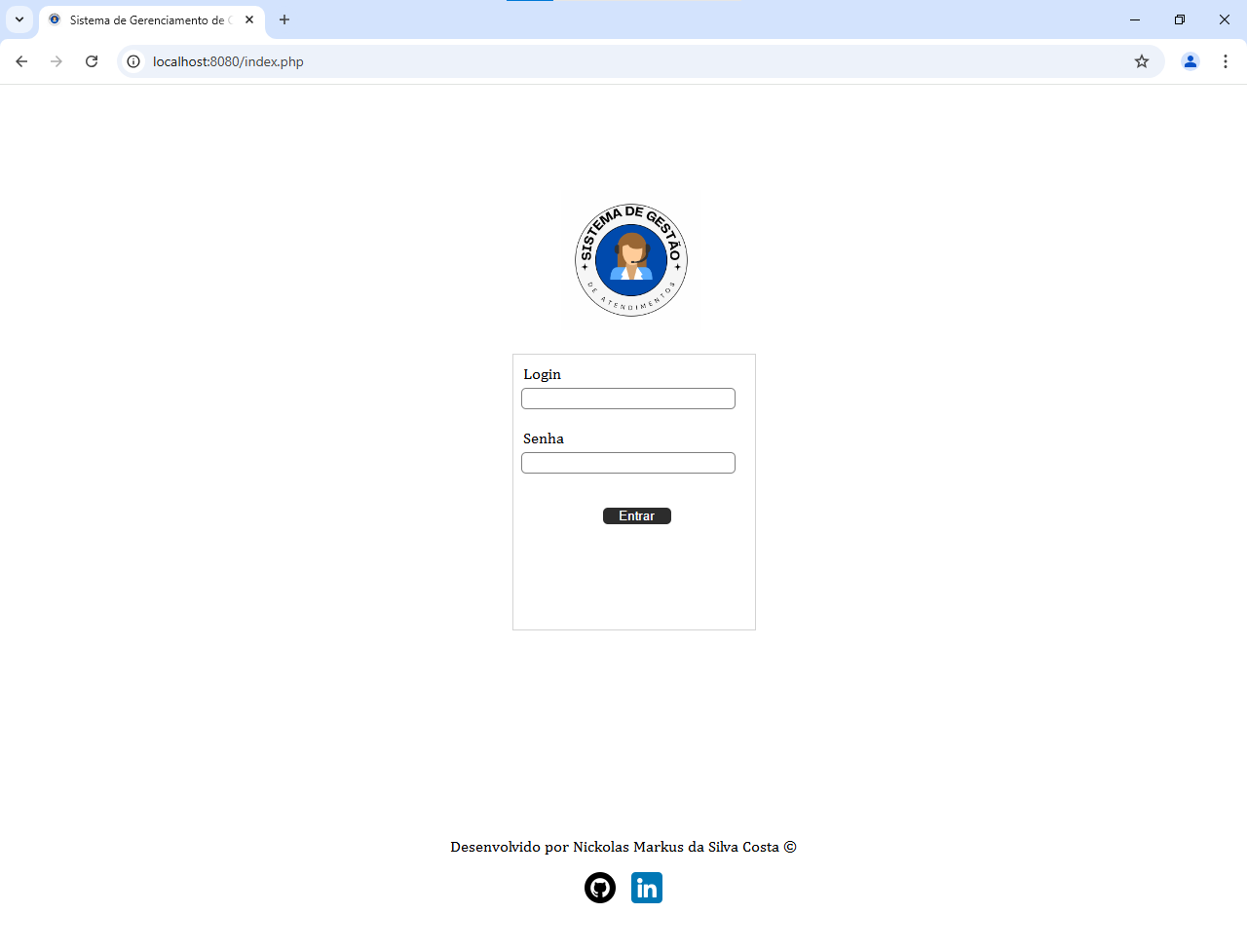
Nesse capítulo, são abordados os resultados que foram obtidos com o desenvolvimento do Sistema de Gerenciamento de Clientes.

4.1 TELA DE *LOGIN*

Ao acessar a URL do sistema, o primeiro contato do usuário com ele é a página de *login*. Nessa tela, ele irá colocar o seu *login* e a senha que foi cadastrada anteriormente pela empresa. Ao colocar essas informações, o sistema faz a autenticação de segurança e permite o acesso.

Caso o usuário esteja com *status* inativo, ele não irá obter sucesso no acesso, pois o sistema verifica o *status* do usuário e permite o acesso apenas aqueles que estão ativos. A figura 30 mostra o resultado final da tela de *login* do sistema.

Figura 30 – Tela de *login*



Fonte: Desenvolvimento próprio (2024)

4.2 TELA PRINCIPAL

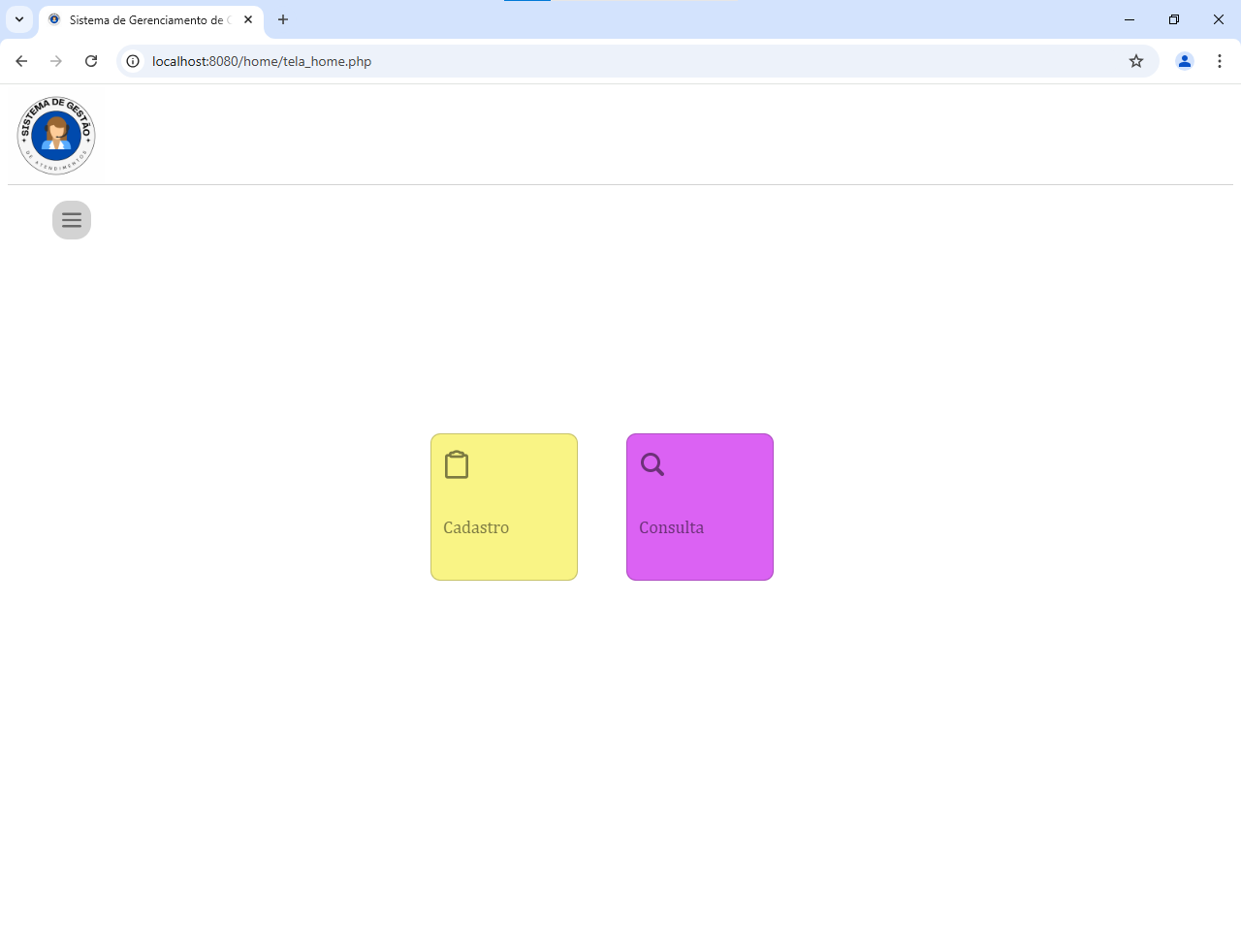
Ao acessar o sistema, o usuário visualiza a tela principal do *software*, conhecida como *home*. O sistema é composto por duas partes: o cabeçalho, onde se encontra um ícone para identificação, e o corpo, onde se visualiza o conteúdo da tela. O cabeçalho está em todas as telas, enquanto o conteúdo do corpo varia de acordo com a tela que o usuário se encontra.

No caso da tela *home*, o corpo é composto por três ícones, sendo eles:

* Cadastro: ao clicar nesse ícone, o usuário será encaminhado para a tela de cadastro;
* Consulta: nesse ícone, o sistema irá encaminhar para a tela de consulta;
* *Menu* lateral: ao apertar nesse botão, será aberto um *menu* lateral, no qual se encontram mais opções para o usuário, bem como um botão para que ele possa sair do sistema.

A figura 31 demonstra o resultado final do processo de desenvolvimento da tela principal do sistema.

Figura 31 – Tela principal

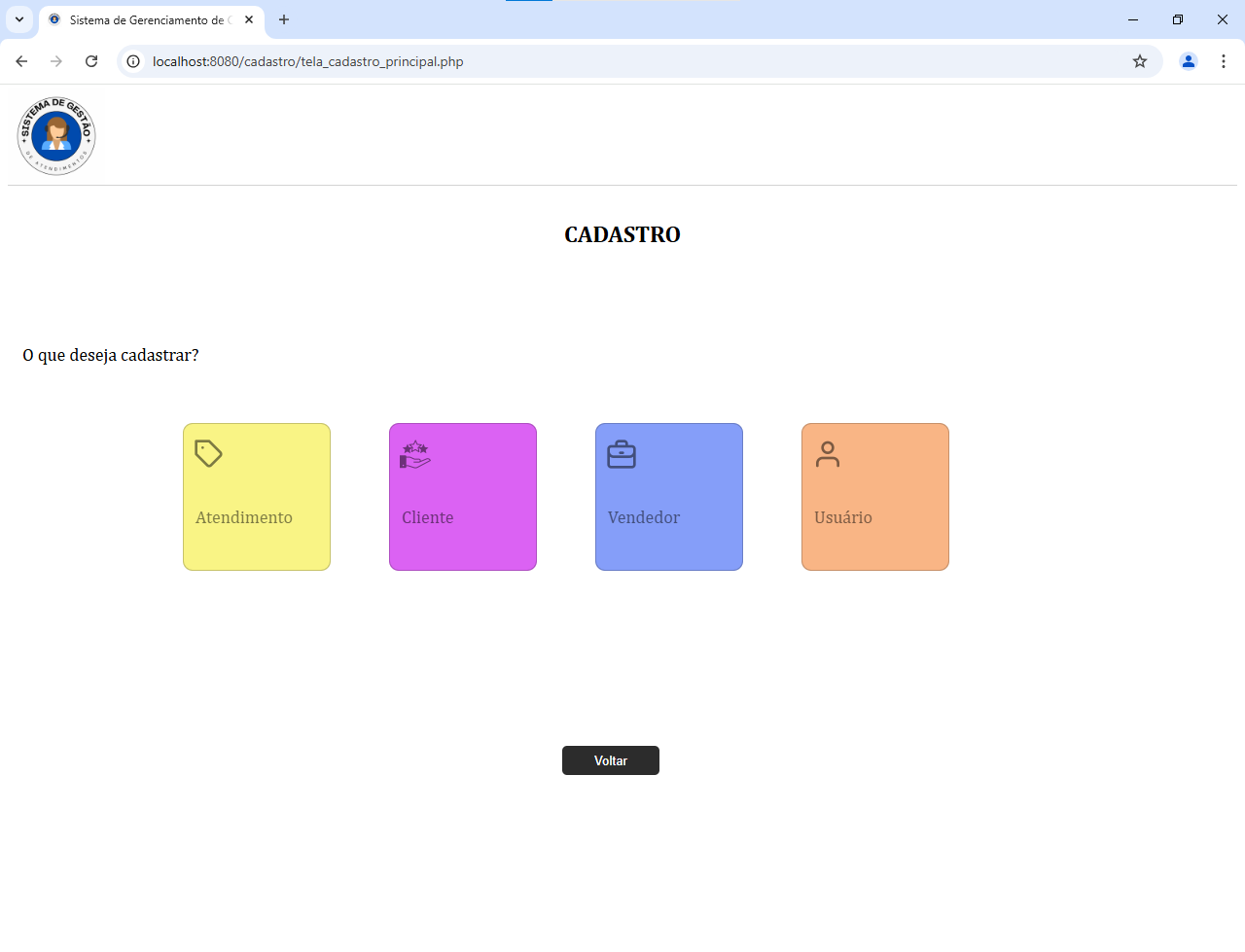


Fonte: Desenvolvimento próprio (2024)

4.3 TELA DE CADASTRO

Ao clicar na opção de cadastro, o usuário será direcionado para a tela de cadastro. Nessa página, ele pode escolher uma das opções de cadastro no sistema, que são: atendimento, cliente, vendedor e usuário. A figura 32 permite visualizar a tela de cadastro do sistema.

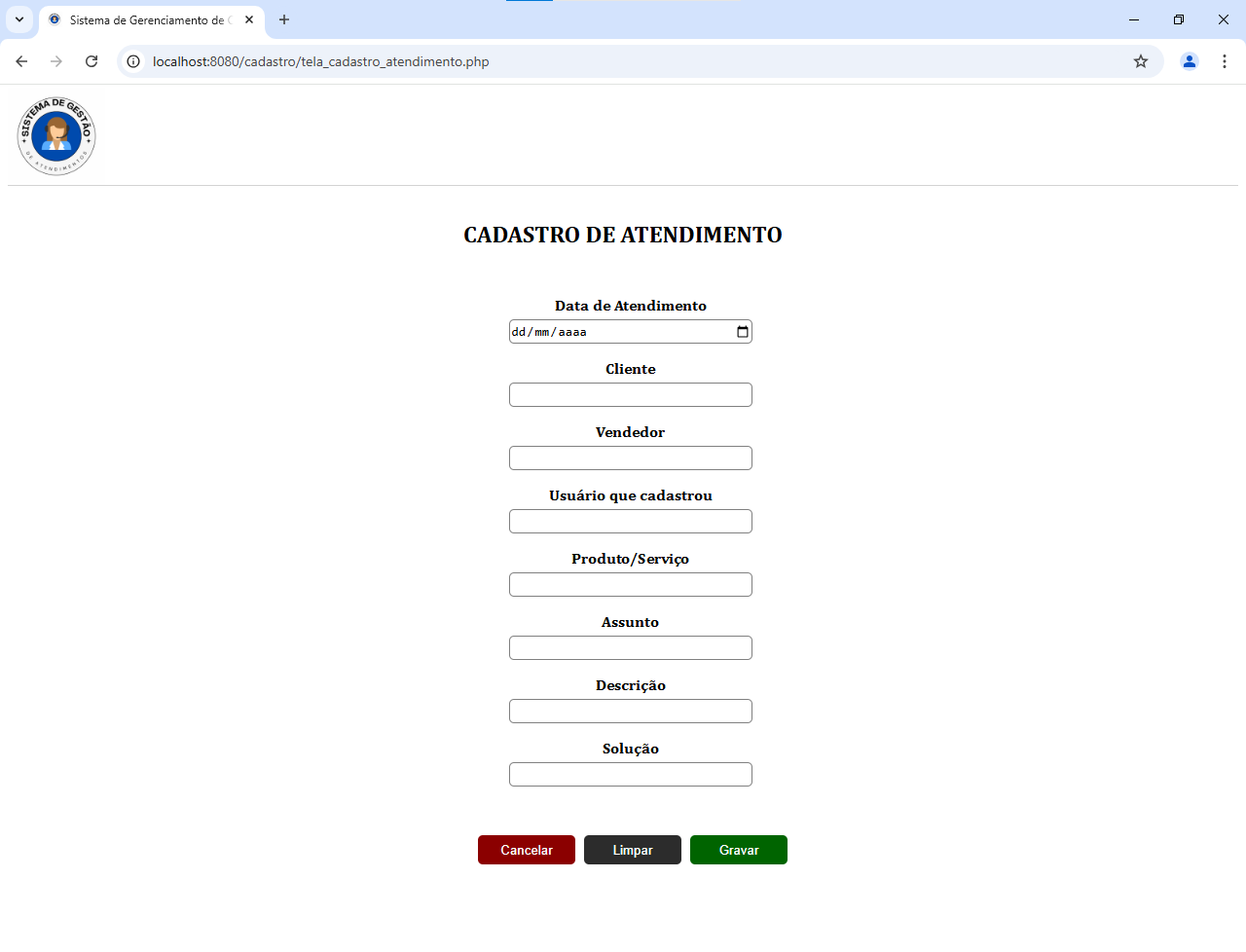
Figura 32 – Tela de cadastro



Fonte: Desenvolvimento próprio (2024)

Cada opção de cadastro é composta por um formulário, onde o usuário irá inserir os dados referentes àquela determinada opção que ele escolheu. Ao final da do formulário, há três botões, onde ele pode gravar os dados no sistema, limpar os campos do formulário ou cancelar a operação, no qual o sistema irá direcioná-lo para a tela principal. Ao apertar o botão de gravar, o sistema exibe uma mensagem de que a operação foi realizada e exibe um botão para voltar para a tela principal, conforme demonstrado na figura 33.

Figura 33 – Formulário de cadastro do atendimento



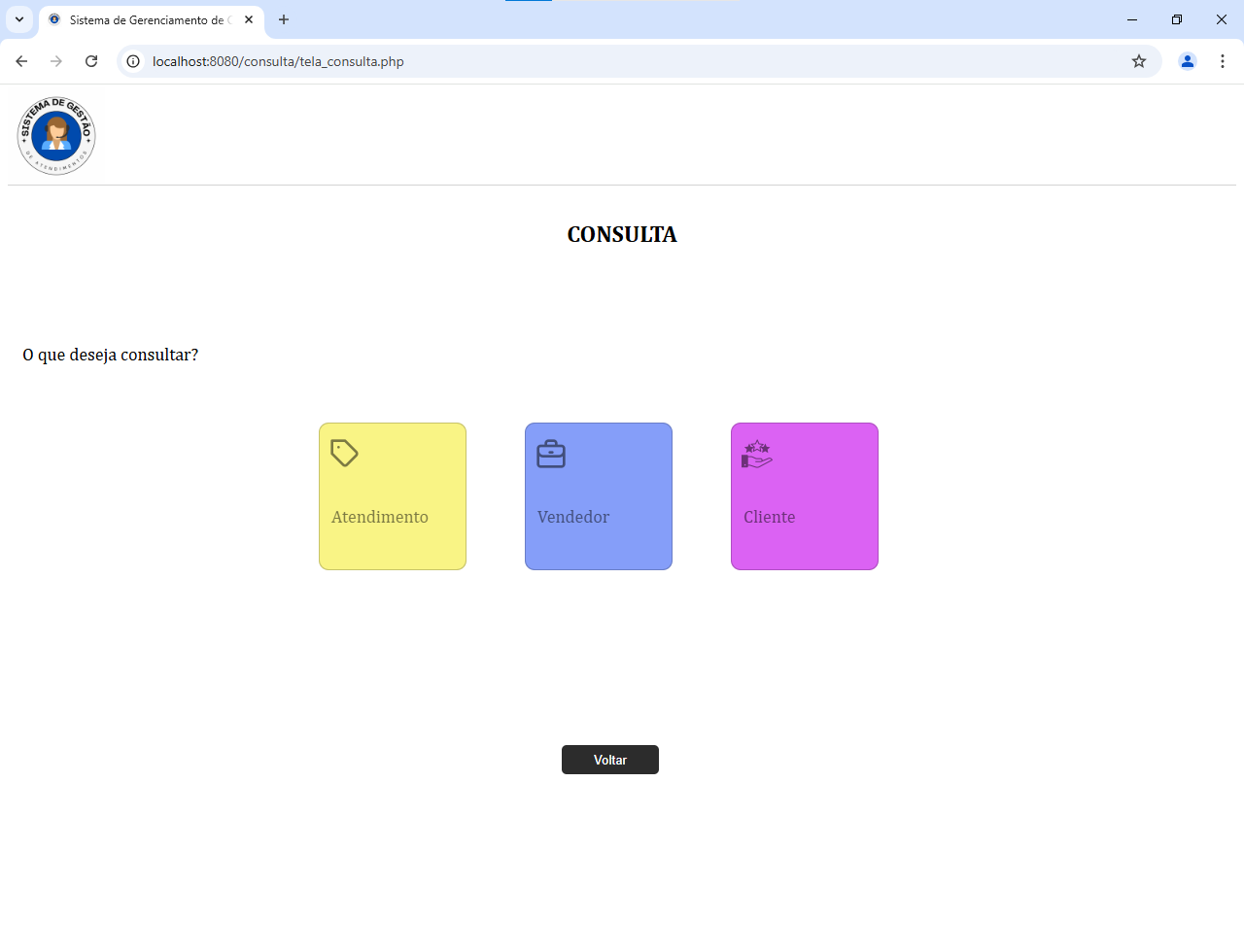
Fonte: Desenvolvimento próprio (2024)

No caso do cadastro do atendimento, o usuário só poderá cadastrar um novo se tanto o cliente quanto o vendedor já possuir cadastro no sistema. Em relação ao cadastro do cliente e do vendedor, o sistema não permite registros duplicados, ou seja, se o usuário tentar cadastrar um cliente ou um vendedor e o mesmo já possuir cadastro, o sistema irá retornar um erro e não irá registrar os dados.

4.4 TELA DE CONSULTA

Ao clicar na opção de consulta na tela principal, o usuário é encaminhado para a tela de consulta, onde o sistema oferece três opções de consulta dos dados cadastrados no sistema do atendimento, do cliente e do vendedor. A figura 34 demonstra a tela de consulta do sistema.

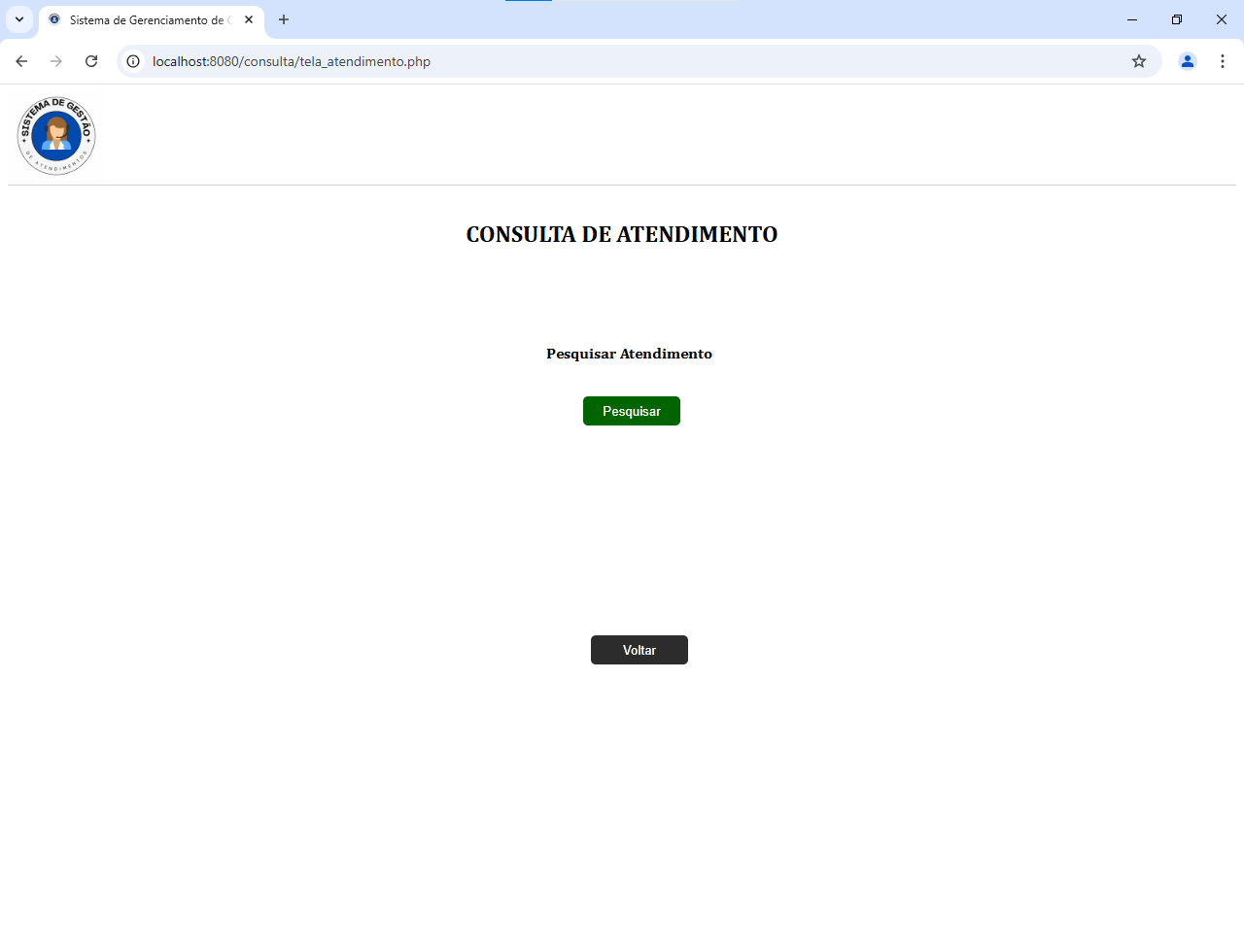
Figura 34 – Tela de consulta



Fonte: Desenvolvimento próprio (2024)

Ao clicar em uma dessas opções, o usuário terá um botão que, ao ser clicado, irá exibir os dados que estão cadastrados no sistema referente a opção que ele escolheu. Esse mesmo *layout* também se aplica às outras telas. A figura 35 demonstra a tela de consulta do atendimento.

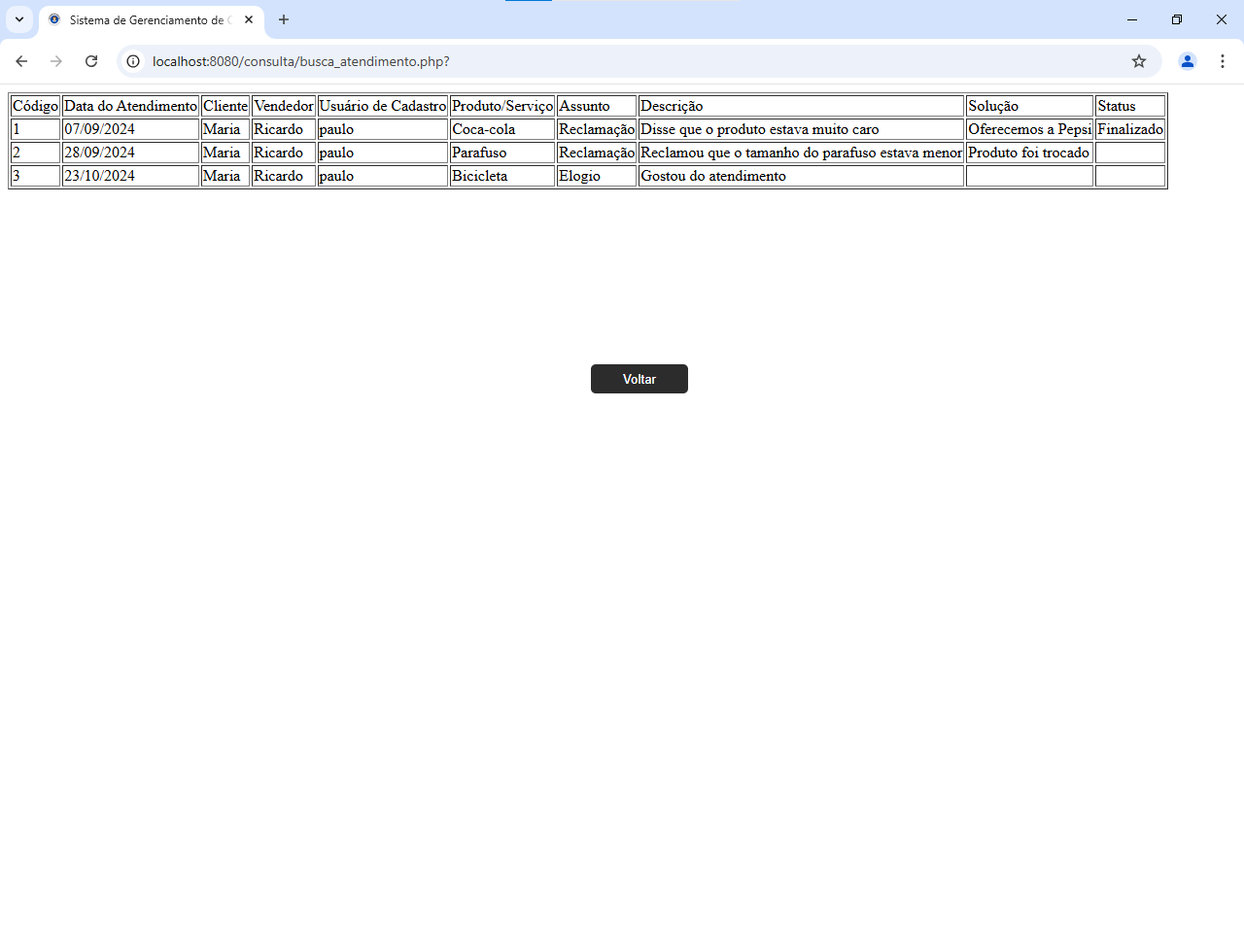
Figura 35 – Consulta de atendimento



Fonte: Desenvolvimento próprio (2024)

Na figura 36, é possível visualizar os resultados da busca, após o usuário clicar no botão de pesquisar.

Figura 36 – Resultado da busca

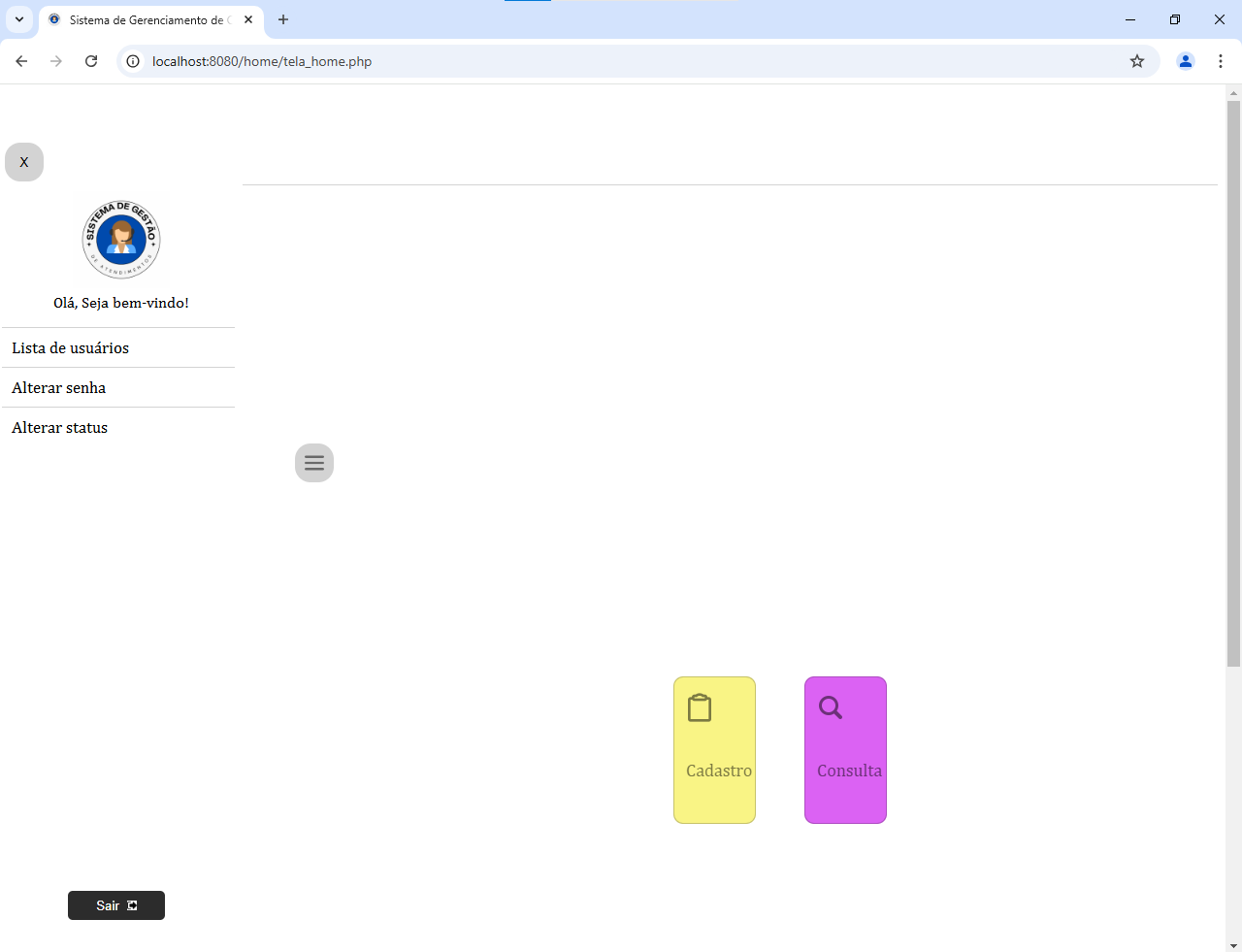


Fonte: Desenvolvimento próprio (2024)

4.5 OUTRAS FUNCIONALIDADES

Além de cadastrar e consultar, o sistema também dispõe de outras funcionalidades, nas quais estão disponíveis no *menu* lateral da tela principal, conforme ilustrado na figura 37.

Figura 37 – *Menu* lateral

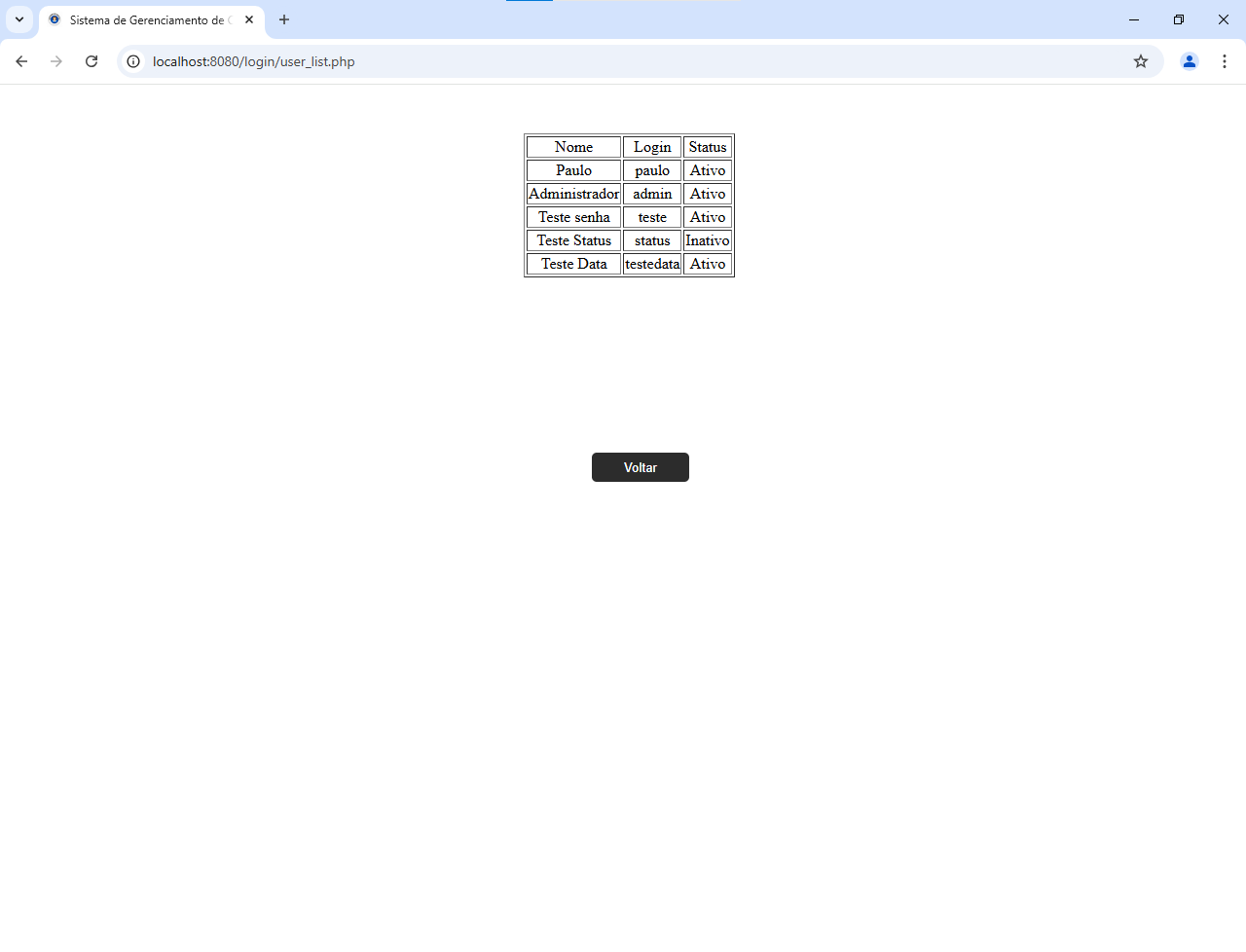


Fonte: Desenvolvimento próprio (2024)

Quem vê essa tela? Adm? Usuário fim? Quem acessa tem acesso a lista clientes?

A primeira opção do *menu* lateral é a lista de usuários. Nessa opção, o usuário será encaminhado para uma tela onde ele poderá visualizar uma lista de usuários que estão cadastrados no sistema. Diferentemente da tela de consulta, onde todos os dados do banco são exibidos, nessa tela são exibidos apenas o nome, o *login* e o *status* dos usuários. A figura 38 demonstra a tela de consulta dos usuários cadastrados no sistema.

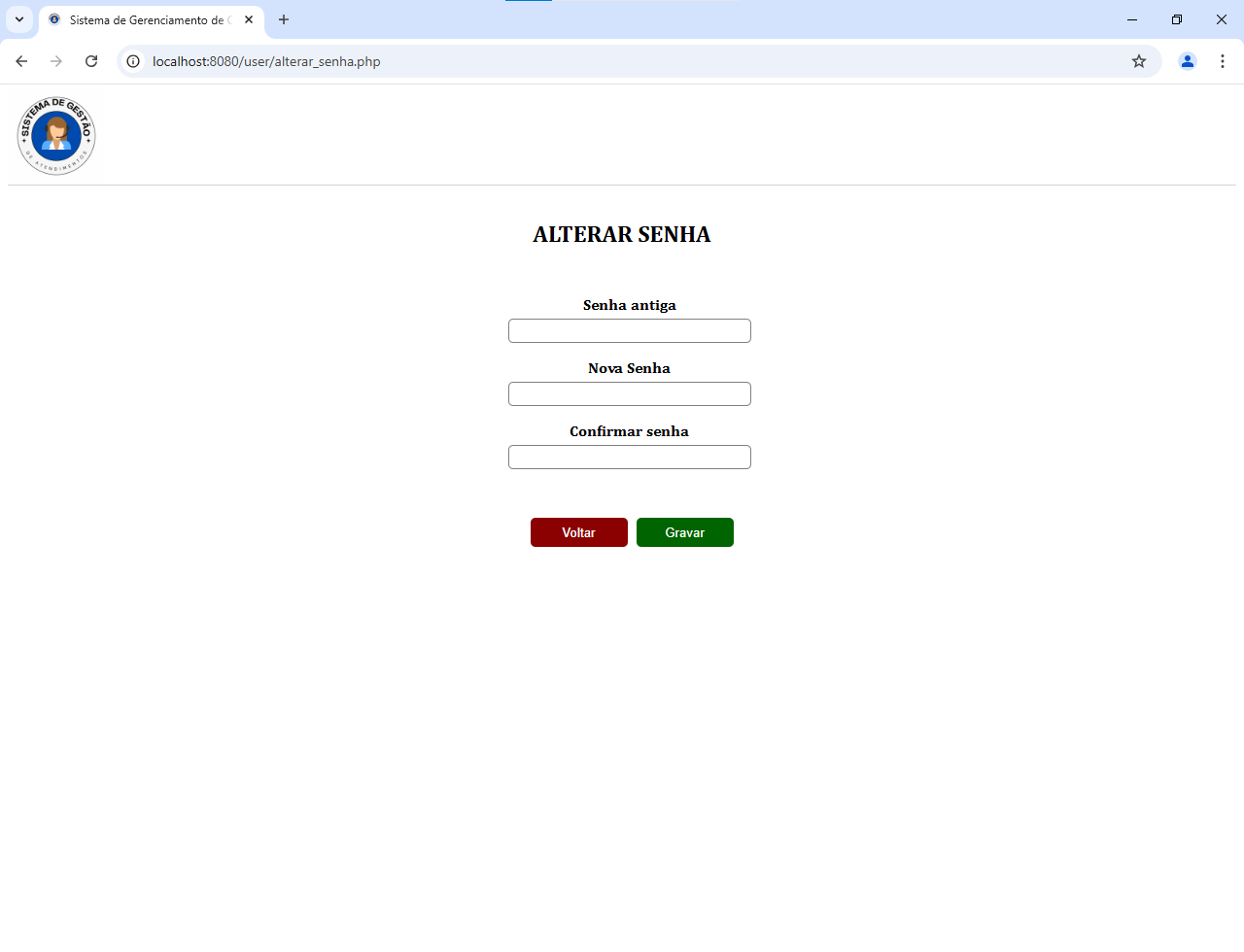
Figura 38 – Tela de consulta de usuários



Fonte: Desenvolvimento próprio (2024)

A segunda opção é de alterar a senha. Nela, o usuário será direcionado para um formulário onde ele irá informar a senha antiga e a nova senha, além de ter que preencher um campo para confirmar a nova senha. Após clicar no botão de gravar, o sistema realiza essa alteração. A figura 39 demonstra a tela de alteração de senha do sistema.

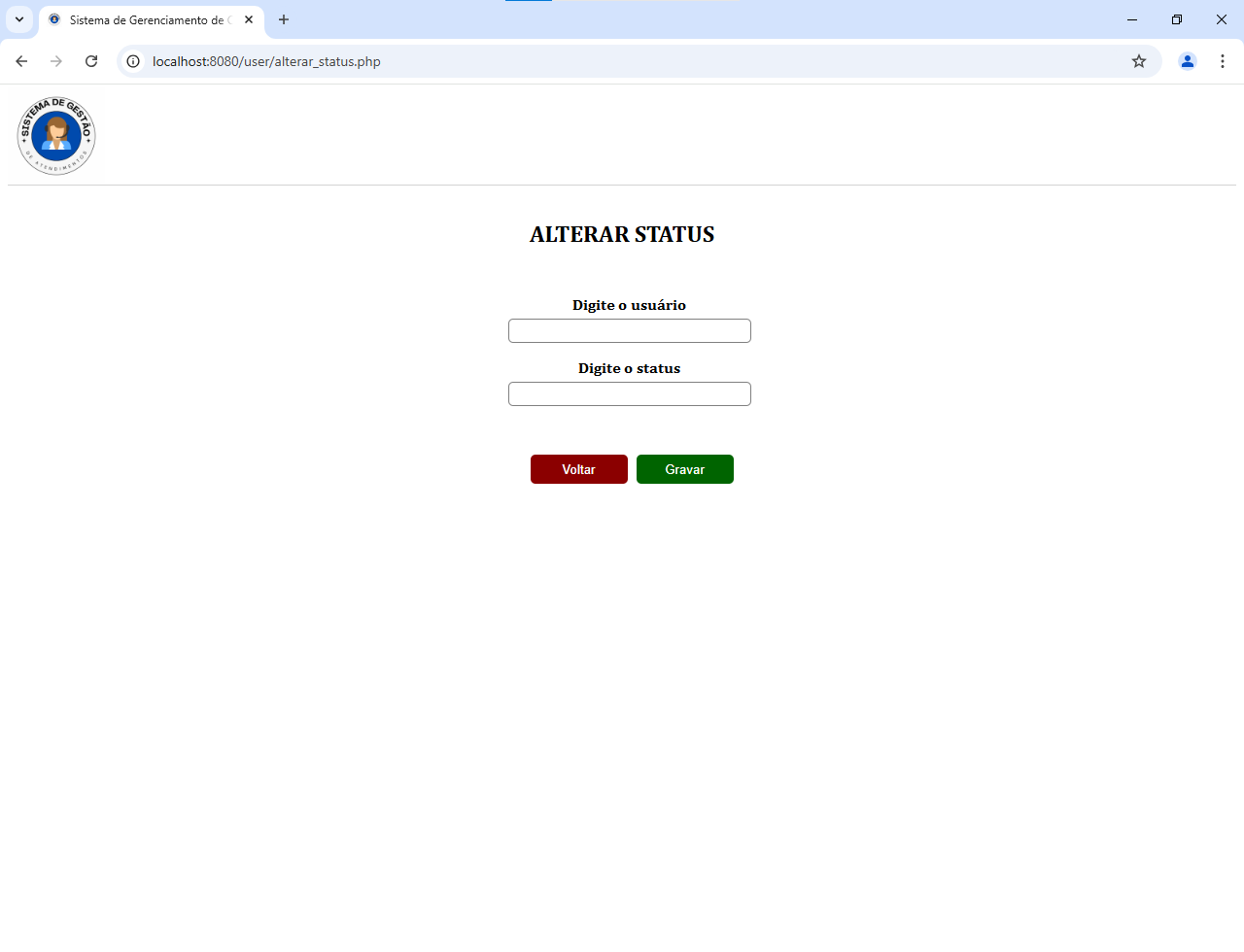
Figura 39 – Tela de alteração de senha



Fonte: Desenvolvimento próprio (2024)

A última opção consiste na alteração de *status*. Ao selecionar essa opção, o usuário será encaminhado para uma tela, onde ele terá que informar o usuário que deseja alterar o *status,* bem como o novo *status* que será definido para ele. Após isso, o sistema busca o usuário informado no banco de dados e, ao encontrar o registro, o mesmo faz a modificação do *status*. A figura 40 mostra o resultado final do processo de desenvolvimento da tela de alteração de *status*.

Figura 40 – Tela de alteração de *status*



Fonte: Desenvolvimento próprio (2024)

**5 CONSIDERAÇÕES FINAIS**

Sendo o consumo uma das principais atividades econômicas de um determinado local, é crucial que as empresas possuam meios de organização para seus processos de atendimento, a fim de evitar impactos negativos em seu fluxo de caixa e na sua imagem. Os *softwares* são uma das maneiras que elas utilizam para trazer essa organização, porém nem todas elas possuem recursos para adquirir um. E aquelas que possuem, acabam tendo problemas com os sistemas por conta da grande quantidade de funcionalidades ou de uma interface confusa e poluída. Vc conclui isso baseado em que?

Com base nos resultados obtidos do desenvolvimento do projeto proposto, é possível concluir que esse sistema, apesar de simples, é capaz de auxiliar os *stakeholders* no gerenciamento dos atendimentos aos clientes por conta da interface simples do *software*, o que traz segurança para o usuário quando for utilizá-lo. Ao cumprir os requisitos estabelecidos, o sistema traz segurança para as ações dos usuários e permite que eles utilizem o *software* da maneira mais correta possível. Além disso, outro fator importante, é a questão de sua disponibilidade, na qual é gratuito e de código aberto, o que promove a comunidade de desenvolvimento de *softwares open-source*, atraindo melhorias para o sistema, e facilita o caixa das pequenas empresas, que não possui um alto orçamento para adquirir *softwares* de gerenciamento pagos.de novo “concluir sem mostrar é tirar leite de pedra”

O presente projeto trouxe grandes desafios e novos conhecimentos para o autor do trabalho, pois a busca de soluções para os *bugs* e da consulta da bibliografia disponível permitiram essa expansão da compreensão teórica e prática dos conceitos trabalhados.

O projeto tem potencial para melhorias futuras e adições de novas funcionalidades, nas quais irão agregar mais ao funcionamento do sistema e transformá-lo em uma solução de referência na comunidade de desenvolvimento de *software*. Deve citar aspectos que contribuirão para isso.

**REFERÊNCIAS BIBLIOGRÁFICAS**

Não vi uma referência sobre gerenciamento de clientes, sistemas para tais gerenciamentos.

AGÊNCIA BRASIL. **Consumo nos lares brasileiros encerra 2022 com alta de 3,89%**. 2023. Disponível em: https://agenciabrasil.ebc.com.br/economia/noticia/2023-01/consumo-nos-lares-brasileiros-encerra-2022-com-alta-de-389. Acesso em: 16 jul. 2024.

ALVES, William Pereira. **Linguagem e Lógica de Programação**. 1. ed. São Paulo: Érica, 2014.

AQUILES, Alexandre; FERREIRA, Rodrigo. **Controlando Versões com Git e Github**. 1. ed. São Paulo: Casa do Código, 2014.

ASCENCIO, Ana Fernanda Gomes; ARAÚJO, Graziela Santos de. **Estrutura de Dados**: algoritmos, análise de complexidade e implementações em Java e C/C++. 1. ed. São Paulo: Pearson Prentice Hall, 2010.

ATLASSIAN. **Conheça o Trello**. 2024. Disponível em: https://trello.com/guide/trello-101. Acesso em: 02 nov. 2024.

BENTO, Evaldo Junior. **Desenvolvimento Web com PHP e MySQL**. 1. ed. São Paulo: Casa do Código, 2021.

BROOKS, Frederick P. **O Mítico Homem-Mês**: ensaios sobre engenharia de software. 20. ed. Rio de Janeiro: Alta Books, 2018.

CARVALHO, Vinícius. **MySQL**: comece com o principal banco de dados *open* *source* do mercado. 1. ed. São Paulo: Casa do Código, 2015.

CORMEN, Thomas H.; LEISERSON, Charles E.; RIVEST, Ronald L.; STEIN, Clifford. **Algoritmos**: teoria e prática. 2. ed. Rio de Janeiro: Elsevier, 2002.

DALL'OGLIO, Pablo. **PHP**: programando com orientação a objetos. 2. ed. São Paulo: Novatec, 2007.

DATE, C.J. **Introdução a Sistemas de Banco de Dados**. 8. ed. Rio de Janeiro: Elsevier, 2003.

DIAS, Ariel da Silva. **Algoritmos e Linguagens de Programação**. 1. ed. São Paulo: Senac, 2023.

ELDEWEISS, Nina; LIVI, Maria Aparecida Castro. **Algoritmos e Programação com Exemplos em Pascal e C**. 1. ed. Porto Alegre: Bookman, 2014.

FERREIRA, Rodrigo. **Segurança em Aplicações Web**. 1. ed. São Paulo: Casa do Código, 2021.

FLANAGAN, David. **JavaScript**: o guia definitivo. 6. ed. Porto Alegre: Bookman, 2013.

FONTES, Edison Luiz Gonçalves. **Políticas de Segurança da Informação**. 1. ed. Rio de Janeiro: Escola Superior de Redes, 2015.

FOROUZAN, Behrouz A. **Comunicação de Dados e Redes de Computadores**. 4. ed. Porto Alegre: Bookman, 2010.

FRAGA, Bruno. **Técnicas de Invasão**: aprenda as técnicas usadas por hackers em invasões reais. 1. ed. São Paulo: Editora Labrador, 2019.

FRANCO, Cristiano Roberto. **Programação Orientada a Objetos**. 1. ed. Indaial: Centro Universitário Leonardo da Vinci, 2014.

G1. **Hackers causaram prejuízos a cerca de 25% das empresas brasileiras em 2022, diz pesquisa**. 2023. Disponível em: https://g1.globo.com/tecnologia/noticia/2023/03/08/hackers-causaram-prejuizos-a-cerca-de-25-das-empresas-brasileiras-em-2022-diz-pesquisa.ghtml. Acesso em: 06 out. 2024

GALVÃO, Luis Eduardo Ferreira. **O que são Portas de Rede?**. 2024. Disponível em: https://a3aengenharia.com.br/blog/network-port/. Acesso em: 05 set. 2024.

GIL, Antônio Carlos. **Como Elaborar Projetos de Pesquisa**. 4. ed. São Paulo: Atlas, 2002.

GOODRICH, Michael T.; TAMASSIA, Roberto. **Estrutura de Dados e Algoritmos em Java**. 4. ed. Porto Alegre: Bookman, 2007.

GRONER, Loiane. **Estrutura de Dados e Algoritmos em Javascript**. 2. ed. São Paulo: Novatec, 2018.

IBM. **Palavras-chave reservadas em C**. 2024. Disponível em: https://www.ibm.com/docs/pt-br/debug-for-zos/16.0?topic=programs-c-reserved-keywords. Acesso em: 28 set. 2024.

JUNIOR, Joaquim Celestino; PATRICIO, Robério Gomes. **Desenvolvimento para Web**. 2. ed. Fortaleza: EduECE, 2015.

KASPERSKY. **O que é hacking? E como se prevenir**. 2024. Disponível em: https://www.kaspersky.com.br/resource-center/definitions/what-is-hacking. Acesso em: 06 out. 2024.

LOPES, Renata. **O que são variáveis e constantes na programação?.** 2023. Disponível em: https://hub.asimov.academy/blog/variaveis-constantes-programacao/. Acesso em: 29 set. 2024.

MANZANO, José Augusto Navarro Garcia. **Algoritmos Funcionais**: introdução minimalista à lógica de programação funcional pura aplicada à teoria de conjuntos. 1. ed. Rio de Janeiro: Alta Books, 2020.

MANZANO, José Augusto N. G.; OLIVEIRA, Jayr Figueiredo de. **Algoritmos**: lógica para desenvolvimento de programação de computadores. 29. ed. São Paulo: Erica, 2019.

MELO, Ana Cristina Vieira de; SILVA, Flávio Soares Corrêa da. **Princípios de Linguagem de Programação**. 3. ed. São Paulo: Blucher, 2014.

MONARI, Willian. **Desconstruindo a Web**: as tecnologias por trás de uma requisição. 1. ed. São Paulo: Casa do Código, 2016.

PAOLESCHI, Bruno. **Almoxarifado e Gestão de Estoques**. 3. ed. São Paulo: Érica, 2019.

PODER 360. **Vendas no varejo crescem 3,7% na semana do Dia das Mães**. 2023. Disponível em: https://www.poder360.com.br/poder-economia/economia/vendas-no-varejo-crescem-37-na-semana-do-dia-das-maes/. Acesso em: 16 jul. 2024.

PRESSMAN, Roger S.; MAXIM, Bruce R. **Engenharia de Software**: uma abordagem profissional. 8. ed. Porto Alegre: Bookman, 2016.

RED HAT. **O que é IDE (Ambiente de desenvolvimento integrado)?.** 2023. Disponível em: https://www.redhat.com/pt-br/topics/middleware/what-is-ide#:~:text=A%20sigla%20IDE%20significa%20(Integrated,de%20usu%C3%A1rio%20gr%C3%A1fica%20(GUI). Acesso em: 02 nov. 2024.

RIBEIRO, Maria Ivanilse Calderon; COSTA, Juliana Braz da; BRAVIM, Jhordano Malacarne. **Projeto de Sistemas WEB**. 1. ed. Cuiabá: UFMT, 2015.

ROBBINS, Jennifer. **HTML & XHTML Pocket Reference**. 4. ed. Sebastopol: O’Reilly Media, 2009.

ROSSI, Thiago. **Escopo de Variáveis no Javascript**. 2024. Disponível em: https://www.dio.me/articles/escopo-de-variaveis-no-javascript. Acesso em: 29 set. 2024.

SEBESTA, Robert W. **Conceitos de Linguagens de Programação**. 11. ed. Porto Alegre: Bookman, 2018.

SERVIÇO NACIONAL DE APRENDIZAGEM INDUSTRIAL. **Arquitetura de Redes**. 1. ed. Brasília: Senai, 2012.

SOMMERVILLE, Ian. **Engenharia de Software**. 9. ed. São Paulo: Pearson, 2011.

SOUZA, Marco A. Furlan de; GOMES, Marcelo Marques; SOARES, Marcio Vieira; CONCILIO, Ricardo. **Algoritmos e Lógica de Programação**: um texto introdutório para a engenharia. 3. ed. São Paulo: Cengage Learning, 2019.

STAIANO, Fabio. **Designing and Prototyping Interfaces with Figma**. 1. ed. Mumbai: Packt, 2023.

TANENBAUM, Andrew; FEAMSTER, Nick; WETHERALL, David. **Redes de Computadores**. 6. ed. Porto Alegre: Bookman, 2021.

TAYLOR, Ian. **Web Programming with Go**. 1. ed. Victoria: GitforGits, 2023.

VELOSO, Thássius. **Seguro contra ataque hacker cresce 512% no Brasil**. 2024. Disponível em: https://cbn.globo.com/tecnologia/noticia/2024/10/04/seguro-contra-ataque-hacker-cresce-512percent-no-brasil.ghtml. Acesso em: 06 out. 2024.

W3SCHOOLS. **Java Strings**. 2024. Disponível em: https://www.w3schools.com/java/java\_strings.asp. Acesso em: 28 set. 2024.

WAMPLER, Dean. **Programação Funcional Para Desenvolvedores Java**. 1. ed. São Paulo: Novatec Editora, 2012.

Da defesa:

Flávio

Paradigma Funcional

Conhecimento sobre o sistema de gerenciamento de clientes

Papel do sistema de gerenciamento