

UNIVERSIDADE PAULISTA

ANDRÉ MENEZES GUILHERME

DIEGO KAUJON BISCAIA

ENZO NAGATOMO VIEGAS

JOÃO VITOR NARDINI CALISTO

MARCOS ANTÔNIO DA SILVA FERNANDES

MATHEUS HENRIQUE DE SOUSA CORDEIRO

INFRAESTRUTURA DE GESTÃO INTERNA DA STARTUP QUINOTEK:
Sistema ERP

SOROCABA
2022

ANDRÉ MENEZES GUILHERME

DIEGO KAUJON BISCAIA

ENZO NAGATOMO VIEGAS

JOÃO VITOR NARDINI CALISTO

MARCOS ANTÔNIO DA SILVA FERNANDES

MATHEUS HENRIQUE DE SOUSA CORDEIRO

**INFRAESTRUTURA DE GESTÃO INTERNA DA STARTUP QUINOTEK:
Sistema ERP**

Projeto Integrado Multidisciplinar do curso de Análise e Desenvolvimento de Sistemas, apresentado à Universidade Paulista - UNIP.

Orientador: Prof. Me. Waldir Antonio da Silva

**SOROCABA
2022**

ANDRÉ MENEZES GUILHERME

DIEGO KAUJON BISCAIA

ENZO NAGATOMO VIEGAS

JOÃO VITOR NARDINI CALISTO

MARCOS ANTÔNIO DA SILVA FERNANDES

MATHEUS HENRIQUE DE SOUSA CORDEIRO

INFRAESTRUTURA DE GESTÃO INTERNA DA STARTUP QUINOTEK:
Sistema ERP

Projeto Integrado Multidisciplinar do curso de Análise e Desenvolvimento de Sistemas, apresentado à Universidade Paulista - UNIP.

Aprovado em:

BANCA EXAMINADORA

_____/_____/_____
Universidade Paulista – UNIP

_____/_____/_____
Universidade Paulista – UNIP

_____/_____/_____
Universidade Paulista UNIP

RESUMO

Neste projeto Integrado Multidisciplinar serão abrangidas diversas práticas sobre conteúdos que envolvem a projeção e o desenvolvimento de uma empresa Startup chamada Quinotek, em um recente projeto de expansão, envolvendo os ambientes de trabalho, com seus devidos equipamentos e dispositivos, incluindo um sistema de redes de internet e compartilhamento de dados, as equipes de colaboradores que irão trabalhar para a empresa e as devidas regras de conduta que também contemplam as leis de ética que estarão inseridas nesse ambiente, que ao mesmo tempo, entram em contexto com a sua regra de negócio. A pequena empresa de Tecnologia Quinotek trabalha com aplicativos destinados à área da educação, mas precisa de um ambiente de controle de dados organizado para poder gerenciar e cuidar de seus negócios de uma forma melhor, por isso, a Startup contratou temporariamente uma equipe de engenheiros de Software de outra empresa para desenvolver esse sistema de gestão de dados. Com base nisso o trabalho Acadêmico Multidisciplinar irá montar esse Software de gestão de informações baseando-se em princípios fundamentais da engenharia de Software, com suas metodologias, cronogramas e fluxogramas de processos, detalhando cada etapa e seus devidos motivos. Esse Software será arquitetado e produzido em linguagem de programação C, assim como foi encomendado pela Quinotek, e além disso, nesse Software existirá um sistema de segurança baseado nas leis de proteção de dados brasileiras, em formato de criptografia, proteção ao usuário de forma a estabelecer transparência e também proteger as informações contidas no Software contra possíveis falhas.

Palavras-chave: Software; Gestão; Engenharia; Redes; Programação.

ABSTRACT

This Integrated Multidisciplinary project will cover several practices on contents involving the projection and development of a startup company called Quinotek, in a recent expansion project, involving the workplace, with its proper equipment and devices, including a network system of internet and data sharing, the teams of employees who will be working for the company and the proper rules of behavior that also contemplate the ethical laws that will be inserted in this environment, which at the same time, come into context with its business rules. The small technology company Quinotek works with applications aimed at the educational area but needs an organized data control system to be able to manage and take care of its business in a better way, so the startup temporarily hired a team of Software engineers from a third-party company to develop this data management system. Based on this, the Multidisciplinary Academic work will build this information management software based on fundamental principles of software engineering, with its methodologies, schedules, and process flowcharts, detailing each step and its respective reasons. This Software will be architected and produced in the C programming language, as it was commissioned by Quinotek, and furthermore, in this Software there will be a security system based on the Brazilian data protection laws, in encryption format, protection to the user in order to establish transparency and also protect the information contained in the Software against potential failures.

Keywords: Software; Management; Engineering; Network; Programming.

SUMÁRIO

1	CONTEXTO DA STARTUP QUINOTEK.....	9
1.1	O que é uma Startup?	9
1.2	O espaço e seu Código de Ética.....	9
1.2.1	Código de Ética.....	9
1.2.2	O Código de Ética na Startup Quinotek.....	10
1.3	O produto desenvolvido pela Quinotek para a área da Educação	10
2	O COWORKING DA QUINOTEK	11
2.1	Funcionamento do Coworking	11
2.2	Vantagens do Coworking	11
2.3	Desvantagens do Coworking	12
2.4	Os custos do Coworking	12
3	O SOFTWARE DE GESTÃO ERP INTERNO DA STARTUP	13
3.1	Linguagem de programação e IDE	13
3.1.1	O que é uma Linguagem de Programação.....	13
3.1.2	Linguagem de Programação em Alto nível.....	14
3.1.3	A IDE de programação	14
3.1.4	A IDE de programação escolhida pela equipe terceirizada	15
3.2	Sistema ERP e o funcionamento do Software para a Quinotek.....	15
3.2.1	Tela Inicial	15
3.2.2	Tela de Login	16
3.2.3	Tela de Redefinição de Senha	16
3.2.4	Tela de Menu Principal	16
3.2.5	Tela de Informações dos Clientes.....	16
3.2.6	Tela de Informações dos Funcionários	16
3.3	O Fluxograma do Software.....	17
4	OS FUNCIONÁRIOS E OS LUCROS DAS EMPRESAS	20

4.1	Tabelas de informações dos Funcionários.....	20
4.1.1	O Gerente de Projetos.....	21
4.1.2	Os Desenvolvedores Back-end e Front-end.....	21
4.1.3	O que é Front-end e Back-end, e suas diferenças.....	22
4.1.4	Analista de Projetos.....	22
4.2	Os Custos e Lucros.....	23
4.2.1	Custos e Lucros gerados pela Empresa Terceirizada	23
4.2.2	Custos e Lucros gerados pela Quinotek.....	24
5	METODOLOGIA DE DESENVOLVIMENTO DE SOFTWARE	26
5.1	Metodologias Tradicionais e Ágeis	26
5.1.1	Metodologia Tradicional em Espiral.....	27
5.1.2	Metodologia Tradicional em Cascata.....	28
5.2	A Metodologia de desenvolvimento da Quinotek	30
5.3	Cronograma de desenvolvimento do Software	31
6	REQUISITOS DE SISTEMA.....	32
6.1	O Sistema Operacional Windows e suas vantagens	32
6.1.1	Desvantagens do Sistema Operacional Windows.....	33
6.1.2	Conclusão sobre o Sistema Operacional	33
6.2	A Memória HD e SSD	34
6.3	A Memória RAM	35
6.4	O Desempenho do Software de Gestão	37
7	INFRAESTRUTURA DE REDES	38
7.1	O que são Redes Computacionais	38
7.2	Tipos de Redes	38
7.2.1	Rede WLAN	39
7.2.2	Rede LAN.....	39
7.3	Topologias de Rede.....	40

7.3.1	Topologia de rede Estrela	40
7.4	A Infraestrutura de Rede utilizada na Quinotek	41
8	SEGURANÇA DE DADOS, LGPD E CRIPTOGRAFIA	43
8.1	LGPD e proteção de dados	43
8.1.1	Como funciona a LGPD.....	44
8.1.2	Tipos de dados protegidos pela LGPD	44
8.1.3	A LGPD no Software de Gestão da Quinotek	45
8.2	Criptografia de dados.....	45
8.2.1	Tipos de Criptografia	45
8.2.2	A Criptografia no Software de gestão	46
9	MANUAL DO USUÁRIO	47
9.1	Tela Inicial	47
9.1.1	Tela de Login	47
9.1.2	Tela de Redefinição de Senha	48
9.1.3	Sair do Programa.....	49
9.2	Tela de Menu Principal.....	49
9.2.1	Tela de Dados dos Clientes	49
9.2.2	Tela de Dados dos Funcionários	50
9.2.3	Voltar à Tela Inicial	51
9.3	Tela de Dados dos Clientes.....	51
9.3.1	Visualizar Informações dos Clientes	51
9.3.2	Editar Informações dos Clientes	52
9.3.3	Cadastrar Clientes	53
9.3.4	Excluir Cadastros de Clientes	53
9.3.5	Gerar relatório em TXT	54
9.3.6	Voltar ao Menu Principal	54
9.4	Tela de Dados dos Funcionários.....	54

9.4.1	Visualizar Informações dos Funcionários	54
9.4.2	Editar Informações dos Funcionários	55
9.4.3	Cadastrar Funcionários	56
9.4.4	Excluir Cadastros de Funcionários	57
9.4.5	Gerar relatório em TXT	57
9.4.6	Voltar ao Menu Principal	57
10	O CÓDIGO EM C E SUAS FUNCIONALIDADES	58
10.1	Imagens dos Comentários no Código	58
	CONCLUSÃO	67
	REFERÊNCIAS	69

1 CONTEXTO DA STARTUP QUINOTEK

A Startup Quinotek é uma pequena Empresa Startup de TI (Tecnologia da informação) que começou em uma pequena sala em uma Faculdade, com pouquíssimos recursos, mas então com uma ajuda financeira, conseguiu se expandir para um pequeno edifício que possui um método de *Coworking* e é constituído de 1 andar e térreo, com 3 salas cada.

A proposta de negócios da Startup Quinotek é inovar sistemas de gestão e organização de informações, atuando nesse momento inicial principalmente na área da educação, tendo como principal objetivo ampliar a utilização da tecnologia nas escolas de ensino fundamental e ensino médio.

1.1 O que é uma Startup?

Uma Startup é uma empresa que está em seu momento inicial no mundo dos negócios, que visa seus ganhos a médio e curto prazo para crescer, e possui uma proposta diferente e inovadora para uma ou mais áreas, que propõe ideias incomuns, repetíveis e escaláveis comparadas ao mercado geral. As startups começam com pouca estrutura, poucos funcionários e empreendedores, e no caso da Quinotek não foi diferente (SEBRAE, 2022).

1.2 O espaço e seu Código de Ética

Nesse pequeno edifício de 6 salas, terá no primeiro andar duas salas para 2 gerentes e uma sala reservada para reuniões. No andar térreo terá 1 sala de recepção com uma recepcionista e 2 salas para 4 funcionários, sendo 3 deles desenvolvedores, e 1 prospector de clientes.

1.2.1 Código de Ética

O Código de Ética é um tipo de regulamento muito utilizado nas empresas como uma forma de prevenir riscos, e fornece um conjunto de regras, orientações e valores que vão estabelecer ações aos seus colaboradores e terceiros. É um documento que

vai orientar sobre as regras e o padrão de comportamento que deve ser seguido por todos os membros de uma empresa, independentemente do nível hierárquico. Além disso, irá deixar claro quais são os mecanismos disponíveis caso uma pessoa queira relatar um caso de desvio de conduta, e irá apresentar quais atitudes não são aceitas naquele ambiente (ERREIRA, 2022).

1.2.2 O Código de Ética na Startup Quinotek

O Código de Ética da Quinotek não vai aceitar os seguintes desvios de conduta: Assédio moral e sexual, discriminação, fraudes, conflito de interesses, corrupção etc. O código de ética também vai garantir a proteção de dados, com priorização nos clientes, diversidade, respeito, proteção a quem realizar denúncias, uma conscientização com o meio ambiente e transparência.

Além disso, os funcionários terão que ter responsabilidade e comprometimento, não podendo fazer o consumo de nenhum tipo de bebida alcoólica e drogas de qualquer tipo (Que não se enquadre á medicamentos), no local de trabalho, sejam elas lícitas ou não. Os funcionários deverão ter empatia com seus colegas de trabalho, deverão ser fiéis a empresa e demonstrar relação de confiança aos clientes. Este Código de Conduta será colocado em cartazes que serão exibidos em todas as salas da Quinotek.

1.3 O produto desenvolvido pela Quinotek para a área da Educação

O produto que a empresa Quinotek está desenvolvendo para as escolas é um *Software* de gestão, cujo objetivo é organizar as funções de um diário de classe físico em um diário de classe digital, para facilitar o trabalho dos docentes em encontrar e trabalhar com as informações contidas no diário de classe de uma maneira mais fácil e prática, como também proteger essas informações para que elas não sejam perdidas e para que não caiam em mãos erradas. Ficarão responsáveis pelo desenvolvimento desse aplicativo uma equipe de 3 Desenvolvedores.

Este *Software* é considerado um *Software* Genérico, pois será vendida a assinatura deste produto para as escolas e prefeituras que desejarem adquirir o diário de classe digital.

2 O COWORKING DA QUINOTEK

A Quinotek é uma startup, logo precisa de um planejamento de curto a médio prazo para gerar renda no início de seus negócios, e pensando nisso, existe um meio que oferece um grande custo-benefício para a contratação de um espaço, onde nele várias empresas iriam trabalhar juntas dividindo o local, esse meio é chamado de *Coworking* (Trabalho compartilhado). Logo, a Quinotek irá adotar esse método para suas atividades de negócios.

2.1 Funcionamento do Coworking

No *Coworking* é feita uma prática de compartilhamento de espaço entre empresas de diversos ramos e áreas diferentes, e os seus serviços podem variar, tanto de prestação de serviços de pessoas, como também ao emprego de mobílias e o próprio espaço físico, como uma sala de escritório por exemplo, as pessoas que irão trabalhar lá e os móveis que estão presentes nesse espaço. O *Coworking* é um serviço compartilhado de atividades que possui diversas vantagens para as empresas que irão trabalhar com ele (AFONSO, 2022).

2.2 Vantagens do Coworking

A primeira vantagem é o espaço físico, que integra mobílias adequadas a saúde dos envolvidos, design de interior produtivo para escritórios, limpeza e organização etc. (AFONSO, 2022).

Outra vantagem é a flexibilidade que terão algumas empresas, ou seja, que não precisarão trabalhar no espaço do *Coworking* todos os dias, como por exemplo uma empresa de manutenção de equipamentos e dispositivos, que fará essas manutenções periodicamente (AFONSO, 2022).

O custo-benefício de um Coworking entra em âmbitos de contas mensais como de energia, água e internet, por exemplo, e entre outros benefícios que temos ao observar a rotina de trabalho em um ambiente assim, como o *Networking* (Interação entre pessoas), possibilitando uma visão mais ampla de negócios e melhorando também o trabalho em equipe no geral (AFONSO, 2022).

2.3 Desvantagens do Coworking

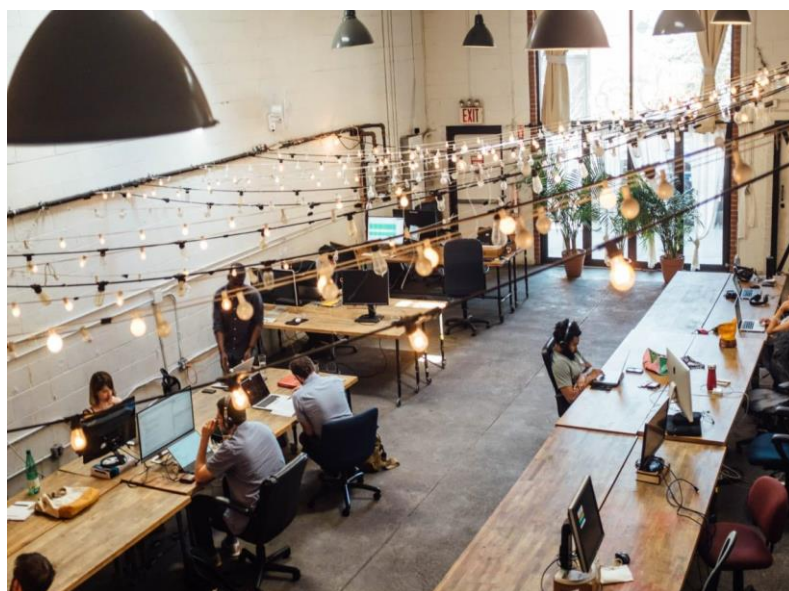
Algumas vezes também podem ser observadas adversidades em relação ao trabalho de vários colaboradores em um serviço compartilhado, como ruído e a falta de privacidade em uma rotina de trabalho, como também o compartilhamento de necessidades de várias pessoas a serem atendidas em um mesmo local e ao mesmo tempo (AFONSO, 2022).

2.4 Os custos do Coworking

O custo varia muito dependendo de localidade, tipos de serviços contratados e as empresas que irão oferecer esses serviços. Existe uma média aproximada de R\$ 700 a R\$ 800 para o contrato de cada empresa que irá trabalhar em um *Coworking*.

A Figura 1 abaixo, mostra um exemplo de espaço de *Coworking*.

Figura 1 - Espaço de *Coworking*



Fonte: Blog da Contabilizei, 2022.

3 O SOFTWARE DE GESTÃO ERP INTERNO DA STARTUP

A Quinotek como empresa também precisa de um sistema de gestão, e com a mudança de paradigma que a startup se encontra atualmente, foi contratada uma equipe para o desenvolvimento desse *Software*, composta por 6 Analistas de Sistemas de uma empresa terceirizada, que irão desenvolver a infraestrutura de redes e a Aplicação que será responsável por coordenar e documentar todas as informações da equipe de colaboradores e clientes envolvidos com a empresa, mostrando informações dos funcionários e dos clientes, como pagamentos de salários dos Funcionários e confirmações de pagamento das últimas faturas dos Clientes, e por fim, salvando essas informações em relatórios.

É importante apontar que este *Software* é sob encomenda, pois está sendo feito pela equipe terceirizada, exclusivamente para a empresa Quinotek.

Software consiste em: (1) instruções (programas de computador) que, quando executadas, fornecem características, funções e desempenho desejados; (2) estruturas de dados que possibilitam aos programas manipular informações adequadamente; e (3) informação descritiva, tanto na forma impressa quanto na virtual, descrevendo a operação e o uso dos programas (PRESSMAN; MAXIM, 2016, p. 4).

3.1 Linguagem de programação e IDE

No desenvolvimento de um *Software* uma das etapas mais importantes a se fazer é definir uma linguagem de programação e uma ferramenta adequada para se trabalhar com essa linguagem em questão, escolhida para se desenvolver um aplicativo.

3.1.1 O que é uma Linguagem de Programação

Uma linguagem de programação é uma linguagem formal que apresenta uma semântica e uma sintaxe composta por símbolos, palavras-chave e cálculos que, através de um Desenvolvedor que definirá regras e etapas, é capaz de dar instruções a serem seguidas por um sistema computacional, que interpretará ordens, que conterão dados lógicos, ações físicas e de controle de fluxo de informações. A

linguagem de programação é o meio de comunicação que um programador tem para se comunicar com um dispositivo computacional. (CARRER, 2019).

O estudo de linguagens de programação, como o estudo de linguagens naturais, pode ser dividido em exames acerca da sintaxe e da semântica. A sintaxe de uma linguagem de programação é a forma de suas expressões, sentenças e unidades de programas. Sua semântica é o significado dessas expressões (SEBESTA, 2018, p. 136).

3.1.2 Linguagem de Programação em Alto nível

Para que seja feito um contato mais indireto do programador com o dispositivo utilizado por ele, são utilizadas as linguagens de programação de Alto Nível, que são capazes de converter instruções e ordens de uma maneira mais compreensiva para facilitar o trabalho dos programadores que as utilizarem, proporcionando a possibilidade de utilização de termos e palavras utilizadas em idiomas normalmente falados pelo mundo, como o Inglês, português etc. (CARRER, 2019).

Da mesma forma que as outras ciências, a Computação utiliza a Matemática para determinar fatores precisos, além de uma notação poderosa que garanta abstrações corretas e raciocínios rigorosos. O objetivo da Lógica de programação é garantir nossa compreensão das linguagens e das ferramentas de programação (SOFFNER, 2013, p. 16).

A linguagem C, que é a linguagem escolhida para desenvolver o *Software* de gestão ERP, é uma linguagem de alto nível.

3.1.3 A IDE de programação

Existem ferramentas que possibilitam ambientes para a programação em alto nível, que trazem maneiras muito mais fáceis e até mesmo essenciais para se programar um *Software*, e essas ferramentas são chamadas de IDE (*Integrated Development Environment*) ou Ambiente Integrado de Desenvolvimento.

Essas IDEs podem proporcionar essa facilidade aos programadores por meio de funcionalidades como preenchimento de palavras no código, *Plugins*, gerenciamento de conexões com bancos de dados, integrações com sistemas de controle de versão, simuladores, compiladores de código e diversas outras funcionalidades integradas.

“Um compilador lê o programa inteiro e converte-o em um Código-Objeto, que é uma tradução do Código-Fonte do programa em uma forma que o computador possa executar diretamente” (SCHILDT, 1997, p. 9).

Existem diversas IDEs, dê das mais específicas, utilizadas para uma ou duas linguagens, como também existem IDEs que podem ser utilizadas para se programar em diversas linguagens. Uma das mais famosas dessas IDEs mais abrangentes é, por exemplo, o Microsoft Visual Studio, compatível com o sistema operacional Windows.

3.1.4 A IDE de programação escolhida pela equipe terceirizada

Uma IDE mais específica para se programar em C ou até em C++, *Open source* (Código aberto), com fácil instalação e diversas funcionalidades é a IDE CodeBlocks, que por ter código aberto, não requer licenças de *software*, evitando assim um possível custo a mais para o desenvolvimento do aplicativo de gestão.

3.2 Sistema ERP e o funcionamento do Software para a Quinotek

O tipo de *software* que a Quinotek está solicitando, as suas atividades fazem parte de um Sistema ERP (*Enterprise Resource Planning*), que significa Sistema Integrado de Gestão Empresarial.

Este tipo de *Software* de gestão empresarial faz com que os processos internos da Startup sejam unificados, controlando essas informações de diferentes áreas como vendas, finanças, contabilidade, estoque, compras, produção, logística etc., automatizando os seus processos manuais, e facilitando o fluxo de trabalho entre essas áreas.

3.2.1 Tela Inicial

A primeira tela será um menu inicial, onde será possível decidir qual será a próxima tela a ser mostrada, sendo elas a opção de login, redefinição de senha e sair do programa.

3.2.2 Tela de Login

A próxima tela, a de Login, vai funcionar para os 2 gerentes da Quinotek, que serão os únicos a utilizar o sistema e terão um cadastro pré-definido, do qual irá colocar o nome em um campo, e a senha em outro, para assim poder fazer o seu Login e entrar no menu principal.

3.2.3 Tela de Redefinição de Senha

Caso se esqueça da sua Senha, os 2 gerentes podem redefini-la colocando o seu nome e CPF, e após validar ele com o cadastro no sistema, pode gerar uma nova senha.

3.2.4 Tela de Menu Principal

No Menu terá as opções de telas de dados dos clientes, dados dos funcionários e voltar para a tela inicial.

3.2.5 Tela de Informações dos Clientes

Na tela de informações dos clientes, os administradores terão a opção de cadastrar um novo cliente que usufrui dos serviços da Quinotek, editar as Informações, visualizar e excluir informações de cadastros, gerar relatórios em formato de arquivo TXT e voltar para o menu principal.

As informações de cadastro dos clientes são: Nome da escola, CNPJ da instituição de ensino, e-mail, telefone, estado, cidade, bairro, rua e número, data de vencimento da fatura do mês e se já pagou ou não.

3.2.6 Tela de Informações dos Funcionários

Na tela de informações dos funcionários, os administradores terão a opção de cadastrar os funcionários da Quinotek, editar as Informações, visualizar e excluir

informações de cadastros, gerar relatórios em formato de arquivo TXT e voltar para o menu principal.

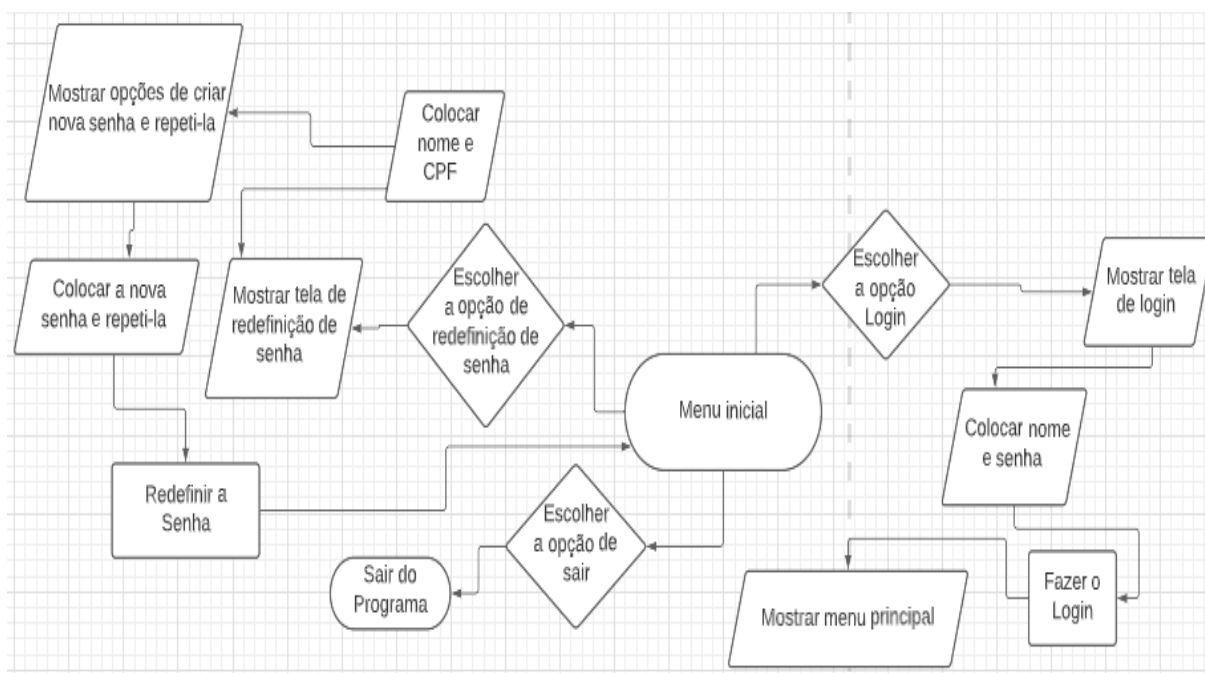
As informações de cadastro dos funcionários são: Nome, CPF, e-mail, telefone, estado, cidade, bairro, rua, número, complemento, cargo, data de admissão, datas de pagamento dos seus salários e os respectivos valores pagos.

3.3 O Fluxograma do Software

O fluxo de funcionamento desses processos de *Software* pode ser observado no fluxograma das figuras 2, 3, 4 e 5.

A Figura 2 mostra desde o menu inicial, até chegar no menu principal.

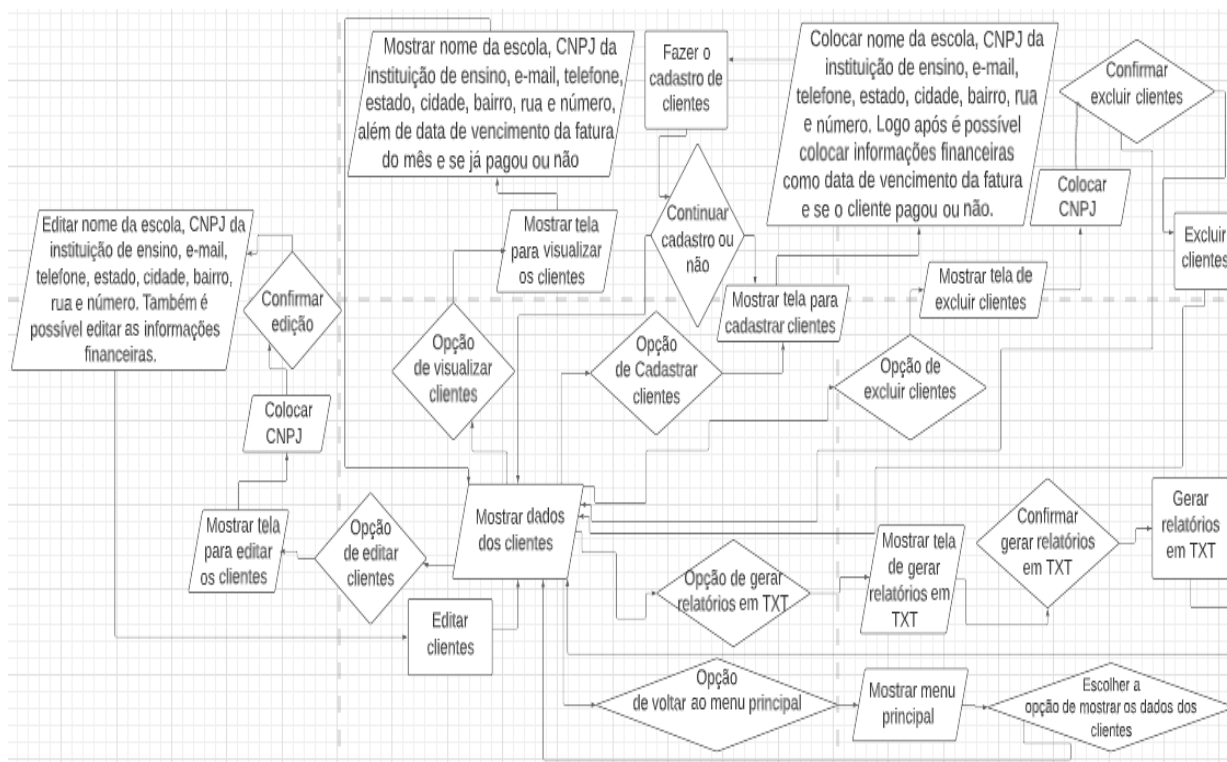
Figura 2 - Fluxograma de processos do Software (Parte 1)



Fonte: Autoria própria, 2022.

A Figura 3 começa do menu principal, representando apenas a opção de mostrar dados dos clientes.

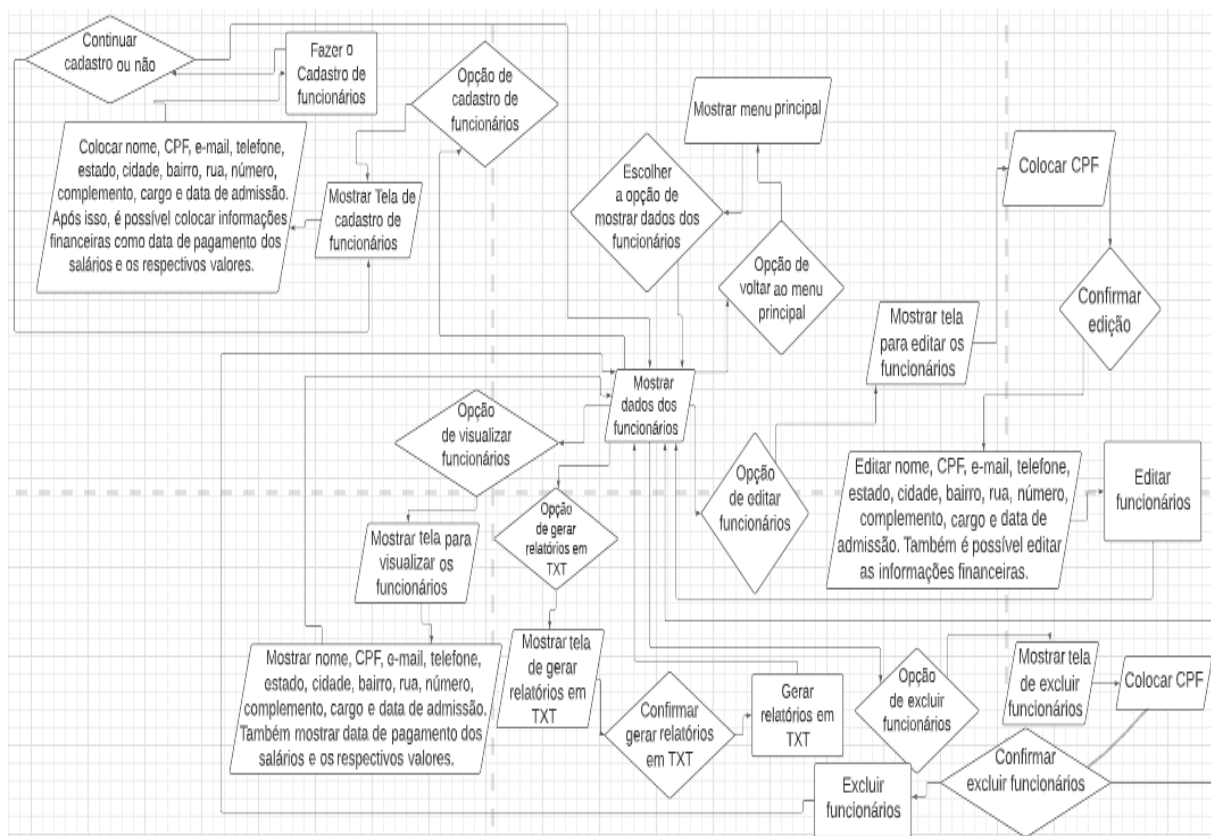
Figura 3 - Fluxograma de processos do Software (Parte 2)



Fonte: Autoria Própria, 2022.

A Figura 4 apresenta a segunda opção do menu principal: mostrar dados dos funcionários.

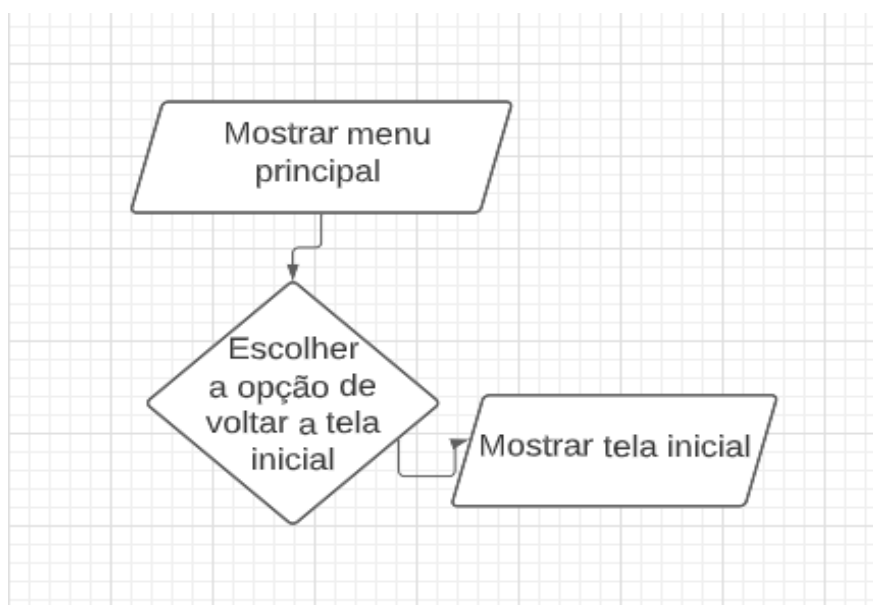
Figura 4 - Fluxograma de processos do Software (Parte 3)



Fonte: Autoria Própria, 2022.

A Figura 5 termina as opções do menu principal: voltar a tela inicial.

Figura 5 - Fluxograma de processos do Software (Parte 4)



Fonte: Autoria Própria, 2022.

4 OS FUNCIONÁRIOS E OS LUCROS DAS EMPRESAS

As tabelas a seguir, mostram informações dos Funcionários da Startup Quinotek (Tabela 1), e também da Empresa Terceirizada (Tabela 2).

4.1 Tabelas de informações dos Funcionários

Tabela 1 - Informações dos Funcionários da Startup Quinotek

NOME	CARGO	HORÁRIO DE TRABALHO	SALÁRIO BRUTO
Samira	Gerente geral	07:00 – 16:00	R\$8.000,00
Rodrigo	Gerente de funcionários	07:00 – 16:00	R\$6.500,00
Claudia	Recepcionista	07:00 – 16:00	R\$1.400,00
Ana Júlia	Prospector de clientes	07:00 – 16:00	R\$1.600,00 a R\$3.000,00
Enzo	Desenvolvedor	07:00 – 16:00	R\$3.500,00
Maria	Desenvolvedor	07:00 – 16:00	R\$3.500,00
Marcos	Desenvolvedor	07:00 – 16:00	R\$3.500,00

Fonte: Autoria Própria, 2022.

Tabela 2 - Informações dos Funcionários da Empresa Terceirizada

NOME	CARGO	HORÁRIO DE TRABALHO	SALÁRIO BRUTO
Giovanna	Gerente de Projetos	07:00 – 16:00	R\$8.250,00
Ewerton	Analista de Projetos	07:00 – 16:00	R\$3.000,00
Luiz	Desenvolvedor Front-end	07:00 – 16:00	R\$3.820,00
Júlia	Desenvolvedor Front-end	07:00 – 16:00	R\$3.820,00
Marcos	Desenvolvedor Back-end	07:00 – 16:00	R\$3.840,00
Mario	Desenvolvedor Back-end	07:00 – 16:00	R\$3.840,00

Fonte: Autoria Própria, 2022.

Os 6 desenvolvedores do projeto de criação do *Software* ERP da empresa terceirizada estão separados e categorizados por diferentes áreas do mercado de TI (Tecnologia da informação), tornando então essa equipe capacitada para os vários processos de desenvolvimento e arquitetura do *Software*.

4.1.1 O Gerente de Projetos

A primeira função a ser designada é a do gerente de projetos, esse profissional irá ser o responsável por se comunicar com os clientes, levantar os requisitos funcionais do *Software* e elaborar documentações, elaborar cronogramas com as devidas datas, elaborar custos de desenvolvimento, designar responsáveis por cada etapa e descrever tarefas a serem cumpridas respectivamente até a entrega do projeto. Esse profissional também é responsável por estar sempre atento para com o resto da equipe, já que ele irá cuidar de partes mais burocráticas do desenvolvimento, portanto a supervisão do trabalho é um dos seus maiores deveres (ESPINHA, 2019).

Além dessas coisas, um gerente de projetos também precisa ter uma boa capacidade de comunicação, tanto para satisfação ao cliente como também para poder gerenciar a equipe de desenvolvimento de forma eficaz e produtiva, e também é inevitável que uma das capacidades de um gerente de projetos seja a de trabalhar sob pressão, capacidade de determinar prazos e segui-los.

Para um Gerente de projetos também é fundamental um nível considerável de conhecimento prévio sobre todos os temas abordados no projeto, para que ele possa então ter um controle de qualidade na hora de liderar todos os outros integrantes da equipe. Considerando essas diversas responsabilidades essenciais para dar vida ao desenvolvimento de um projeto, o custo desse profissional também é mais alto do que dos demais envolvidos (ESPINHA, 2019).

4.1.2 Os Desenvolvedores Back-end e Front-end

Depois que as etapas de levantamento de requisitos funcionais, cronogramas, documentos, custos e demais responsabilidades iniciais principais, de um gerente de projetos, são determinadas e repassadas para a equipe, são os Desenvolvedores *Back-end* e *Front-end* que irão dar vida a criação do *software*, pois são os responsáveis por programar a aplicação utilizando uma IDE e uma linguagem de programação para tal. Porém, por mais que essas duas áreas sejam responsáveis pelos códigos e algoritmos, elas possuem várias diferenças, que irão determinar se uma etapa é de *Front-end* ou de *Back-end*.

4.1.3 O que é Front-end e Back-end, e suas diferenças

O programador *Front-end* é o profissional responsável por programar a parte mais gráfica de um *software*, como por exemplo, o design de um site ou de uma aplicação. O seu trabalho é essencial para o desenvolvimento do *software*, pois com ele serão definidas funções gráficas que o usuário final deseja ver em seu produto, assim como é ditado nos requisitos funcionais, para que seja mais fácil o entendimento do usuário final quando o *software* for entregue. O desenvolvedor *Front-end* trabalha em um meio de design mais próximo ao código, fazendo com que ele também precise ter conhecimentos sobre linguagens de programação, assim como, também, linguagens de marcação (SOUTO, 2022).

Assim como o profissional do *Front end*, o responsável pelo *Back-end* também trabalha com uma linguagem de programação, porém, no *Back-end*, a parte gráfica com elementos feitos para exibição final e design não são feitos, pois nessa área é trabalhada toda a parte lógica e matemática de um *software*, o código puro que irá dar funcionalidade aos elementos do *software*, incluindo etapas e partes do desenvolvimento *Front end*. No *Back-end* também são feitas etapas de segurança, ou de desenvolvimento que restringe o usuário final de ter acesso a uma parte específica de um código, como por exemplo, um banco de dados (SOUTO, 2022).

O trabalho desse profissional também está completamente ligado as outras duas áreas citadas acima, pois precisam um do outro, para que no final o produto esteja em perfeitas condições de atender aos seus usuários.

É importante citar também o chamado *Full Stack Developer*, que é o profissional que faz dos dois, tanto *Front-end*, quanto *Back-end*, porém para exercer essas duas funções é preciso que o profissional já tenha bastante experiência e maturidade como desenvolvedor, para que ele possa executar das duas funções de maneira a atingir bons resultados.

4.1.4 Analista de Projetos

O Analista de Projetos é um cargo que apresenta muitas similaridades com o de gerente de projetos, porém diferente do gerente, o analista é mais responsável por acompanhamento de perto do desenvolvimento, garantindo entregas consecutivas e

a execução das atividades de uma maneira mais geral, proporcionando controles de qualidade e de aceitabilidade de um produto com base em requisitos principais. O Analista de Projetos não necessariamente precisa ter um grande conhecimento sobre todas as áreas envolvidas nas etapas de criação de um *Software*, como *Back-end* ou *Front-end* por exemplo, pois sua vocação é mais direcionada a supervisão do trabalho em equipe. (COLOSSETTI, 2020).

4.2 Os Custos e Lucros

Toda empresa tem seus custos de manutenção de funcionários, espaço e desenvolvimento do produto que é comercializado por ela no geral, assim como os seus ganhos de lucro mensal, que dependendo do caso, também pode ser distribuído por certos colaboradores em forma de participação nos lucros.

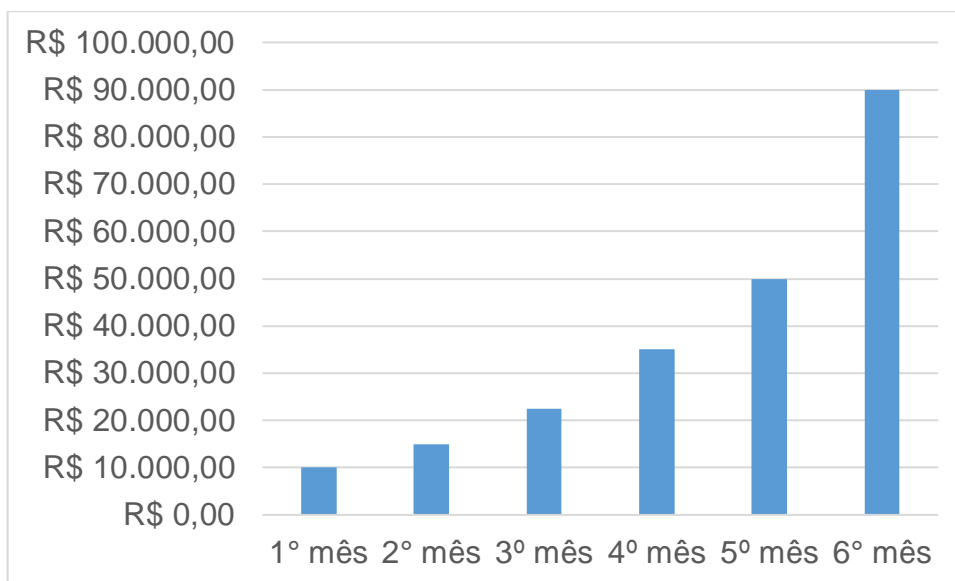
4.2.1 Custos e Lucros gerados pela Empresa Terceirizada

No caso da empresa que irá trabalhar no *Software* de gestão para a Quinotek, eles fizeram um custo de manutenção mensal após o fim do desenvolvimento do *Software*, mais o custo integral do *software*, esse último custo será pago ao final dos 6 meses de desenvolvimento pela Quinotek.

Após a entrega do *Software* é importante notar que, por causa da metodologia de desenvolvimento utilizada, será necessário uma manutenção para equipar ou então checar os dispositivos da Quinotek para se ter noção se eles serão capazes de executar a nova aplicação, e após isso, existirão manutenções periódicas mensais para se ter um controle de segurança quanto ao funcionamento eficaz do novo *Software* nos equipamentos da empresa.

O gráfico 1 mostra os lucros gerados pela empresa terceirizada durante o período de desenvolvimento do *software*.

Gráfico 1 - Crescimento de lucros da Empresa Terceirizada

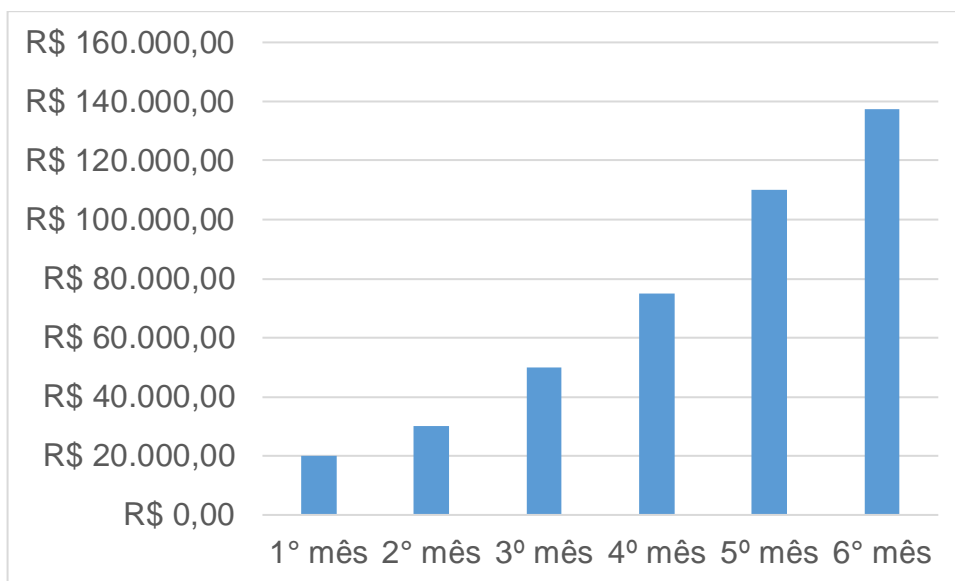


Fonte: Autoria Própria, 2022.

4.2.2 Custos e Lucros gerados pela Quinotek

No gráfico 2, observa-se os lucros gerados pela empresa Quinotek, com seus próprios desenvolvimentos do *Software* de educação, e também mostra uma pequena diferença de ganhos de lucro entre o 5º e o 6º mês do período de desenvolvimento do *software* de gestão, devido ao custo integral do *software* de gestão ser pago a empresa terceirizada após os 6 meses de desenvolvimento deles.

Gráfico 2 - Crescimento de lucros da Startup Quinotek



Fonte: Autoria Própria, 2022.

5 METODOLOGIA DE DESENVOLVIMENTO DE SOFTWARE

Cada *Software* possui as suas próprias características e precisa atender a necessidades específicas. A mesma lógica vale para os seus meios de desenvolvimento, que possuem as suas peculiaridades e parâmetros próprios dos diferentes tipos de sistemas. As metodologias de desenvolvimento de *Software* são um conjunto de abordagens que podem ser utilizadas para a criação de sistemas de processamento de dados (MONITORA, 2020).

5.1 Metodologias Tradicionais e Ágeis

Para Projetos de *Softwares* que vão ter uma maior documentação do projeto, as metodologias tradicionais são mais usadas, isso, com isso o conhecimento tem menos chance de ser perdido rapidamente, já que o período de desenvolvimento será maior, além de ter uma melhor estimativa de prazos e custos.

As etapas da metodologia tradicional devem ser seguidas sequencialmente, ou seja, enquanto a tarefa anterior não for cumprida, o processo não avança.

Já Metodologia Ágil, é uma metodologia de desenvolvimento de *Software* com foco no próprio projeto ou produto e acaba tendo foco em sua realização de melhorias e alterações mais frequentes, que são baseadas com base no Feedback (Opinião / Avaliação) dos usuários, clientes e equipe interna.

Sem ser um tipo de estrutura mais rígida, acontece períodos curtos de desenvolvimento, para que os resultados e Feedbacks sejam obtidos de maneira mais rápida, fazendo os problemas do *software* serem corrigidos ainda na fase inicial, garantindo uma qualidade nas primeiras versões do sistema.

A metodologia ágil possui vários princípios, e eles são os seguintes: ter prioridade em satisfazer o cliente por meio de entregas contínuas e rápidas, estar aberto a alterações durante o processo, realizar entregas do produto no menor período de tempo possível, manter a colaboração, dar ferramentas e suporte necessários, estimular a comunicação pessoal, o progresso consiste no *Software* em funcionamento, assim como o sucesso, manter um ritmo constante, manter atenção ao design e detalhes técnicos, tirar esforços que não geram valor ao produto, ter uma

boa arquitetura e atender aos requisitos do projeto, e em intervalos regulares a equipe faz uma reflexão sobre como melhorar a sua eficiência e eficácia (TOTVS, 2021).

5.1.1 Metodologia Tradicional em Espiral

No modelo em espiral, o desenvolvimento de *Software* é um ciclo iterativo que se repete até que os objetivos estabelecidos sejam alcançados, e tem a capacidade de lidar com o grande número de riscos e mudanças que podem ocorrer durante o desenvolvimento do *Software*, o modelo é demonstrado em forma de espiral, onde as diferentes etapas do modelo são distribuídas em diferentes ciclos, não sendo números de ciclos fixos, e isso pode variar de projeto para projeto (JACKSON 2022).

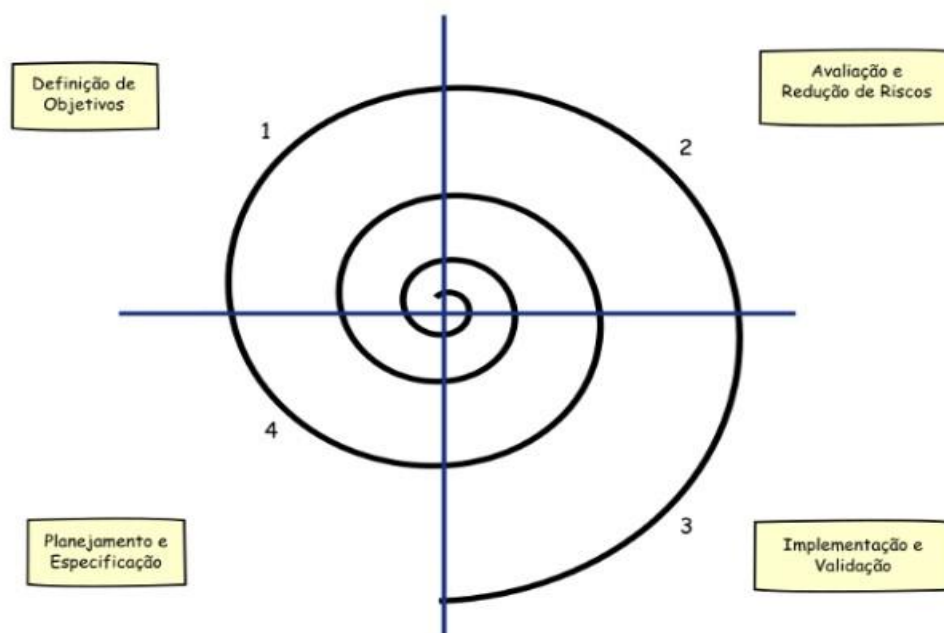
Este modelo é uma alternativa ao modelo tradicional em Cascata, que não é tão aberto a mudanças durante a execução do projeto.

O modelo espiral é uma abordagem realista para o desenvolvimento de sistemas e de software em larga escala. Como o software evolui à medida que o processo avança, o desenvolvedor e o cliente compreendem e reagem melhor aos riscos em cada nível evolucionário. (PRESSMAN; MAXIM, 2016, p. 49).

No modo em espiral, cada volta da espiral vai representar um ciclo completo, pelo qual sempre passam os quatro quadrantes, representando as quatro etapas do modelo. Essas etapas são executadas várias vezes em espiral e embora isso faça com que o projeto avance lentamente, o risco de que o processo de desenvolvimento falhe é cada vez menor (JACKSON, 2022).

Na figura 6 é representado um modelo em espiral.

Figura 6 - Exemplo de Modelo em Espiral



Fonte: Medium, 2019.

5.1.2 Metodologia Tradicional em Cascata

Este tipo de metodologia é recomendada para projetos menores e com requisitos bem claros.

Ela se baseia em uma hierarquia rígida, onde todas as etapas devem ser executadas sequencialmente, onde cada etapa depende da outra, uma etapa seguinte não pode ser feita sem que a anterior esteja concluída também. A previsibilidade do projeto e uma documentação mais extensa estão presentes também. Ela é marcada pelas seguintes etapas: levantamento de requisitos, planejamento, modelagem, desenvolvimento, testes e implantação (SACRAMENTO, 2021).

O modelo cascata, algumas vezes chamado ciclo de vida clássico, sugere uma abordagem sequencial e sistemática para o desenvolvimento de software, começando com a especificação dos requisitos do cliente, avançando pelas fases de planejamento, modelagem, construção e disponibilização, e culminando no suporte contínuo do software concluído (PRESSMAN; MAXIM, 2016, p. 42).

As etapas da metodologia em cascata começam com a análise de requisitos, que consiste em uma série de questionamentos feitos para o cliente, para que seja possível entender o que deve ser feito no *software*, quais são as suas necessidades

e assim estabelecer os requisitos funcionais principais do sistema, sendo então definido com isso, um escopo de projeto a ser passado aos demais profissionais da equipe de desenvolvimento. Na etapa de planejamento, com o escopo de projeto em mãos, será possível desenvolver a documentação, pontos como, orçamento do projeto, linguagens de programação a serem utilizadas para codificar o *Software*, prazos e as tarefas de cada pessoa na equipe de produção (SACRAMENTO, 2021).

Existe também a etapa da Modelagem, que consiste em planejar coisas como, plataformas utilizadas, interfaces de dados do sistema e modelos delas. Depois dessas etapas iniciais, na etapa de Desenvolvimento propriamente dito, é separar cada informação dos responsáveis por cada etapa, o escopo com as metodologias e requisitos funcionais, e então começar a fazer os códigos com suas principais funcionalidades e requisitos não funcionais necessários para o funcionamento do *Software* (SACRAMENTO, 2021).

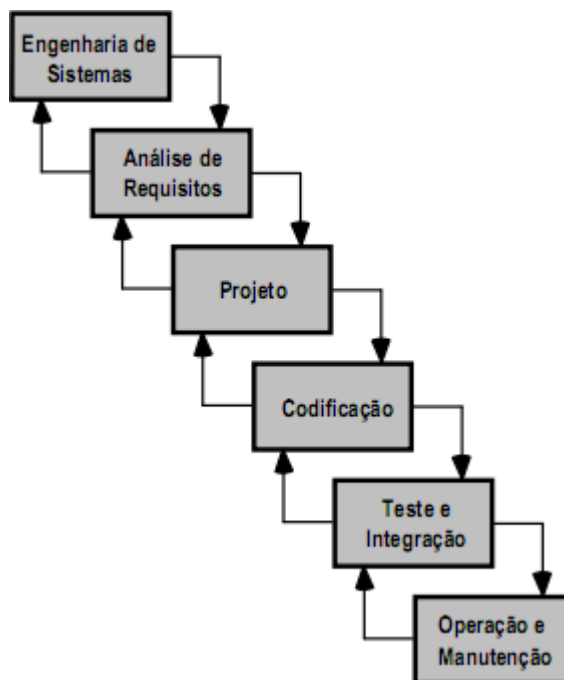
Na parte de testes o *Software* irá ser executado de diversas maneiras de várias formas para que seja garantido que suas funcionalidades estão funcionando como o planejado. É importante analisar todos os documentos anteriores novamente para verificar essas combinações ao realizar os testes, já se tendo um teste de qualidade no meio desse processo. A etapa final, a de Implantação, que significa que o *Software* já está pronto para ser entregue ao cliente, com todas os requisitos acima concluídos, incluindo o espaço em que o produto entrará em uso, sua manutenção e distribuição e o contato direto com o cliente para que seja avaliado o produto final e demais tipos de suporte ao usuário (SACRAMENTO, 2021).

As vantagens são que é uma metodologia adequada para projetos menores, acontece um planejamento e visão mais clara do projeto, uma previsibilidade melhor, sendo possível definir de antemão quanto vai utilizar de dinheiro e qual é o seu limite de prazo, além de fazer um detalhamento do projeto na documentação, onde tudo que é feito sempre será registrado, inclusive nos códigos de programação que serão comentados (SACRAMENTO, 2021).

As desvantagens incluem a burocracia que pode se tornar um processo demorado para avançar de etapa, e lidar com imprevistos e erros é algo difícil, é uma metodologia que não lida bem com mudanças. Além disso, o contato com o cliente acaba sendo muito pequeno durante o processo, deve-se esperar até o final do projeto para visualizar o resultado, não pode interferir no meio do desenvolvimento, já que existe um roteiro pré-definido (SACRAMENTO, 2021).

A figura 7 abaixo mostra um exemplo da Metodologia Cascata.

Figura 7 - Exemplo de Metodologia Tradicional em Cascata



Fonte: Felipe Lira Rocha, 2012.

5.2 A Metodologia de desenvolvimento da Quinotek

Devido ao projeto de *software* ser menor e com requisitos claros, o seu tempo de execução ser maior em comparação a Metodologia Ágil, depender de cada etapa ser seguida sequencialmente, e a alta necessidade de documentar tudo que for feito, a metodologia de desenvolvimento escolhida pela Quinotek será no Modelo Tradicional, em Cascata.

Devido a pouca comunicação com o cliente que acontece por usar essa metodologia, a equipe de desenvolvedores vai se comunicar com a Quinotek por Telefone, E-mail e também usando o Microsoft Teams, sendo um *software* parte do pacote Office, onde a empresa também vai utilizar outros serviços do pacote, como o Word, Excel, Powerpoint e o OneDrive.

5.3 Cronograma de desenvolvimento do Software

A tabela 3 a seguir, demonstra como foi definido e separado as tarefas e funções para cada cargo e com suas datas de entrega, tanto na parte da documentação, quanto da programação.

Tabela 3 - Cronograma de desenvolvimento do Software

Tarefa	Data	Quem vai fazer
Discutir o projeto	05/09 a 11/09	Todos os integrantes
Fazer o cronograma	05/09 a 11/09	Gerente de projetos
Descrever o software	12/09 a 18/09	Analista de projetos
Detalhar requisitos do software	12/09 a 18/09	Gerente de projetos
Fluxograma do software	19/09 a 25/09	Gerente de projetos
Definir Metodologia de desenvolvimento	26/09 a 02/10	Gerente de projetos
Tela inicial, login e redefinir senha	26/09 a 12/10	Desenvolvedor Front e Back-end
Contextualização da rede e intranet	03/10 a 22/10	Desenvolvedor Back-end
Telas de menu, cadastro, alteração, exclusão e exibição dos dados, além dos relatórios	13/10 a 30/10	Desenvolvedor Front e Back-end
Contextualização da LGPD, Criptografia e segurança	23/10 a 10/11	Desenvolvedor Back-end
Prática da rede e intranet	31/10 a 13/11	Desenvolvedor Back-end
Manual do software (para Funcionários)	11/11 a 25/11	Analista de projetos
Prática da LGPD, Criptografia e segurança	14/11 a 25/11	Desenvolvedor Back-end

Fonte: Autoria própria, 2022.

A definição do cronograma de projeto é o processo de decidir como o trabalho em um projeto será organizado em tarefas distintas, quando e de que maneira essas tarefas serão executadas. São estimados os prazos necessários para realizar cada tarefa e o esforço necessário, além de serem indicadas as pessoas que trabalharão nas tarefas que foram identificadas (SOMMERVILLE, 2019, p. 638).

6 REQUISITOS DE SISTEMA

Todo dispositivo eletrônico possui suas capacidades e limitações em relação ao poder de processamento e memória para armazenar dados. Sabendo disso, um *Software* também ocupa espaço interno dentro de uma memória em um dispositivo, assim como também precisa de um certo poder de processamento para conseguir processar e executar uma aplicação.

O *Software* de gestão ERP que está sendo desenvolvido para a Quinotek é bem simples e direto ao ponto, ou seja, não ocupa muito espaço de armazenamento e também não requer um grande poder de processamento para ser executado.

Também é ideal saber qual será o sistema operacional compatível com essa aplicação, já que ele será responsável por interpretar o algoritmo que irá fazer o *software* funcionar.

6.1 O Sistema Operacional Windows e suas vantagens

O sistema operacional mais utilizado do mercado atualmente é o Windows, da empresa Microsoft, e devido a sua grande popularidade, possui uma gama enorme de aplicações das mais recentes e mais conhecidas do mercado da tecnologia, assim como IDEs de programação (Integrated Development Environment) ou (Ambiente de Desenvolvimento Integrado) famosas e com muitas funcionalidades para auxiliar no desenvolvimento de *Softwares*, sendo então um sistema operacional ideal tanto para a equipe dos 6 desenvolvedores de sistemas, quanto também para a própria Quinotek.

Muitos outros *Softwares* que são muito utilizados para diversas atividades no ambiente de trabalho como as ferramentas do Pacote Office, e as ferramentas da Google como Gmail, Meet e Drive são compatíveis com o Windows (PÚBLIO, 2014).

Além disso é muito provável que a grande maioria das pessoas já tenha tido algum contato anterior com o sistema operacional Windows, eliminando então quase todo o processo de adaptação ao sistema operacional, fazendo com que a empresa e os colaboradores possam economizar esse tempo de adaptação para utilizar em outras tarefas importantes e agilizar o trabalho de todos (PÚBLIO, 2014).

O Sistema Operacional da Microsoft também conta com uma grande compatibilidade com a maioria dos equipamentos fabricados hoje em dia. Empresas

como a Intel, AMD e outras empresas de tecnologia já trabalham com dispositivos com o Windows como Sistema Operacional principal (CANTO, 2021).

O Windows também já conta com um sistema de segurança de fábrica, o Windows Defender Antivírus, ou seja, assim que comprado, o Sistema Operacional já terá defesas contra *Softwares* corrompidos e serviços mal-intencionados que poderiam, eventualmente, infectar o sistema e causar prejuízos ao bom funcionamento do equipamento e à segurança das atividades da empresa, caso o sistema operacional não possuísse um antivírus. Além disso com o Windows Defender Antivírus é possível evitar gastos de assinaturas de outros *Softwares* Antivírus que existem no mercado, e ainda assim ter uma segurança muito eficaz contra diversas ameaças, pois o Windows está constantemente sendo atualizado pela Microsoft (PÚBLIO, 2014).

6.1.1 Desvantagens do Sistema Operacional Windows

Todo Sistema Operacional possui suas falhas, uma delas no Microsoft Windows é que por mais que uma enorme quantidade de softwares sejam compatíveis com ele, nem todos são, como por exemplo *Softwares* exclusivos da Apple store, da empresa Apple, e entre outros *softwares* no mercado.

Outro ponto negativo é o fato de o Windows não ser um Sistema Operacional gratuito, ou seja, para adquiri-lo é preciso comprar uma chave do Windows para cada dispositivo que for utilizar o Sistema Operacional, o que não é barato. Como o Windows não é Open Source (Código Aberto), e é vendido por uma licença, não é possível fazer modificações através do seu código. O Pacote Office e suas ferramentas precisam ser adquiridos com licença separadamente, elevando ainda mais o custo para ter um ambiente de trabalho ideal (CANTO, 2021).

6.1.2 Conclusão sobre o Sistema Operacional

Devido aos fatores citados anteriormente, ficou definido que o Sistema Operacional a ser utilizado nos equipamentos da Quinotek cujo irão rodar o *Software* de gestão será o Windows 10, devido principalmente a suas compatibilidades de

hardware e de aplicações em geral, além de ser o sistema operacional de maior uso das pessoas. Logomarca do Microsoft Windows na Figura 8.

Figura 8 - Microsoft Windows 10



Fonte: Tecmundo, 2015.

A empresa Quinotek também vai pagar para utilizar o pacote Office, e assim utilizar o Word, Excel, PowerPoint etc.

6.2 A Memória HD e SSD

Existem dois tipos de disco principais de armazenamento de dados na grande maioria dos computadores encontrados no mercado, esses discos de armazenamento são os HDs e SSDs.

HD (*Hard Disk*), é uma memória secundária relativamente antiga, porém, até hoje muito utilizada, pois consegue armazenar uma grande quantidade de dados e também é muito durável.

Assim como a memória RAM, o disco rígido (ou HD) armazena programas e dados, porém existem algumas diferenças. O disco rígido tem uma capacidade muito maior. Seus dados não são apagados quando o computador é desligado (VASCONCELOS, 2007, p. 3).

Porém, como se trata de uma tecnologia antiga, já é de se esperar que nos dias de hoje já teríamos componentes diferentes, como o SSD (*Solid State Driver*), que é uma memória secundária assim como o HD, porém com a diferença de que o SSD tem menos espaço interno de armazenamento de dados, mas por outro lado, possui

muito mais velocidade de leitura e escrita de dados, fazendo então com que o SSD possa executar muito mais rápido as aplicações, ligar os computadores com mais velocidade e renderizar processos de forma mais consistente com o auxílio do processador e da memória RAM.

Na Quinotek, além do *Software* de gestão que terá nos computadores, também existirá outros aplicativos como o Pacote Office e entre outros *Softwares* como os que a própria Startup produz, no geral não será necessário uma quantidade exorbitante de espaço de armazenamento na memória secundária dos computadores da empresa, e pelo custo benefício, principalmente vindos da velocidade de execução de tarefas e inicialização dos sistemas foi então decidido que a Quinotek utilizará SSDs de 240 GBs (*Gigabytes*) em suas máquinas. A figura 9 mostra um HD e um SSD.

Figura 9 - HD e SSD



Fonte: Hardware, 2022.

6.3 A Memória RAM

A memória RAM, (*Random Access Memory*) ou Memória de acesso aleatório é uma memória volátil feita para auxiliar o processador na leitura e escrita de dados, dando acesso instantâneo para as informações acessadas pelo processador. A memória RAM é considerada uma memória primária e volátil, o que significa que os dados que são armazenados nela, apenas ficam lá temporariamente para que sejam feitas essas tarefas pelo processador.

“A memória RAM é muito mais rápida, e é necessário que os programas e dados sejam copiados para ela para que o processador possa acessá-los” (VASCONCELOS, 2007, p. 4).

Na Startup, a capacidade da memória RAM não precisará ser tão grande, pois as tarefas feitas na empresa, incluindo a utilização do *Software* de gestão, não serão muito exigentes e não ocuparão muito espaço, como exemplo disso, o próprio software de gestão da empresa ocupará apenas 8 MBs (*Megabytes*) de espaço na memória RAM. Além disso, a memória RAM também permite a execução de várias tarefas ao mesmo tempo e dependendo da capacidade da memória RAM é possível fazer isso com mais velocidade e com maior desempenho. Pensando nisso, foi decidido que na Quinotek, os dispositivos serão equipados com memórias RAM de 4 GBs, pois será o suficiente para ser possível a utilização das multitarefas mais simples que compõem a rotina na empresa. A figura 10 é um exemplo de como são Memórias RAM.

Figura 10 - A Memória RAM



Fonte: SOS, 2019.

6.4 O Desempenho do Software de Gestão

O processador é a unidade central de processamento (CPU), o cérebro do computador, que vai interagir e fazer as conexões necessárias entre todos os programas do computador, interpretando as informações que foram enviadas por eles.

“O processador é o responsável por executar as instruções que formam os programas. Quanto mais rápido o processador executar essas instruções, mais rápida será a execução dos programas” (VASCONCELOS, 2007, p. 2).

Os computadores que vão usar o *Software* de gestão terão que ter no mínimo um processador Intel Core i3 para rodar ele, devido a não ser um *Software* que vai ter uma exigência de desempenho tão alto para ser executado.

A sua velocidade de resposta é bem rápida, assim que o usuário executar uma ação, em questões de milissegundos a ação é executada, devido ao processador trabalhando em conjunto com o SSD e a memória RAM anteriormente citados.

Durante a execução do *Software*, o consumo de bateria será bem baixo, garantindo um maior tempo de execução do programa.

O *Software* vai garantir a proteção dos dados e evitar que os dados vazem, além de mostrar para o usuário quais dados serão coletados durante o uso, estando de acordo com as Lei Geral de Proteção de Dados (LGPD).

A figura 11 retrata um Processador da marca Intel.

Figura 11 - Processador Intel Core i3



Fonte: Dell, 2022.

7 INFRAESTRUTURA DE REDES

A Startup Quinotek assim como qualquer outra empresa atualmente, precisa de uma boa infraestrutura de redes computacionais para poder se comunicar com outros serviços e ferramentas diversas disponíveis com o universo da internet. Serviços esses sendo o próprio acesso à navegadores de pesquisa ou até ferramentas da Empresa Google por exemplo.

7.1 O que são Redes Computacionais

Redes de computadores são estruturas responsáveis por permitir a comunicação e compartilhamento de dados entre diversos dispositivos diferentes, isso é possível devido a utilização de uma estrutura de rede, que é feita a partir de conexões, estações de rede, roteamentos e protocolos de segurança que são incluídos em cada etapa e nível desse processo.

“Os atacantes podem tentar colocar *worms* nos hospedeiros na rede, adquirir segredos corporativos, mapear as configurações internas da rede e lançar ataques de DoS” (KUROSE; ROSS, 2015, p. 496).

Redes podem ser compostas e distribuídas por meios físicos (cabos) e por meios sem fio também, sendo esses meios Placas de rede, Roteadores, Cabos de par trançado, switches etc. Esses meios são mais comumente utilizados, existem também outros para cada tipo de rede (PESSANHA, 2022).

7.2 Tipos de Redes

Existem diversos tipos de redes de computadores para diversas finalidades, com cada um sendo executado da forma mais eficiente e específica, justificando um uso de um tipo de rede. Esses tipos de rede se diferenciam por diversos aspectos, sendo eles, área de conectividade, tipo de manutenção, taxas de transmissão e quantidade de conexões simultâneas, distância do alcance da rede, topologias, e suas infraestruturas, sendo elas físicas ou por outros meios sem fio.

7.2.1 Rede WLAN

A rede WLAN (*Wireless Local Area Network*) ou Rede de Área local Sem Fio que é utilizada em locais de pequeno e médio porte, como residências e pequenas empresas, e por isso, possui uma taxa de transmissão de dados alta de até 2.4 GBs/s. Ela possui conectividade *wireless*, ou seja, para se conectar a ela, são utilizados sinais de rádio transmitidos por um roteador, logo, uma conexão Wi-Fi (*Wireless Fidelity*). A manutenção dessa rede mais cara, assim como sua instalação (ROMVIEL, 2021).

7.2.2 Rede LAN

A rede LAN (Local Area Network) ou Rede de área local, é uma rede de curto alcance que serve para conectar dispositivos em locais como pequenos escritórios e residências. A rede LAN tem como conectividade o meio físico por intermédio de cabos de Ethernet que são conectados de um dispositivo para um roteador. Além disso, em comparação com a rede WLAN ela é mais segura, tanto em questões de invasões e ataques virtuais, quanto em questões externas como o ambiente e a qualidade dos cabos utilizados, um ponto negativo é que essa rede acabou perdendo popularidade para a rede WLAN com o passar do tempo (ROMVIEL, 2021).

As LANs com fios utilizam uma série de tecnologias de transmissão diferentes. A maioria delas usa fios de cobre, mas algumas usam fibra óptica. As LANs são restritas em tamanho, o que significa que o tempo de transmissão, no pior caso, é limitado e conhecido com antecedência. Conhecer esses limites ajuda na tarefa de projetar protocolos de rede (TANENBAUM; WETHERALL, 2011, p. 12).

Um exemplo do funcionamento da rede LAN pode ser observado na figura 12.

Figura 12 - Exemplo de Funcionamento da rede LAN em topologia Estrela



Fonte: Freepik, 2019.

7.3 Topologias de Rede

Topologias de redes são a organização em que um tipo de rede opera, como o compartilhamento de uma estrutura e a conexão entre dispositivos conectados em um tipo de rede. Essa estrutura é muito importante ser levada em conta na hora de organizar um plano de rede, pois irá afetar na qualidade de transmissão de dados e na estabilidade das conexões, além disso dependendo da topologia escolhida, a manutenção e instalação de uma rede em um local pode mudar (NOLETO, 2022).

7.3.1 Topologia de rede Estrela

Existem diversos tipos de topologias diferentes, com variados meios para conectar dispositivos em uma rede, porém, a topologia de rede Estrela é a ideal para uma conexão central e segura em um escritório de uma pequena empresa, como a Startup Quinotek.

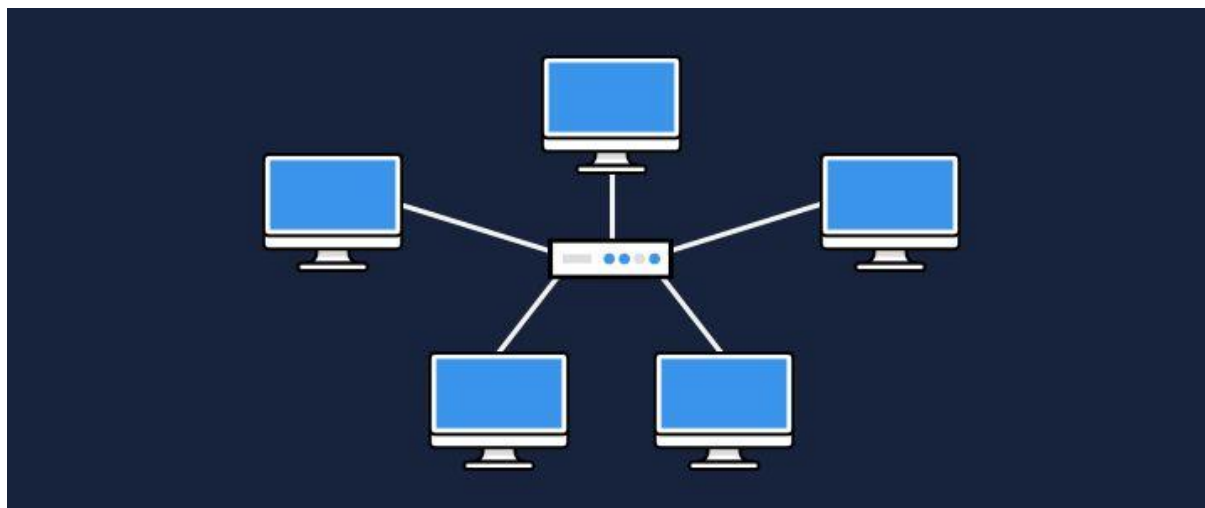
A topologia Estrela é caracterizada por ter conexões entre computadores que se conectam a um ponto central (*Switch*), e a instalação da rede é feita separadamente para cada nó de rede em cada dispositivo conectado nessa topologia, fazendo com que, se feita uma alteração em um dos dispositivos, essa alteração não irá afetar os outros dispositivos conectados (RESENDE, 2022).

“Sendo assim, *Ethernet* com uma topologia de estrela baseada em um *hub* também é uma LAN de difusão sempre que um *hub* recebe um bit de uma de suas

interfaces, ele envia uma cópia para todas as outras interfaces” (KUROSE; ROSS, 2015, p. 348).

Um exemplo de uma Topologia Estrela pode ser observado na figura 13.

Figura 13 - Exemplo do funcionamento da Topologia Estrela



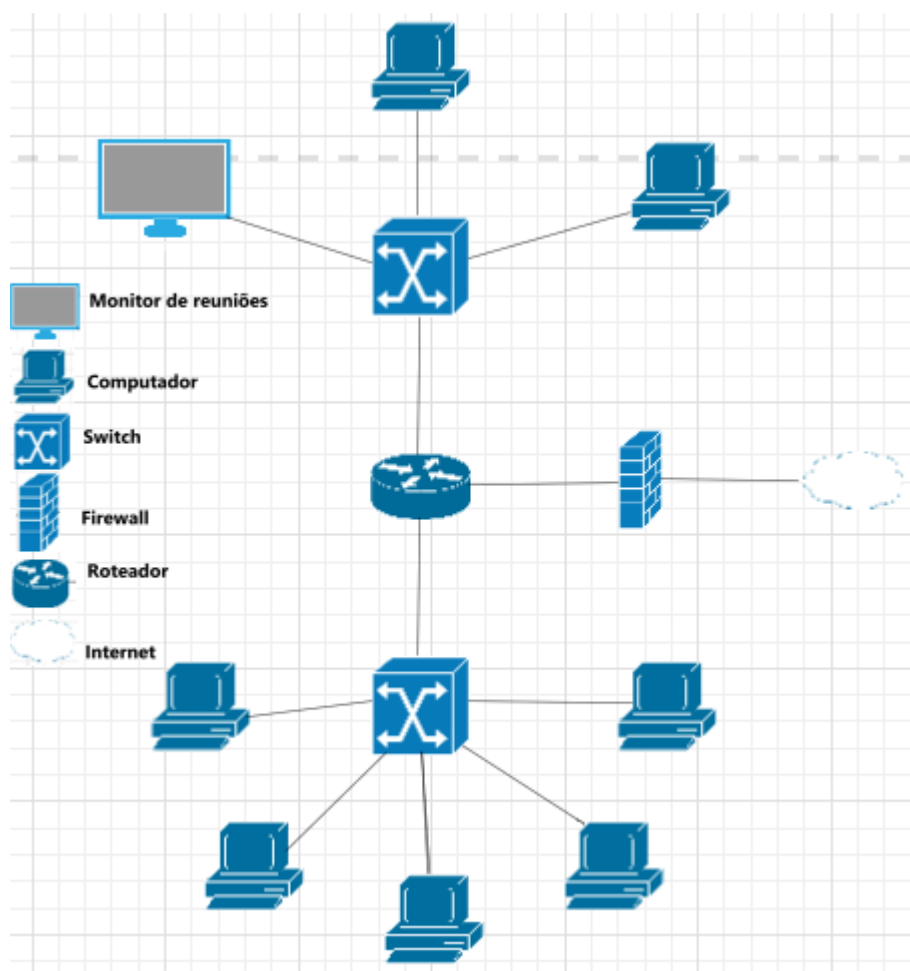
Fonte: Gerenciatec, 2022.

7.4 A Infraestrutura de Rede utilizada na Quinotek

Foram observados dois tipos de redes e uma topologia de rede, sendo as redes LAN e WLAN e a Topologia Estrela, e devido ao espaço, segurança, custos e a arquitetura de topologia das redes explicados anteriormente, constata-se que a rede LAN em Topologia Estrela se encaixa melhor no contexto da Quinotek.

Foi desenvolvido um diagrama de redes para demonstrar de forma mais clara como funcionará a topologia da rede estrela na rede LAN no espaço da empresa, mostrando a conexão dos elementos, isso pode ser observado na figura 14.

Figura 14 - Diagrama de redes da Quinotek



Fonte: Autoria própria, 2022.

8 SEGURANÇA DE DADOS, LGPD E CRIPTOGRAFIA

Em todo bom sistema existe camadas de segurança para manter dados e informações importantes longe de mãos erradas, e para ter essa segurança de dados, existem diversas maneiras e inovações, que contam com regras estritas em suas etapas de desenvolvimento e implementação.

Em *Softwares* a segurança de dados pode ser implementada para evitar vazamentos de informações na internet, se for um serviço web ou com conectividade com a rede, e/ou para proteger as informações contidas no código fonte, em casos de um software que não possui seus serviços *Open source*.

Além disso, a segurança de um *Software* é determinada em funções como o login e cadastro de um usuário, já que protege os usuários de terem suas informações acessadas por um terceiro, e com uma boa regra de senha e nome de perfil, pode tornar uma conta de um usuário em um aplicativo muito mais segura.

8.1 LGPD e proteção de dados

A LGPD (Lei Geral de Proteção de Dados) é uma lei brasileira que foi aprovada em Agosto de 2018, mas só entrou em vigor em Setembro de 2020.

“Inspirada na lei europeia de proteção de dados, conhecida como General Data Protection Regulation” (GARCIA; FERNANDES; GONÇALVES; BARRETTO, 2020, p. 12).

A LGPD começou aqui no Brasil por questões de segurança do usuário no meio digital, para poder limitar a quantidade de dados que empresas diversas espalhadas pela web ou por meio de conexões periféricas (internet das coisas) possam adquirir dos dispositivos das pessoas, e caso as regras e códigos descritos na LGPD não forem respeitados, existem penalidades para punir quem for que esteja descumprindo as leis de proteção de dados (SCHULTZ, 2019).

8.1.1 Como funciona a LGPD

Empresas no meio digital hoje em dia no Brasil, tem a obrigação de estabelecer com total transparência e visibilidade do usuário, quais tipos de dados elas irão utilizar de seus dispositivos, seja em um *Software* ou em serviços *online*, ou seja, pedindo a permissão do usuário para poder utilizar os dados dele. Desta maneira então dando importância as responsabilidades de ambas as partes em um acordo de compartilhamento de dados. Assim como também tornar possível a possibilidade de alteração ou devolução de dados do usuário pela empresa (SCHULTZ, 2019).

Art. 1º Esta Lei dispõe sobre o tratamento de dados pessoais, inclusive nos meios digitais, por pessoa natural ou por pessoa jurídica de direito público ou privado, com o objetivo de proteger os direitos fundamentais de liberdade e de privacidade e o livre desenvolvimento da personalidade da pessoa natural (BRASIL, 2018).

Hoje em dia, na chamada era da informação, as principais estratégias de *marketing* estão sendo utilizadas principalmente por serviços digitais com o compartilhamento constante de informações de pessoas para outras, e de pessoas para empresas, principalmente em redes sociais e aplicativos de vendas *online* ou em sites. Isso também acarreta a necessidade de deixar claro o porquê e como os dados do usuário serão utilizados de maneira clara e objetiva, sem nenhuma forma de induzir o usuário a concordar com tais termos, de acordo com a legislação.

8.1.2 Tipos de dados protegidos pela LGPD

Dados pessoais como nome, endereço residencial, endereço de IP, E-mail, senha, números de telefone etc. Também incluem dados sensíveis mais reveladores como sexualidade, religiosidade, posicionamentos políticos e dados bancários (SCHULTZ, 2019).

“Art. 5º Para os fins desta Lei, considera-se:

I - Dado pessoal: informação relacionada a pessoa natural identificada ou identificável;” (BRASIL, 2018).

8.1.3 A LGPD no Software de Gestão da Quinotek

No Software de gestão da Startup Quinotek será incluída as leis de proteção de dados da LGPD, que atuarão na privacidade e segurança tanto das informações dos clientes da empresa, como também protegendo as informações dos colaboradores que trabalham com ela. A figura 15 ilustra a LGPD no Brasil.

Figura 15 - Imagem da LGPD no Brasil



Fonte: Dome, 2021.

8.2 Criptografia de dados

A Criptografia de dados tem o objetivo de eliminar as chances de terceiros obterem acesso a dados, é a prática de codificar e decodificar dados, a fim de embaralhar as informações armazenadas a fim de que só aqueles que obtém uma chave para descriptografar os dados tenham acesso a ela. Após os dados serem criptografados, se aplica um algoritmo para codificá-los de modo que não se tenha mais o seu formato original, logo, não possam ser lidos, e os dados só podem ser decodificados ao formato original usando uma chave de descriptografia específica. (JANUARIO, 2021).

Entre eles, os militares tiveram o papel mais importante e definiram as bases para a tecnologia. Nas organizações militares, as mensagens a ser criptografadas eram entregues habitualmente a auxiliares mal remunerados, que se encarregavam de criptografá-las e transmiti-las (TANENBAUM; WETHERALL, 2011, p. 481).

8.2.1 Tipos de Criptografia

Entre os tipos de Criptografia temos a Criptografia Simétrica, que usa uma única chave para cifrar e decifrar a mensagem, sendo o seu segredo compartilhado.

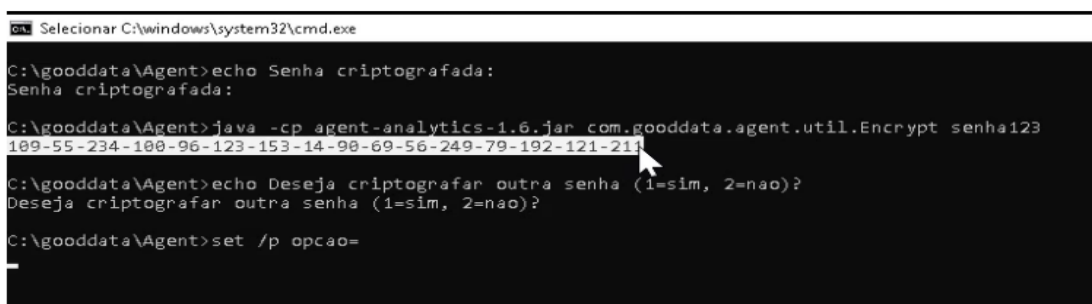
Tem também a Criptografia Assimétrica que utiliza um par de chaves: uma chave pública e outra privada que se relacionam, só será descriptografado apenas nessa condição (JANUARIO, 2021).

8.2.2 A Criptografia no Software de gestão

O software que a Quinotek vai usar, estará com a criptografia dos dados, com o intuito de prevenir que pessoas mal-intencionadas tenham acesso as informações dos usuários, e sempre garantindo a proteção e segurança desses dados sensíveis.

A Figura 16 exemplifica uma Criptografia de Senha.

Figura 16 - Exemplo de Criptografia de Senha

A screenshot of a Windows command prompt window. The title bar reads "Selecionar C:\windows\system32\cmd.exe". The command prompt shows the following sequence of commands and outputs:
1. Command: `C:\gooddata\Agent>echo Senha criptografada:`
Output: `Senha criptografada:`
2. Command: `C:\gooddata\Agent>java -cp agent-analytics-1.6.jar com.gooddata.agent.util.Encrypt senha123`
Output: `109-55-234-100-96-123-153-14-90-69-56-249-79-192-121-211`
3. Command: `C:\gooddata\Agent>echo Deseja criptografar outra senha (1=sim, 2=nao)?`
Output: `Deseja criptografar outra senha (1=sim, 2=nao)?`
4. Command: `C:\gooddata\Agent>set /p opcao=`
Output: A cursor is visible on the next line, indicating the prompt is waiting for input.

Fonte: TOTVS, 2021.

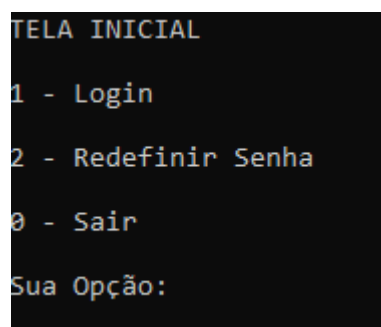
9 MANUAL DO USUÁRIO

O Manual do usuário serve para mostrar um guia de instruções sobre como os gerentes da Quinotek irão utilizar o *Software* de Gestão.

9.1 Tela Inicial

A primeira tela aberta pelo programa, nessa tela o usuário pode escolher uma de três opções de como prosseguir no programa. É possível observar essa tela na figura 17.

Figura 17 - Tela Inicial



Fonte: Autoria própria, 2022.

9.1.1 Tela de Login

A tela onde o usuário irá fornecer as informações necessárias, de usuário e senha, para assim poder fazer o seu Login e entrar no menu principal. O Login vai funcionar para os 2 gerentes da Quinotek, que serão os únicos a utilizar o sistema e terão um cadastro pré-definido. Se errar o login, irá aparecer uma mensagem de erro e irá retornar para a tela inicial. Se o login for bem-sucedido entrará no menu principal.

A figura 18 e 19 mostra como essa tela será visualizada pelo usuário.

Figura 18 - Tela de Login

```
LOGIN  
  
Usuário: paulo nogueira  
Senha: thrat349
```

Fonte: Autoria própria, 2022.

Figura 19 - Login feito com sucesso

```
Bem-vindo paulo nogueira!  
Pressione ENTER Para Continuar!
```

Fonte: Autoria própria, 2022.

9.1.2 Tela de Redefinição de Senha

Nesta tela, o usuário irá fornecer o seu nome e seu CPF, e ao validar essas informações, poderá escolher uma nova senha e depois confirmar a senha escolhida, e se as novas senhas digitadas forem iguais, o programa irá cadastrar a nova senha informada no usuário desejado. Após isso, será retornado à tela inicial para executar outra ação. Se errar o nome e o CPF no início, voltará para a tela inicial. Essa tela pode ser visualizada na figura 20.

Figura 20 - Tela de Redefinição de Senha

```
REDEFINIR SENHA  
  
Informe seu Usuário: paulo nogueira  
Informe seu CPF: 32165498755  
  
Digite sua Nova Senha:
```

Fonte: Autoria própria, 2022.

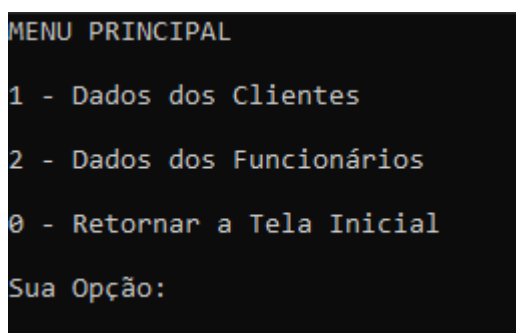
9.1.3 Sair do Programa

Ao escolher essa opção, o programa irá ser encerrado.

9.2 Tela de Menu Principal

Após confirmado que o usuário e a senha fornecidos na tela de login são compatíveis com os cadastrados no programa, o usuário será redirecionado para esta tela, onde o usuário poderá escolher uma de três ações para executar. A tela pode ser observada na figura 21.

Figura 21 - Menu Principal



Fonte: Autoria própria, 2022.

9.2.1 Tela de Dados dos Clientes

Na tela de dados dos clientes, os administradores terão a opção de cadastrar um novo cliente que usufrui dos serviços da Quinotek, editar as informações, visualizar cadastros, excluir cadastros, gerar relatórios em formato de arquivo TXT e voltar para o menu principal. Essa tela de informações está representada na figura 22.

Figura 22 - Tela de Dados dos Clientes

```
DADOS DOS CLIENTES  
  
1 - Cadastrar Clientes  
2 - Editar Clientes  
3 - Exibir Clientes  
4 - Excluir Clientes  
0 - Retornar ao Menu Principal  
Sua Opção:
```

Fonte: Autoria própria, 2022.

9.2.2 Tela de Dados dos Funcionários

Na tela de dados dos funcionários, os administradores terão a opção de cadastrar os funcionários da Quinotek, editar as informações, visualizar cadastros, excluir cadastros, gerar relatórios em formato de arquivo TXT e voltar para o menu principal, como mostra a figura 23.

Figura 23 - Tela de dados dos Funcionários

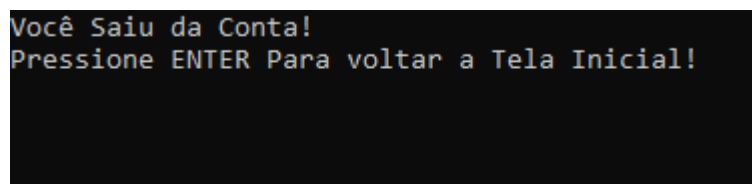
```
DADOS DOS FUNCIONÁRIOS  
  
1 - Cadastrar Funcionários  
2 - Editar Funcionários  
3 - Exibir Funcionários  
4 - Excluir Funcionários  
0 - Retornar ao Menu Principal  
Sua Opção:
```

Fonte: Autoria própria, 2022.

9.2.3 Voltar à Tela Inicial

Ao escolher essa opção o usuário irá sair do login e será redirecionado à tela inicial. Isso pode ser observado na figura 24.

Figura 24 - Sair da conta



Fonte: Autoria própria, 2022.

9.3 Tela de Dados dos Clientes

Nesta tela, o usuário poderá escolher uma entre seis ações relacionadas às informações dos clientes.

9.3.1 Visualizar Informações dos Clientes

Nesta tela o usuário poderá visualizar todas as informações de todos os clientes cadastrados, como: Nome da escola, CNPJ da instituição de ensino, e-mail, telefone, estado, cidade, bairro, rua e número, data de vencimento da fatura do mês e se o cliente já pagou ou não a fatura. Após visualizar, poderá voltar a tela de dados dos clientes. Essa tela de informações pode ser visualizada na figura 25.

Figura 25 - Exibição de Dados dos Clientes

```
EXIBIR CLIENTES

Pressione ENTER para voltar!

-----
Nome da Instituição: Escola Tereza Monteiro
CNPJ: 87598536836875
Email: terezamonteiro@gmail.com
Telefone: 32648484
Estado: SP
Cidade: Tatui
Bairro: Jardim Nina Gomes
Rua: Leopoldo Soares
Número: 4444
Data de vencimento da última fatura: 10/12/2022
Situação da fatura: nao foi pago
-----
```

Fonte: Autoria própria, 2022.

9.3.2 Editar Informações dos Clientes

Nesta tela o usuário poderá editar todas as informações de todos os clientes cadastrados, como: Nome da escola, CNPJ da instituição de ensino, e-mail, telefone, estado, cidade, bairro, rua e número, data de vencimento da fatura do mês e se o cliente já pagou ou não a fatura, utilizando o CNPJ dos clientes para selecionar o usuário a ser editado e depois confirmando sua decisão. Após editar as informações irá retornar a tela de dados dos clientes. Se errar o CNPJ, voltará para a tela de dados dos clientes. Isso pode ser visto na figura 26.

Figura 26 - Tela de edição de informações dos Clientes

```
EDITAR CLIENTES

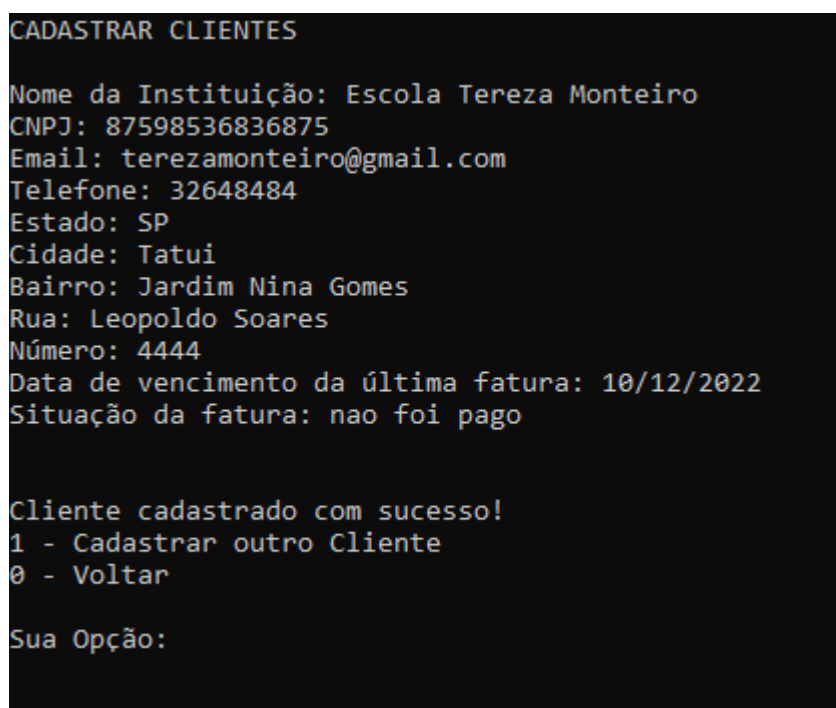
Digite o CNPJ da Instituição que deseja editar
CNPJ: 87598536836875
```

Fonte: Autoria própria, 2022.

9.3.3 Cadastrar Clientes

Nesta tela o usuário poderá cadastrar todas as informações necessárias para inserir um novo cliente que utiliza dos serviços da Quinotek: Nome da escola, CNPJ da instituição de ensino, e-mail, telefone, estado, cidade, bairro, rua e número, data de vencimento da fatura do mês e se o cliente já pagou ou não a fatura. Após cadastrar um cliente, irá perguntar se quer cadastrar mais um, se sim cadastra mais clientes, se não volta a tela de dados dos clientes. É possível observar essa tela na figura 27.

Figura 27 - Tela de cadastro de Clientes



```
CADASTRAR CLIENTES

Nome da Instituição: Escola Tereza Monteiro
CNPJ: 87598536836875
Email: terezamonteiro@gmail.com
Telefone: 32648484
Estado: SP
Cidade: Tatui
Bairro: Jardim Nina Gomes
Rua: Leopoldo Soares
Número: 4444
Data de vencimento da última fatura: 10/12/2022
Situação da fatura: nao foi pago

Cliente cadastrado com sucesso!
1 - Cadastrar outro Cliente
0 - Voltar

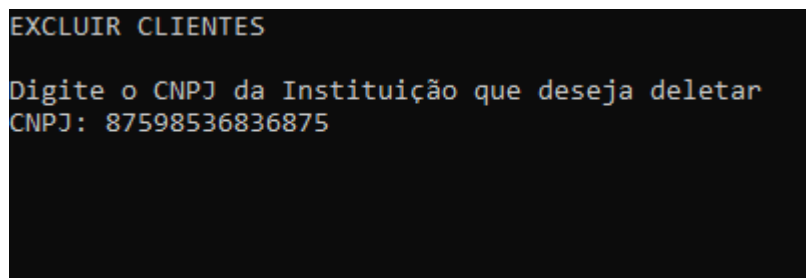
Sua Opção:
```

Fonte: Autoria própria, 2022.

9.3.4 Excluir Cadastros de Clientes

Nesta tela o usuário poderá excluir o cadastro dos clientes inserindo o CNPJ do cadastro a ser excluído e depois confirmando sua decisão. Após excluir o cliente, irá retornar a tela de dados dos clientes. Se errar o CNPJ, voltará para a tela de dados dos clientes. Um exemplo disso pode ser visto na figura 28.

Figura 28 - Tela de exclusão de informações de Clientes



Fonte: Autoria própria, 2022.

9.3.5 Gerar relatório em TXT

Ao escolher essa opção, o programa irá pedir ao usuário para confirmar sua decisão. Caso o usuário confirme, o programa irá gerar um arquivo TXT contendo todas as informações de todos os clientes e funcionários cadastrados, depois irá retornar à tela de dados dos clientes. Caso o usuário não confirme a sua decisão ele irá retornar à tela de dados dos clientes.

9.3.6 Voltar ao Menu Principal

Ao escolher essa opção o usuário será redirecionado à tela de menu principal, onde poderá escolher iniciar uma nova ação.

9.4 Tela de Dados dos Funcionários

Nesta tela, o usuário poderá escolher uma entre seis ações relacionadas às informações dos funcionários.

9.4.1 Visualizar Informações dos Funcionários

Nesta tela o usuário poderá visualizar todas as informações de todos os funcionários cadastrados, como: Nome, CPF, e-mail, telefone, estado, cidade, bairro, rua, número, complemento, cargo, data de admissão, datas de pagamento dos seus salários e os respectivos valores pagos. Após visualizar, poderá voltar a tela de dados dos funcionários. A Representação da tela pode ser vista na figura 29.

Figura 29 - Tela de exibição de informações dos Funcionários

```
Pressione ENTER para voltar!
-----
Nome: Roberval da Silva Costa
CPF: 76576456787
Email: robervalsc@gmail.com
Telefone: 991788687
Estado: SP
Cidade: Jundiai
Bairro: parque das esferas
Rua: Valerio pinheiros
Número: 888
Complemento: não tem
Cargo: Desenvolvedor Back-end
Data de Admissão: 10/01/2022
Salário: 3500,00
Data do Pagamento: 15/12/2022
-----
Nome: Claudia
CPF: 46478788566
Email: claudiamorgana@gmail.com
Telefone: 998746474
Estado: SP
Cidade: Sorocaba
Bairro: Jardim torres
Rua: Wesley Sareal
Número: 999
Complemento: não
Cargo: Recepcionista
Data de Admissão: 20/01/2022
Salário: 1400
Data do Pagamento: 15/12/2022
-----
```

Fonte: Autoria própria, 2022.

9.4.2 Editar Informações dos Funcionários

Nesta tela o usuário poderá editar todas as informações de todos os funcionários cadastrados, como: Nome, CPF, e-mail, telefone, estado, cidade, bairro, rua, número, complemento, cargo, data de admissão, datas de pagamento dos seus salários e os respectivos valores pagos, utilizando o CPF dos funcionários para selecionar o cadastro a ser editado e depois confirmando sua decisão. Após editar as informações irá retornar a tela de dados dos funcionários. Se errar o CPF, voltará para a tela de dados dos funcionários. A figura 30 demonstra essas informações.

Figura 30 - Tela para editar informações dos Funcionários

```

EDITAR FUNCIONÁRIOS

Digite o CPF do Funcionário para editar seus dados
CPF: 76576456787

```

Fonte: Autoria própria, 2022.

9.4.3 Cadastrar Funcionários

Nesta tela o usuário poderá cadastrar todas as informações necessárias para inserir um novo funcionário no programa, como: Nome, CPF, e-mail, telefone, estado, cidade, bairro, rua, número, complemento, cargo, data de admissão, datas de pagamento dos seus salários e os respectivos valores pagos. Após cadastrar um funcionário, irá perguntar se quer cadastrar mais um, se sim cadastra mais funcionários, se não volta a tela de dados dos funcionários. Isso pode ser visto na figura 31.

Figura 31 - Tela de cadastro de Funcionários

```

CADASTRAR FUNCIONÁRIOS

Nome: Claudia
CPF: 46478788566
Email: claudiamorgana@gmail.com
Telefone: 998746474
Estado: SP
Cidade: Sorocaba
Bairro: Jardim torres
Rua: Wesley Sareal
Número: 999
Complemento: não
Cargo: Recepcionista
Data de Admissão: 20/01/2022
Salário: 1400
Data do Pagamento: 15/12/2022

Funcionário cadastrado com sucesso!
1 - Cadastrar outro Funcionário
0 - Voltar

Sua Opção:

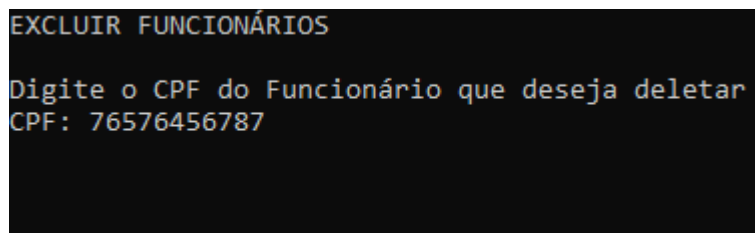
```

Fonte: Autoria própria, 2022.

9.4.4 Excluir Cadastros de Funcionários

Nesta tela o usuário poderá excluir o cadastro dos funcionários inserindo o CPF do cadastro a ser excluído e depois confirmando sua decisão. Após excluir o funcionário, irá retornar a tela de dados dos funcionários. Se errar o CPF, voltará para a tela de dados dos funcionários. A figura 32 demonstra isso.

Figura 32 - Tela de exclusão de cadastro de Funcionários



Fonte: Autoria própria, 2022.

9.4.5 Gerar relatório em TXT

Ao escolher essa opção, o programa irá pedir ao usuário para confirmar sua decisão. Caso o usuário confirme, o programa irá gerar um arquivo TXT contendo todas as informações de todos os funcionários cadastrados, depois irá retornar à tela de dados dos funcionários. Caso o usuário não confirme a sua decisão ele irá retornar à tela de dados dos funcionários.

9.4.6 Voltar ao Menu Principal

Ao escolher essa opção o usuário será redirecionado à tela de menu principal, onde poderá escolher iniciar uma nova ação.

10 O CÓDIGO EM C E SUAS FUNCIONALIDADES

Para que seja possível o entendimento da lógica utilizada para se fazer o código, é indispensável que sejam escritos comentários ao percorrer do algoritmo, explicando cada funcionalidade que existe nele, as figuras abaixo mostram respectivamente partes do código que foram comentadas em frente a cada funcionalidade.

“A Lógica organiza de forma metódica o pensar de quem se preocupa com a atividade do raciocínio” (SOFFNER, 2013, p. 16).

10.1 Imagens dos Comentários no Código

O código está separado por funções, ou seja, etapas que lhe organizam de maneira mais dinâmica e eficiente.

“Em C, cada função é um bloco discreto de código. um código de uma função é privativo àquela função e não pode ser acessado por nenhum comando em uma outra função, exceto por meio de uma chamada à função” (SCHILDT, 1997, p. 139).

As figuras 33 á 49 não mostram o código inteiro, somente as partes em que existem comentários.

Figura 33 - “Define” e “Struct”

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <locale.h>
//#define linha 2" Serve para definir quantos cadastros podem ser feitos na estrutura administrador.
#define linha 2
/*#define cad_fun 1000" Serve para definir a quantidade de cadastros a serem realizados nas
estruturas cad_funcionario e cad_cliente.*/
#define cad_fun 1000

int op;
static int fun_li=0;
static int cli_li=0;

void tela_init();
void login();
void red_senha();
void tela_ad();
void funfas();
void funfas_cad();
void exi_fun();
void red_fun();
void del_fun();
void cliente();
void cad_cli();
void exi_cli();
void red_cli();
void del_cli();
//Struct cria um tipo de grupo de dados a serem guardados em uma unica variavel.
struct administrador{
    char user[linha][16];
```

Fonte: Autoria própria, 2022.

Figura 34 - "Strcpy", "For" e "If"

```

};
struct cad_cliente cli;

int main(){
    setlocale(LC_ALL, "portuguese");
    int i;
    //strcpy é um comando usado para inserir dados dentro de uma variável do tipo char.
    strcpy(ad.user[0], "paulo nogueira");
    strcpy(ad.sen[0], "thrat349");
    strcpy(ad.user[1], "david gomes");
    strcpy(ad.sen[1], "thrat747");
    ad.cpf[0]=32165498755;
    ad.cpf[1]=55789456123;
    tela_init();
    return 0;
}

void del_cli(){
    int del;
    int i;
    char rec_cnpj[10];
    printf("EXCLUIR CLIENTES\n\n");
    printf("Digite o CNPJ da Instituição que deseja deletar\n");
    printf("CNPJ: ");
    gets(rec_cnpj);
    //for nesse caso esta sendo utilizado para apagar o cadastro de um cliente.
    for(i=0;i<cad_fun;i++){
        //if nesse caso esta sendo usado para validar a existência de um cadastro através do CNPJ inserido anteriormente.
        if(strcmp(rec_cnpj, cli.cnpj[i])==0){
            system("cls");
            printf("Instituição: %s\n\n", cli.nome[i]);

```

Fonte: Autoria própria, 2022.

Figura 35 - "if"

```

    setbuf(stdin, NULL);
    //esse if serve para o usuário ter certeza que quer deletar o cadastro do cliente.
    if(del==1){
        strcpy(cli.nome[i], "");
        strcpy(cli.cnpj[i], "");
        strcpy(cli.email[i], "");
        strcpy(cli.telefone[i], "");
        strcpy(cli.estado[i], "");
        strcpy(cli.cidade[i], "");
        strcpy(cli.bairro[i], "");
        strcpy(cli.rua[i], "");
        strcpy(cli.numero[i], "");
        strcpy(cli.data[i], "");
        strcpy(cli.fat[i], "");
        printf("\nCadastro deletado com Sucesso!\n");
        getchar();
        cliente();
    }
    else{
        cliente();
    }
}
system("cls");
printf("Usuário não Encontrado!");
getchar();
cliente();
}

void red_cli(){
    int val;

```

Fonte: Autoria própria, 2022.

Figura 36 - “for” e “if”

```

};

void red_cli() {
    int val;
    int i;
    char rec_cnpj[10];
    printf("EDITAR CLIENTES\n\n");
    printf("Digite o CNPJ da Instituição que deseja editar\n");
    printf("CNPJ: ");
    gets(rec_cnpj);
    //for nesse caso esta sendo utilizado para editar o cadastro de um cliente.
    for(i=0;i<cad_fun;i++){
        //if nesse caso esta sendo usado para validar a existência de um cadastro através do CNPJ inserido anteriormente.
        if(strcmp(rec_cnpj, cli.cnpj[i])==0){
            system("cls");
            printf("Usuário: %s\n\n - Editar Informações\n\n0 - Retornar\n\nSua Opção: ", cli.nome[i]);
            scanf("%d", &val);
            setbuf(stdin, NULL);
            //esse if serve para o usuário ter certeza que quer editar o cadastro do Cliente.
            if(val==1){
                system("cls");
                printf("Nome da Instituição: ");
                gets(cli.nome[i]);
                printf("CNPJ: ");
                gets(cli.cnpj[i]);
                printf("Email: ");
                gets(cli.email[i]);
                printf("Telefone: ");
                gets(cli.telefone[i]);
                printf("Estado: ");
                gets(cli.estado[i]);
            }
        }
    }
}

```

Fonte: Autoria própria, 2022.

Figura 37 - “for” e “if”

```

void exi_cli() {
    int i;
    printf("EXIBIR CLIENTES\n\n");
    printf("Pressione ENTER para voltar!\n\n");
    printf("-----\n");
    //Esse for está sendo usado para exibir todos os clientes cadastrados.
    for(i=0;i<cad_fun;i++){
        //esse if serve para validar a existência dos clientes para exibi-los.
        if(strlen(cli.nome[i])>0){
            printf("Nome da Instituição: %s\n", cli.nome[i]);
            printf("CNPJ: %s\n", cli.cnpj[i]);
            printf("Email: %s\n", cli.email[i]);
            printf("Telefone: %s\n", cli.telefone[i]);
            printf("Estado: %s\n", cli.estado[i]);
            printf("Cidade: %s\n", cli.cidade[i]);
            printf("Bairro: %s\n", cli.bairro[i]);
            printf("Rua: %s\n", cli.rua[i]);
            printf("Número: %s\n", cli.numero[i]);
            printf("Data de vencimento da última fatura: %s\n", cli.data[i]);
            printf("Situação da fatura: %s\n", cli.fat[i]);
            printf("-----\n");
        }
    }
    getchar();
    cliente();
}

void cad_cli() {
    int i;
    char cnpj[16];
}

```

Fonte: Autoria própria, 2022.

Figura 38 - “do while” e “for”

```

void cad_cli() {
    int i;
    char cnpj[16];
    /*do while serve para repetir uma ação até o momento em que atingir a condição imposta no while.
    nesse caso serve para repetir o cadastro até o usuário digitar "0".*/
    do{
        system("cls");
        printf("CADASTRAR CLIENTES\n\n");
        printf("Nome da Instituição: ");
        gets(cli.nome[cli_li]);
        printf("CNPJ: ");
        gets(cnpj);
        //Esse for serve para verificar a existência de algum cadastro com o CNPJ inserido anteriormente.
        for(i=0;i<cad_fun;i++){
            while(strcmp(cnpj, cli.cnpj[i])==0){
                printf("\nCNPJ já Cadastrado!\n");
                strcpy(cli.nome[cli_li], "");
                getchar();
                cliente();
            }
        }
        strcpy(cli.cnpj[cli_li], cnpj);
        printf("Email: ");
        gets(cli.email[cli_li]);
        printf("Telefone: ");
        gets(cli.telefone[cli_li]);
        printf("Estado: ");
        gets(cli.estado[cli_li]);
        printf("Cidade: ");
        gets(cli.cidade[cli_li]);
    }
}

```

Fonte: Autoria própria, 2022.

Figura 39 - “do while” e “switch case”

```

void cliente() {
    //esse do while serve para repetir a tela dos clientes enquanto o usuário não digitar 0.
    do{
        system("cls");
        printf("DADOS DOS CLIENTES\n\n");
        printf("1 - Cadastrar Clientes\n\n");
        printf("2 - Editar Clientes\n\n");
        printf("3 - Exibir Clientes\n\n");
        printf("4 - Excluir Clientes\n\n");
        printf("0 - Retornar ao Menu Principal\n\n");
        printf("Sua Opção: ");
        scanf("%d", &op);
        setbuf(stdin, NULL);
        //switch case serve para reduzir a quantidade de ifs e elses, muito utilizado em menus como esse.
        switch(op){
            case 1:
                system("cls");
                setbuf(stdin, NULL);
                cad_cli();
                break;
            case 2:
                system("cls");
                setbuf(stdin, NULL);
                red_cli();
                break;
            case 3:
                system("cls");
                setbuf(stdin, NULL);
                exi_cli();
                break;
        }
    }
}

```

Fonte: Autoria própria, 2022.

Figura 40 - "for" e "if"

```

void del_fun(){
    int del;
    int i;
    char rec_cpf[10];
    printf("EXCLUIR FUNCIONÁRIOS\n\n");
    printf("Digite o CPF do Funcionário que deseja deletar\n");
    printf("CPF: ");
    gets(rec_cpf);
    //esse for serve para apagar o funcionário cadastrado.
    for(i=0;i<cad_fun;i++){
        //esse if serve para validar a existência de um cadastro que tenha o CPF digitado.
        if(strcmp(rec_cpf, fn.cpf[i])==0){
            system("cls");
            printf("-----");
            printf("Funcionário: %s\n\n", fn.nome[i]);
            printf("1 - Deletar Cadastro\n\n0 - Voltar\n\nSua Opção: ");
            scanf("%d", &del);
            setbuf(stdin, NULL);
            //Esse if serve para validar a escolha do usuário.
            if(del==1){
                strcpy(fn.nome[i], "");
                strcpy(fn.cpf[i], "");
                strcpy(fn.email[i], "");
                strcpy(fn.telefone[i], "");
                strcpy(fn.estado[i], "");
                strcpy(fn.cidade[i], "");
                strcpy(fn.bairro[i], "");
                strcpy(fn.rua[i], "");
                strcpy(fn.numero[i], "");
                strcpy(fn.complemento[i], "");
            }
        }
    }
}

```

Fonte: Autoria própria, 2022.

Figura 41 - "for" e "if"

```

void red_fun(){
    int val;
    int i;
    char rec_cpf[10];
    printf("EDITAR FUNCIONÁRIOS\n\n");
    printf("Digite o CPF do Funcionário para editar seus dados\n");
    printf("CPF: ");
    gets(rec_cpf);
    //esse for serve para editar o cadastro de algum funcionário.
    for(i=0;i<cad_fun;i++){
        //esse if serve para validar se existe algum cadastro que contenha o CPF digitado.
        if(strcmp(rec_cpf, fn.cpf[i])==0){
            system("cls");
            printf("Usuário: %s\n\n1 - Editar Informações\n\n0 - Retornar\n\nSua Opção: ", fn.nome[i]);
            scanf("%d", &val);
            setbuf(stdin, NULL);
            //esse if serve para verificar a decisão do usuário para editar o funcionário.
            if(val==1){
                system("cls");
                printf("Nome: ");
                gets(fn.nome[i]);
                printf("CPF: ");
                gets(fn.cpf[i]);
                printf("Email: ");
                gets(fn.email[i]);
                printf("Telefone: ");
                gets(fn.telefone[i]);
                printf("Estado: ");
                gets(fn.estado[i]);
            }
        }
    }
}

```

Fonte: Autoria própria, 2022.

Figura 42 - “for” e “if”

```

void exi_fun() {
    int i;
    printf("EXIBIR FUNCIONÁRIOS\n\n");
    printf("Pressione ENTER para voltar!\n\n");
    printf("-----\n");
    //Esse for serve para exibir todos os cadastros existentes.
    for(i=0;i<cad_fun;i++){
        //Esse if serve para validar a existência dos cadastros dos funcionários nos vetores.
        if(strlen(fn.nome[i])>0){
            printf("Nome: %s\n", fn.nome[i]);
            printf("CPF: %s\n", fn.cpf[i]);
            printf("Email: %s\n", fn.email[i]);
            printf("Telefone: %s\n", fn.telefone[i]);
            printf("Estado: %s\n", fn.estado[i]);
            printf("Cidade: %s\n", fn.cidade[i]);
            printf("Bairro: %s\n", fn.bairro[i]);
            printf("Rua: %s\n", fn.rua[i]);
            printf("Número: %s\n", fn.numero[i]);
            printf("Complemento: %s\n", fn.complemento[i]);
            printf("Cargo: %s\n", fn.cargo[i]);
            printf("Data de Admissão: %s\n", fn.data[i]);
            printf("Salário: %s\n", fn.salario[i]);
            printf("Data do Pagamento: %s\n", fn.recebe[i]);
            printf("-----\n");
        }
    }
    getchar();
    funfas();
}

```

Fonte: Autoria própria, 2022.

Figura 43 - “do”, “for” e “while”

```

void funfas_cad(){
    int i;
    char cpf[16];
    //Esse do serve para realizar os cadastros enquanto o usuario não digitar 1.
    do{
        system("cls");
        printf("CADASTRAR FUNCIONÁRIOS\n\n");
        printf("Nome: ");
        gets(fn.nome[fun_li]);
        printf("CPF: ");
        gets(cpf);

        //esse for serve para verificar a existência de um cadastro com o mesmo CPF inserido anteriormente.
        for(i=0;i<cad_fun;i++){
            //esse while serve para verificar a existência de um cadastro com o mesmo CPF inserido antes.
            while(strcmp(cpf, fn.cpf[i])==0){
                printf("\nCPF já Cadastrado!\n");
                strcpy(fn.nome[fun_li], "");
                getchar();
                funfas();
            }
        }
        strcpy(fn.cpf[fun_li], cpf);
        printf("Email: ");
        gets(fn.email[fun_li]);
        printf("Telefone: ");
        gets(fn.telefone[fun_li]);
        printf("Estado: ");
        gets(fn.estado[fun_li]);
        printf("Cidade: ");
        gets(fn.cidade[fun_li]);
    } while (getchar() != '1');
}

```

Fonte: Autoria própria, 2022.

Figura 44 - “do” e “switch case”

```

};

void funfas(){
    //Esse do serve para repetir a tela de funcionários enquanto o usuario não digitar 0 ou qualquer opção.
    do{
        system("cls");
        printf("DADOS DOS FUNCIONÁRIOS\n\n");
        printf("1 - Cadastrar Funcionários\n\n");
        printf("2 - Editar Funcionários\n\n");
        printf("3 - Exibir Funcionários\n\n");
        printf("4 - Excluir Funcionários\n\n");
        printf("0 - Retornar ao Menu Principal\n\n");
        printf("Sua Opção: ");
        scanf("%d", &op);
        setbuf(stdin, NULL);

        //esse switch serve para selecionar as opções de telas existentes.
        switch(op){
            case 1:
                system("cls");
                setbuf(stdin, NULL);
                funfas_cad();
                break;
            case 2:
                system("cls");
                setbuf(stdin, NULL);
                red_fun();
                break;
            case 3:
                system("cls");
                setbuf(stdin, NULL);
                exi_fun();

```

Fonte: Autoria própria, 2022.

Figura 45 - “do” e “switch case”

```

void tela_ad(){
    int a;
    //Esse do serve para repetir a tela de menu principal enquanto o usuario não digitar 0 ou qualquer opção.
    do{
        system("cls");
        printf("MENU PRINCIPAL\n\n");
        printf("1 - Dados dos Clientes\n\n");
        printf("2 - Dados dos Funcionários\n\n");
        printf("0 - Retornar a Tela Inicial\n\n");
        printf("Sua Opção: ");
        scanf("%d", &a);
        setbuf(stdin, NULL);

        //esse switch serve para selecionar as opções de telas existentes.
        switch(a){
            case 1:
                system("cls");
                setbuf(stdin, NULL);
                cliente();
                break;
            case 2:
                system("cls");
                setbuf(stdin, NULL);
                funfas();
                break;
            case 0:
                system("cls");
                setbuf(stdin, NULL);
                printf("Você Saiu da Conta!\nPressione ENTER Para Continuar!\n");
                getchar();
                tela_init();

```

Fonte: Autoria própria, 2022.

Figura 46 - “for”, “if” e “else if”

```

void login() {
    int i;
    char rec_user[16];
    char rec_sen[16];
    printf("LOGIN\n\n");
    printf("Usuário: ");
    gets(rec_user);
    printf("Senha: ");
    gets(rec_sen);
    //esse for serve para verificar a existencia de algum cadastro existente.
    for(i=0;i<linha;i++){
        //esse if serve para verificar se as informações inseridas estão corretas.
        if((strcmp(rec_user, ad.user[i])==0&&strcmp(rec_sen, ad.sen[i])==0)){
            system("cls");
            printf("Bem-vindo %s!\nPressione ENTER Para Continuar!\n", ad.user[i]);
            getchar();
            tela_ad();
            break;
        }
        //esse else if serve para verificar se as informações inseridas estão corretas.
        else if((strcmp(rec_user, ad.user[i+1])==0&&strcmp(rec_sen, ad.sen[i+1])==0)){
            system("cls");
            printf("Bem-vindo %s!\nPressione ENTER Para Continuar!\n", ad.user[i+1]);
            getchar();
            tela_ad();
            break;
        }
        else{
            printf("\nUsuário e/ou Senha Incorreto(s)!");
            getchar();
            tela_init();
        }
    }
}

```

Fonte: Autoria própria, 2022.

Figura 47 - “for”, “if” e “while”

```

void red_senha() {
    int i;
    char search_user[16];
    int search_cpf;
    char nov_sen[16];
    char conf_sen[16];
    printf("REDEFINIR SENHA\n\n");
    printf("Informe seu Usuário: ");
    gets(search_user);
    printf("Informe seu CPF: ");
    scanf("%d", &search_cpf);
    setbuf(stdin, NULL);
    //esse for serve para fazer uma verificação de cadastros para a redefinição de senha.
    for(i=0;i<linha;i++){
        //Esse if compara o Usuario e CPF inseridos com os do codigo fonte.
        if((strcmp(ad.user[i], search_user)==0&&ad.cpf[i]==search_cpf)){
            printf("\nDigite sua Nova Senha: ");
            gets(nov_sen);
            printf("Confirme sua Nova Senha: ");
            gets(conf_sen);
            //esse while Continua rodando até a senha inserida for igual a senha de confirmação.
            while(strcmp(nov_sen, conf_sen)!=0){
                printf("\nAs Senhas não coincidem, Por Favor Repita o Processo!\n\n");
                printf("\nDigite sua Nova Senha: ");
                gets(nov_sen);
                printf("Confirme sua Nova Senha: ");
                gets(conf_sen);
            }
            strcpy(ad.sen[i], conf_sen);
            printf("\nSenha Alterada com Sucesso! Pressione ENTER para voltar a Tela Inicial!");
            getchar();
        }
    }
}

```

Fonte: Autoria própria, 2022.

Figura 48 - “if” e “while”

```

        getchar();
        tela_init();
    }

    //Esse if compara o Usuario e cpf inseridos com os do codigo fonte.
    else if ((strcmp(ad.user[1], search_user)==0&&ad.cpf[1]==search_cpf)){
        printf("\nDigite sua Nova Senha: ");
        gets(nov_sen);
        printf("Confirme sua Nova Senha: ");
        gets(conf_sen);

        //esse while continua rodando até a senha inserida for igual a senha de confirmação.
        while(strcmp(nov_sen, conf_sen)!=0){
            printf("\nAs Senhas não coincidem, Por Favor Repita o Processo!\n\n");
            printf("\nDigite sua Nova Senha: ");
            gets(nov_sen);
            printf("Confirme sua Nova Senha: ");
            gets(conf_sen);
        }

        strcpy(ad.sen[1], conf_sen);
        printf("\nSenha Alterada com Sucesso! Pressione ENTER para voltar a Tela Inicial!");
        getchar();
        tela_init();
    }
    else{
        printf("\nUsuário não encontrado!");
        getchar();
        tela_init();
        break;
    }
}
}

```

Fonte: Autoria própria, 2022.

Figura 49 - “do while” e “switch case”

```

void tela_init(){
    int c;
    //Esse do serve para repetir a tela inicial enquanto o usuario não digitar 0 ou qualquer opção.
    do{
        setbuf(stdin, NULL);
        system("cls");
        printf("TELA INICIAL\n\n");
        printf("1 - Login\n\n");
        printf("2 - Redefinir Senha\n\n");
        printf("0 - Sair\n\n");
        printf("Sua Opção: ");
        scanf("%d", &c);

        //esse switch serve para selecionar as opções de telas existentes.
        switch(c){
            case 1:
                system("cls");
                setbuf(stdin, NULL);
                login();
                break;
            case 2:
                system("cls");
                setbuf(stdin, NULL);
                red_senha();
                break;
            case 0:
                system("cls");
                printf("Programa encerrado!\n");
                exit(EXIT_SUCCESS);
                break;
            default:

```

Fonte: Autoria própria, 2022.

CONCLUSÃO

Neste Projeto Integrado Multidisciplinar foram contemplados conceitos e significados de diversas práticas envolvendo, na Engenharia de *Software*, a metodologia principal utilizada no desenvolvimento do *Software* de gestão da Quinotek pela equipe terceirizada, o cronograma de atividades com suas devidas datas, tarefas e responsáveis, assim como a descrição das funções de cada profissional envolvido na equipe. Foi desenvolvido também um fluxograma de processos de *Software*, que, juntamente com descrições detalhadas, é possível compreender cada função que o *Software* irá realizar.

Ainda no âmbito de desenvolvimento do *Software* de gestão, foi concluído que os requisitos de sistema utilizados pela Quinotek para utilizar tanto o *Software* como outros tipos de serviços, são baixos, ou seja, não requerem um dispositivo muito potente para realizar tais funções, e junto disso, também foi definida a utilização do sistema operacional Windows da empresa Microsoft para ser a plataforma em que o *Software* irá funcionar, e como consequência disso, outras ferramentas como o pacote Office também serão utilizadas nos computadores da empresa, com a aquisição das suas respectivas licenças de *Software*, de modalidade empresarial.

As funções de controle lógico do *Software* descritos também foram projetados de forma a contemplar uma abordagem mais didática do funcionamento da linguagem C e juntamente disso, a contextualização das IDEs de programação, cujo foi dado como veredito, o uso do *Software* CodeBlocks para utilização das funcionalidades facilitadoras de uma IDE que tem como uma das opções de linguagem, a linguagem C.

No contexto geral da empresa também foi integrado sistemas de redes de dados e internet, portanto, foi decidido que, de acordo com a preferência no tipo de rede ser a rede LAN, a topologia dessa rede foi decidida consequentemente como a topologia de rede em Estrela.

Em questões como a segurança nesse projeto de desenvolvimento de *Software*, serão aplicadas as leis do código de proteção de dados descritos na LGPD, e em questões de segurança no *Software* em si, o sistema de segurança é feito em criptografia no código fonte, e assim também protegendo as informações contidas no sistema.

Já sobre o espaço utilizado pela empresa Quinotek nesse novo paradigma, foi tomado como decisão, uma modalidade de *Coworking*, para integrar as diversas necessidades da empresa com o seu negócio, dando oportunidade para contratações terceirizadas que a Quinotek precisou nesse momento.

Além disso, com o seu negócio de curto a médio prazo, foram observados custos e lucros das duas principais empresas envolvidas nesse projeto, por intermédio de gráficos com os valores ganhos nos meses durante o desenvolvimento do *software*, e tabelas indicando os gastos dos profissionais de ambas as empresas durante o período, dando sentido e integrando todos os outros custos anteriores ao valor total do *Software* de gestão, e ainda, integrando os critérios de prazos de desenvolvimento e metodologias presentes na Engenharia de *Software* como um todo, intertextualizando os elementos do desenvolvimento e, também como um todo, documentando esses processos.

REFERÊNCIAS

AFONSO, Joyce. **Coworking: o que é e como funciona este modelo de trabalho?**. Fala, Nubank. 2022. Disponível em: <<https://blog.nubank.com.br/coworking-o-que-e-e-como-funciona/>>. Acesso em: 04 Out. 2022.

BLOG DA CONTABILIZEI. **O que é Coworking? Como funciona esse modelo e vantagens**. 2022. Disponível em: <<https://www.contabilizei.com.br/contabilidade-online/o-que-e-coworking/>>. Acesso em: 09 Out. 2022.

BRASIL. **Constituição Da Republica Federativa Do Brasil**. Lei Nº 13.709. 2018. Brasília, DF. Disponível em: <https://www.planalto.gov.br/ccivil_03/_ato2015-2018/2018/lei/l13709.htm/>. Acesso em: 23 Nov. 2022.

CANTO, Daniel. **Conheça as desvantagens e vantagens do Windows**. KAZUK. 2021. Disponível em: <<https://kazuk.com.br/blog/vantagens-do-windows/>>. Acesso em: 04 Out. 2022.

CARRER, Felipe. **O que é uma linguagem de programação e quais os tipos existem?**. Rock Content. 2019. Disponível em: <<https://rockcontent.com/br/blog/linguagem-de-programacao/>>. Acesso em: 18 Out. 2022.

COLOSSETTI, Adriane. **Analista de projetos x Gerente de projetos**. LinkedIn. 2020. Disponível em: <<https://pt.linkedin.com/pulse/analista-de-projetos-x-gerente-adriane-colossetti-msc-pmp/>>. Acesso em: 18 Out. 2022.

DELL. **O que é um processador?**. 2022. Disponível em: <<https://www.dell.com/pt-br/lp/o-que-e-um-processador/>>. Acesso em: 18 Nov. 2022.

DOMÉ. **A LGPD já está em vigor e na iminência das fiscalizações e aplicação de penalidades, você já adequou sua empresa?**. 2021. Disponível em: <<https://vempradome.com.br/blog/lgpd-esta-em-vigor/>>. Acesso em: 21 Out. 2022.

ERREIRA, Carolina. **Código de ética e conduta: o que é e como implementá-lo na sua empresa**. UpLexis. 2022. Disponível em: <<https://uplexis.com.br/blog/artigos/codigo-de-etica-e-conduta/>>. Acesso em: 07 Out. 2022.

ESPINHA, Roberto Gil. **Gestão de projetos O que é + boas práticas do mercado**. Artia. 2019. Disponível em: <<https://artia.com/blog/gestao-de-projetos-o-que-e-para-que-serve/>>. Acesso em: 18 Out. 2022.

FELIPE LIRA ROCHA. **Modelos de Desenvolvimento de Software: resumo**. 2012. Disponível em: <<https://felipelirarocho.wordpress.com/2012/04/15/diversos-modelos-de-desenvolvimento-de-software-resumo/>>. Acesso em: 20 Out. 2022.

FREEPIK. **Diagrama de rede lan isométrico**. 2019. Disponível em: <https://br.freepik.com/vetores-premium/diagrama-de-rede-lan-isometrico_6029000.htm/>. Acesso em: 16 Out. 2022.

GARCIA, Lara Rocha; FERNANDES, Edson Aguilera; GONÇALVES, Rafael Augusto Moreno; BARRETTO, Marcos Ribeiro Pereira. **Lei geral de proteção de dados (LGPD)**: Guia de implantação. 1. ed. Blucher. 2020.

GERENCIATEC. **Tudo sobre Topologia de Rede: O que é, tipos, benefícios e a melhor escolha para seu provedor de internet**. 2022. Disponível em: <<https://www.gerenciatec.com.br/topologia-de-rede/>>. Acesso em: 16 Out. 2022.

HARDWARE. **Por que o SSD é melhor que o HD?**. 2022. Disponível em: <<https://www.hardware.com.br/artigos/por-que-ssd-e-melhor-que-hd/>>. Acesso em: 18 Nov 2022.

JACKSON, Lewis. **Modelo espiral: história, características, etapas, exemplo**. WARBLETONCOUNCIL. 2022. Disponível em: <<https://pt1.warbletoncouncil.org/modelo-espiral-5931/>>. Acesso em: 08 Out. 2022.

JANUARIO, Mariele. **O que é Criptografia de dados?**. WAN. 2021. Disponível em: <<https://www.wan.com.br/o-que-e-criptografia-de-dados/>>. Acesso em: 21 Out. 2022.

KUROSE, James F.; ROSS, Keith W.. **Redes de computadores e a internet**: Uma abordagem Top-down. 6. ed. Pearson. 2015.

MEDIUM. **O Modelo em Espiral de Boehm**. 2019. Disponível em: <<https://medium.com/contexto-delimitado/o-modelo-em-espiral-de-boehm-ed1d85b7df/>>. Acesso em: 08 Out. 2022.

MONITORA, Equipe. **As principais metodologias de desenvolvimento de software que você precisa saber**. MONITORA. 2020. Disponível em: <<https://www.monitoratec.com.br/blog/metodologias-de-desenvolvimento-de-software/>>. Acesso em: 07 Out. 2022.

NOLETO, Cairo. **Topologias de rede: o que são e quais os tipos?**. Blog da Trybe. 2022. Disponível em: <<https://blog.betrybe.com/tecnologia/topologias-de-rede/>>. Acesso em: 16 Out. 2022.

PESSANHA, Mário. **O que são as Redes de Computadores?**. The Automation Report. 2022. Disponível em: <<https://theautomationreport.com/o-que-sao-as-redes-de-computadores/>>. Acesso em: 14 Out. 2022.

PRESSMAN, Roger S.; MAXIM, Bruce R.. **Engenharia de Software**: Uma Abordagem Profissional. 8. ed. AMGH. 2016.

PÚBLIO, Angelo. **Por que você deve optar pelo sistema Windows para a sua empresa?**. 4Partner. 2014. Disponível em: <<https://blog.4partner.com.br/por-que-optimar-sistema-windows-empresa/>>. Acesso em: 04 Out. 2022.

RESENDE, Gabriela. **O que é LAN? Entenda quando usar essa rede.** Portal de Planos. 2022. Disponível em: <<https://portaldeplanos.com.br/artigos/lan/>>. Acesso em: 16 Out. 2022.

ROMVIEL, Raissa. **Quais são os tipos de Rede de Computadores? Entenda já!** Mais TIM. 2021. Disponível em: <<https://maistim.com.br/blog/tipos-de-rede/>>. Acesso em: 16 Out. 2022.

SACRAMENTO, Gabriel. **Modelo Cascata em Projetos de Software: Entenda essa Metodologia.** Tera. 2021. Disponível em: <<https://blog.somostera.com/desenvolvimento-web/modelo-cascata/>>. Acesso em: 20 Out. 2022.

SCHILD, Herbert. **C: Completo e Total.** 3. ed. Pearson Universidades. 1997.

SCHULTZ, Felix. **LCPD: o que é, como funciona e para que serve (Guia Completo).** Milvus. 2019. Disponível em: <<https://blog.milvus.com.br/guia-lcpd-completo/>>. Acesso em: 21 Out. 2022.

SEBESTA, Robert W.. **Conceitos de linguagens de programação.** 11. ed. Bookman. 2018.

SEBRAE. **O que é uma startup?** SEBRAE. 2022. Disponível em: <<https://www.sebrae.com.br/sites/PortalSebrae/artigos/o-que-e-uma-startup,6979b2a178c83410VgnVCM1000003b74010aRCRD/>>. Acesso em: 03 Out. 2022.

SOFFNER, Renato. **Algoritmos e programação em linguagem C.** 1. ed. Saraiva Uni. 2013.

SOMMERVILLE, Ian. **Engenharia de Software.** 10. ed. Pearson Universidades. 2019.

SOS, **Memória RAM: para que serve?** 2019. Disponível em: <<https://www.sos.com.br/noticias/tecnologia/memoria-ram-para-que-serve/>>. Acesso em 18 Nov. 2022.

SOUTO, Mario. **Front-end, Back-end e Full Stack.** Alura. 2022. Disponível em: <https://www.alura.com.br/artigos/o-que-e-front-end-e-back-end?>. Acesso em: 18 Out. 2022.

TANENBAUM, Andrew S.; WETHERALL, David. **Redes de Computadores.** 5. ed. Pearson Universidades. 2011.

TECMUNDO. **Windows 10: conheça em detalhes o novo sistema operacional da Microsoft.** 2015. Disponível em: <<https://www.tecmundo.com.br/windows-10/83887-windows-10-detalhes-novo-sistema-operacional-microsoft-tudo-sobre-download.htm/>>. Acesso em: 18 Nov. 2022.

TOTVS. **GoodData - Fast Analytics - Como criptografar a senha do usuário no arquivo de configuração do Agent.** 2021. Disponível em: <<https://centraldeatendimento.totvs.com/hc/pt-br/articles/1500000248761-GoodData-Fast-Analytics-Como-criptografar-a-senha-do-usu%C3%A1rio-no-arquivo-de-configura%C3%A7%C3%A3o-do-Agent/>>. Acesso em: 21 Out. 2022.

TOTVS. **Metodologia ágil: o que é e como implementar.** TOTVS. 2021. Disponível em: <<https://www.totvs.com/blog/negocios/metodologia-agil/>>. Acesso em: 07 Out. 2022.

VASCONCELOS, Laércio. **Hardware na prática.** 2. ed. LVC. 2007.

Código do Software em Linguagem C:

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <locale.h>
//#define linha 2" Serve para definir quantos cadastros podem ser feitos na estrutura
administrador.
#define linha 2
/*#define cad_fun 1000" Serve para definir a quantidade de cadastros a serem
realizados nas
estruturas cad_funcionario e cad_cliente.*/
#define cad_fun 1000

int op;
static int fun_li=0;
static int cli_li=0;

void tela_init();
void login();
void red_senha();
void tela_ad();
void funfas();
void funfas_cad();
void exi_fun();
void red_fun();
void del_fun();
void cliente();
void cad_cli();
void exi_cli();
void red_cli();
void del_cli();
//Struct cria um tipo de grupo de dados a serem guardados em uma unica variavel.
struct administrador{
    char user[linha][100];
    char sen[linha][100];
    int cpf[linha];
};
struct administrador ad;

struct cad_funcionario{
    char nome[cad_fun][100];
    char cpf[cad_fun][100];
    char email[cad_fun][100];
    char telefone[cad_fun][100];
    char estado[cad_fun][100];
    char cidade[cad_fun][100];
    char bairro[cad_fun][100];
    char rua[cad_fun][100];
    char numero[cad_fun][100];
```

```

    char complemento[cad_fun][100];
    char cargo[cad_fun][100];
    char data[cad_fun][100];
    char salario[cad_fun][100];
    char recebe[cad_fun][100];
};
struct cad_funcionario fn;

struct cad_cliente{
    char nome[cad_fun][100];
    char cnpj[cad_fun][100];
    char email[cad_fun][100];
    char telefone[cad_fun][100];
    char estado[cad_fun][100];
    char cidade[cad_fun][100];
    char bairro[cad_fun][100];
    char rua[cad_fun][100];
    char numero[cad_fun][100];
    char data[cad_fun][100];
    char fat[cad_fun][100];
};
struct cad_cliente cli;

int main(){
    setlocale(LC_ALL, "portuguese");
    //strcpy é um comando usado para inserir dados dentro de uma variável do tipo
    char.
    strcpy(ad.user[0], "paulo nogueira");
    strcpy(ad.sen[0], "thrat349");
    ad.cpf[0]=32165498755;
    strcpy(ad.user[1], "david gomes");
    strcpy(ad.sen[1], "thrat747");
    ad.cpf[1]=55789456123;
    tela_init();
    return 0;
}

void del_cli(){
    int del;
    int i;
    char rec_cnpj[100];
    printf("EXCLUIR CLIENTES\n\n");
    printf("Digite o CNPJ da Instituição que deseja deletar\n");
    printf("CNPJ: ");
    gets(rec_cnpj);
    //for nesse caso esta sendo utilizado para apagar o cadastro de um cliente.
    for(i=0;i<cad_fun;i++){
        //if nesse caso esta sendo usado para validar a existência de um cadastro através
        do CNPJ inserido anteriormente.
        if(strcmp(rec_cnpj, cli.cnpj[i])==0){

```

```

    system("cls");
    printf("Instituição: %s\n\n", cli.nome[i]);
    printf("1 - Deletar Cadastro\n\n0 - Voltar\n\nSua Opção: ");
    scanf("%d", &del);
    setbuf(stdin, NULL);
    //esse if serve para o usuário ter certeza que quer deletar o cadastro do cliente.
    if(del==1){
        strcpy(cli.nome[i], "");
        strcpy(cli.cnpj[i], "");
        strcpy(cli.email[i], "");
        strcpy(cli.telefone[i], "");
        strcpy(cli.estado[i], "");
        strcpy(cli.cidade[i], "");
        strcpy(cli.bairro[i], "");
        strcpy(cli.rua[i], "");
        strcpy(cli.numero[i], "");
        strcpy(cli.data[i], "");
        strcpy(cli.fat[i], "");
        printf("\nCadastro deletado com Sucesso!\n");
        getchar();
        cliente();
    }
    else{
        cliente();
    }
}
}
system("cls");
printf("Usuário não Encontrado!");
getchar();
cliente();
}

void red_cli(){
    int val;
    int i;
    char rec_cnpj[100];
    printf("EDITAR CLIENTES\n\n");
    printf("Digite o CNPJ da Instituição que deseja editar\n");
    printf("CNPJ: ");
    gets(rec_cnpj);
    //for nesse caso esta sendo utilizado para editar o cadastro de um cliente.
    for(i=0;i<cad_fun;i++){
        //if nesse caso esta sendo usado para validar a existência de um cadastro através
        do CNPJ inserido anteriormente.
        if(strcmp(rec_cnpj, cli.cnpj[i])==0){
            system("cls");
            printf("Usuário: %s\n\n1 - Editar Informações\n\n0 - Retornar\n\nSua Opção:
", cli.nome[i]);
            scanf("%d", &val);

```

```

        setbuf(stdin, NULL);
//esse if serve para o usuário ter certeza que quer editar o cadastro do Cliente.
        if(val==1){
            system("cls");
            printf("Nome da Instituição: ");
            gets(cli.nome[i]);
            printf("CNPJ: ");
            gets(cli.cnpj[i]);
            printf("Email: ");
            gets(cli.email[i]);
            printf("Telefone: ");
            gets(cli.telefone[i]);
            printf("Estado: ");
            gets(cli.estado[i]);
            printf("Cidade: ");
            gets(cli.cidade[i]);
            printf("Bairro: ");
            gets(cli.bairro[i]);
            printf("Rua: ");
            gets(cli.rua[i]);
            printf("Número: ");
            gets(cli.numero[i]);
            printf("Data de vencimento da última fatura: ");
            gets(cli.data[i]);
            printf("Situação da fatura: ");
            gets(cli.fat[i]);
            printf("\nDados Atualizados com Sucesso!\n");
            getchar();
            cliente();
        }
        else{
            cliente();
        }
    }
}
system("cls");
printf("Usuário não Encontrado!");
getchar();
cliente();
}

void exi_cli(){
    int i;
    printf("EXIBIR CLIENTES\n\n");
    printf("Pressione ENTER para voltar!\n\n");
    printf("-----\n");
    //Esse for está sendo usado para exibir todos os clientes cadastrados.
    for(i=0;i<cad_fun;i++){
        //esse if serve para validar a existência dos clientes para exibi-los.
        if(strlen(cli.nome[i])>0){

```

```

        printf("Nome da Instituição: %s\n", cli.nome[i]);
        printf("CNPJ: %s\n", cli.cnpj[i]);
        printf("Email: %s\n", cli.email[i]);
        printf("Telefone: %s\n", cli.telefone[i]);
        printf("Estado: %s\n", cli.estado[i]);
        printf("Cidade: %s\n", cli.cidade[i]);
        printf("Bairro: %s\n", cli.bairro[i]);
        printf("Rua: %s\n", cli.rua[i]);
        printf("Número: %s\n", cli.numero[i]);
        printf("Data de vencimento da última fatura: %s\n", cli.data[i]);
        printf("Situação da fatura: %s\n", cli.fat[i]);
        printf("-----\n");
    }
}

getchar();
cliente();
}

void cad_cli(){
    int i;
    char cnpj[100];
    /*do while serve para repetir uma ação até o momento em que atingir a condição
    imposta no while.
    nesse caso serve para repetir o cadastro até o usuário digitar "0".*/
    do{
        system("cls");
        printf("CADASTRAR CLIENTES\n\n");
        printf("Nome da Instituição: ");
        gets(cli.nome[cli_li]);
        printf("CNPJ: ");
        gets(cnpj);
        //Esse for serve para verificar a existência de algum cadastro com o CNPJ
        inserido anteriormente.
        for(i=0;i<cad_fun;i++){
            while(strcmp(cnpj, cli.cnpj[i])==0){
                printf("\nCNPJ já Cadastrado!\n");
                strcpy(cli.nome[cli_li], "");
                getchar();
                cliente();
            }
        }
        strcpy(cli.cnpj[cli_li], cnpj);
        printf("Email: ");
        gets(cli.email[cli_li]);
        printf("Telefone: ");
        gets(cli.telefone[cli_li]);
        printf("Estado: ");
        gets(cli.estado[cli_li]);
        printf("Cidade: ");
        gets(cli.cidade[cli_li]);
    }
}

```

```

    printf("Bairro: ");
    gets(cli.bairro[cli_li]);
    printf("Rua: ");
    gets(cli.rua[cli_li]);
    printf("Número: ");
    gets(cli.numero[cli_li]);
    printf("Data de vencimento da última fatura: ");
    gets(cli.data[cli_li]);
    printf("Situação da fatura: ");
    gets(cli.fat[cli_li]);
    cli_li++;
    printf("\n\nCliente cadastrado com sucesso!");
    printf("\n1 - Cadastrar outro Cliente\n0 - Voltar\n\nSua Opção: ");
    scanf("%d", &op);
    setbuf(stdin, NULL);
}while(op==1);
cliente();
}

void cliente(){
    //esse do while serve para repetir a tela dos clientes enquanto o usuário não
    digitar 0.
    do{
        system("cls");
        printf("DADOS DOS CLIENTES\n\n");
        printf("1 - Cadastrar Clientes\n\n");
        printf("2 - Editar Clientes\n\n");
        printf("3 - Exibir Clientes\n\n");
        printf("4 - Excluir Clientes\n\n");
        printf("0 - Retornar ao Menu Principal\n\n");
        printf("Sua Opção: ");
        scanf("%d", &op);
        setbuf(stdin, NULL);
        //switch case serve para reduzir a quantidade de ifs e elses, muito utilizado em
        menus como esse.
        switch(op){
            case 1:
                system("cls");
                setbuf(stdin, NULL);
                cad_cli();
                break;
            case 2:
                system("cls");
                setbuf(stdin, NULL);
                red_cli();
                break;
            case 3:
                system("cls");
                setbuf(stdin, NULL);
                exi_cli();

```

```

        break;
    case 4:
        system("cls");
        setbuf(stdin, NULL);
        del_cli();
        break;
    case 0:
        system("cls");
        setbuf(stdin, NULL);
        tela_ad();
        break;
    default:
        system("cls");
        printf("Opção inválida!\n");
        getch();
        break;
    }
}while(op!=0);
tela_ad();
}

void del_fun(){
    int del;
    int i;
    char rec_cpf[100];
    printf("EXCLUIR FUNCIONÁRIOS\n\n");
    printf("Digite o CPF do Funcionário que deseja deletar\n");
    printf("CPF: ");
    gets(rec_cpf);
    //esse for serve para apagar o funcionário cadastrado.
    for(i=0;i<cad_fun;i++){
        //esse if serve para validar a existência de um cadastro que tenha o CPF digitado.
        if(strcmp(rec_cpf, fn.cpf[i])==0){
            system("cls");
            ("-----");
            printf("Funcionário: %s\n\n", fn.nome[i]);
            printf("1 - Deletar Cadastro\n\n0 - Voltar\n\nSua Opção: ");
            scanf("%d", &del);
            setbuf(stdin, NULL);
            //Esse if serve para validar a escolha do usuário.
            if(del==1){
                strcpy(fn.nome[i], "");
                strcpy(fn.cpf[i], "");
                strcpy(fn.email[i], "");
                strcpy(fn.telefone[i], "");
                strcpy(fn.estado[i], "");
                strcpy(fn.cidade[i], "");
                strcpy(fn.bairro[i], "");
                strcpy(fn.rua[i], "");
                strcpy(fn.numero[i], "");
            }
        }
    }
}

```



```

        strcpy(fn.complemento[i], "");
        strcpy(fn.cargo[i], "");
        strcpy(fn.data[i], "");
        strcpy(fn.salario[i], "");
        strcpy(fn.recebe[i], "");
        printf("\nFuncionário Deletado com Sucesso!\n");
        getchar();
        funfas();
    }
    else{
        funfas();
    }
}
}
system("cls");
printf("Usuário não Encontrado!");
getchar();
funfas();
}

void red_fun(){
    int val;
    int i;
    char rec_cpf[100];
    printf("EDITAR FUNCIONÁRIOS\n\n");
    printf("Digite o CPF do Funcionário para editar seus dados\n");
    printf("CPF: ");
    gets(rec_cpf);
    //esse for serve para editar o cadastro de algum funcionário.
    for(i=0;i<cad_fun;i++){
        //esse if serve para validar se existe algum cadastro que contenha o CPF digitado.
        if(strcmp(rec_cpf, fn.cpf[i])==0){
            system("cls");
            printf("Usuário: %s\n\n1 - Editar Informações\n\n0 - Retornar\n\nSua Opção:
", fn.nome[i]);
            scanf("%d", &val);
            setbuf(stdin, NULL);
            //esse if serve para verificar a decisão do usuário para editar o funcionário.
            if(val==1){
                system("cls");
                printf("Nome: ");
                gets(fn.nome[i]);
                printf("CPF: ");
                gets(fn.cpf[i]);
                printf("Email: ");
                gets(fn.email[i]);
                printf("Telefone: ");
                gets(fn.telefone[i]);
                printf("Estado: ");
                gets(fn.estado[i]);
            }
        }
    }
}

```

```

        printf("Cidade: ");
        gets(fn.cidade[i]);
        printf("Bairro: ");
        gets(fn.bairro[i]);
        printf("Rua: ");
        gets(fn.rua[i]);
        printf("Número: ");
        gets(fn.numero[i]);
        printf("Complemento: ");
        gets(fn.complemento[i]);
        printf("Cargo: ");
        gets(fn.cargo[i]);
        printf("Data de Admissão: ");
        gets(fn.data[i]);
        printf("Salário: ");
        gets(fn.salario[i]);
        printf("Data do Pagamento: ");
        gets(fn.recebe[i]);
        printf("\nDados Atualizados com Sucesso!\n");
        getchar();
        funfas();
    }
    else{
        funfas();
    }
}
}
system("cls");
printf("Usuário não Encontrado!");
getchar();
funfas();
}

void exi_fun(){
    int i;
    printf("EXIBIR FUNCIONÁRIOS\n\n");
    printf("Pressione ENTER para voltar!\n\n");
    printf("-----\n");
    //Esse for serve para exibir todos os cadastros existentes.
    for(i=0;i<cad_fun;i++){
        //Esse if serve para validar a existência dos cadastros dos funcionários nos
        vetores.
        if(strlen(fn.nome[i])>0){
            printf("Nome: %s\n", fn.nome[i]);
            printf("CPF: %s\n", fn.cpf[i]);
            printf("Email: %s\n", fn.email[i]);
            printf("Telefone: %s\n", fn.telefone[i]);
            printf("Estado: %s\n", fn.estado[i]);
            printf("Cidade: %s\n", fn.cidade[i]);
            printf("Bairro: %s\n", fn.bairro[i]);

```

```

        printf("Rua: %s\n", fn.rua[i]);
        printf("Número: %s\n", fn.numero[i]);
        printf("Complemento: %s\n", fn.complemento[i]);
        printf("Cargo: %s\n", fn.cargo[i]);
        printf("Data de Admissão: %s\n", fn.data[i]);
        printf("Salário: %s\n", fn.salario[i]);
        printf("Data do Pagamento: %s\n", fn.recebe[i]);
        printf("-----\n");
    }
}
getchar();
funfas();
}

void funfas_cad(){
    int i;
    char cpf[100];
    //Esse do serve para realizar os cadastros enquanto o usuario não digitar 1.
    do{
        system("cls");
        printf("CADASTRAR FUNCIONÁRIOS\n\n");
        printf("Nome: ");
        gets(fn.nome[fun_li]);
        printf("CPF: ");
        gets(cpf);
        //esse for serve para verificar a existência de um cadastro com o mesmo CPF
        inserido anteriormente.
        for(i=0;i<cad_fun;i++){
            //esse while serve para verificar a existência de um cadastro com o mesmo CPF
            inserido antes.
            while(strcmp(cpf, fn.cpf[i])==0){
                printf("\nCPF já Cadastrado!\n");
                strcpy(fn.nome[fun_li], "");
                getchar();
                funfas();
            }
        }
        strcpy(fn.cpf[fun_li], cpf);
        printf("Email: ");
        gets(fn.email[fun_li]);
        printf("Telefone: ");
        gets(fn.telefone[fun_li]);
        printf("Estado: ");
        gets(fn.estado[fun_li]);
        printf("Cidade: ");
        gets(fn.cidade[fun_li]);
        printf("Bairro: ");
        gets(fn.bairro[fun_li]);
        printf("Rua: ");
        gets(fn.rua[fun_li]);
    }
}

```

```

    printf("Número: ");
    gets(fn.numero[fun_li]);
    printf("Complemento: ");
    gets(fn.complemento[fun_li]);
    printf("Cargo: ");
    gets(fn.cargo[fun_li]);
    printf("Data de Admissão: ");
    gets(fn.data[fun_li]);
    printf("Salário: ");
    gets(fn.salario[fun_li]);
    printf("Data do Pagamento: ");
    gets(fn.recebe[fun_li]);
    fun_li++;
    printf("\n\nFuncionário cadastrado com sucesso!");
    printf("\n1 - Cadastrar outro Funcionário\n0 - Voltar\n\nSua Opção: ");
    scanf("%d", &op);
    setbuf(stdin, NULL);
}while(op==1);
funfas();
}

void funfas(){
    //Esse do serve para repetir a tela de funcionários enquanto o usuario não digitar
    0 ou qualquer opção.
    do{
        system("cls");
        printf("DADOS DOS FUNCIONÁRIOS\n\n");
        printf("1 - Cadastrar Funcionários\n\n");
        printf("2 - Editar Funcionários\n\n");
        printf("3 - Exibir Funcionários\n\n");
        printf("4 - Excluir Funcionários\n\n");
        printf("0 - Retornar ao Menu Principal\n\n");
        printf("Sua Opção: ");
        scanf("%d", &op);
        setbuf(stdin, NULL);
    //esse switch serve para selecionar as opções de telas existentes.
    switch(op){
        case 1:
            system("cls");
            setbuf(stdin, NULL);
            funfas_cad();
            break;
        case 2:
            system("cls");
            setbuf(stdin, NULL);
            red_fun();
            break;
        case 3:
            system("cls");
            setbuf(stdin, NULL);

```

```

        exi_fun();
        break;
    case 4:
        system("cls");
        setbuf(stdin, NULL);
        del_fun();
        break;
    case 0:
        system("cls");
        setbuf(stdin, NULL);
        tela_ad();
        break;
    default:
        system("cls");
        printf("Opção inválida!\n");
        getch();
        break;
    }
}while(op!=0);
tela_ad();
}

void tela_ad(){
    int a;
    //Esse do serve para repetir a tela de menu principal enquanto o usuario não
    digitar 0 ou qualquer opção.
    do{
        system("cls");
        printf("MENU PRINCIPAL\n\n");
        printf("1 - Dados dos Clientes\n\n");
        printf("2 - Dados dos Funcionários\n\n");
        printf("0 - Retornar a Tela Inicial\n\n");
        printf("Sua Opção: ");
        scanf("%d", &a);
        setbuf(stdin, NULL);
        //esse switch serve para selecionar as opções de telas existentes.
        switch(a){
            case 1:
                system("cls");
                setbuf(stdin, NULL);
                cliente();
                break;
            case 2:
                system("cls");
                setbuf(stdin, NULL);
                funfas();
                break;
            case 0:
                system("cls");
                setbuf(stdin, NULL);

```

```

        printf("Você Saiu da Conta!\nPressione ENTER Para voltar a Tela Inicial!\n");
        getchar();
        tela_init();
        break;
    default:
        system("cls");
        printf("Opção inválida!\n");
        getch();
        break;
    }
}while(a!=0);
tela_init();
}

void login(){
    int i;
    char rec_user[100];
    char rec_sen[100];
    printf("LOGIN\n\n");
    printf("Usuário: ");
    gets(rec_user);
    printf("Senha: ");
    gets(rec_sen);
    //esse for serve para verificar a existencia de algum cadastro existente.
    for(i=0;i<linha;i++){
        //esse if serve para verificar se as informações inseridas estão corretas.
        if((strcmp(rec_user, ad.user[0])==0&&strcmp(rec_sen, ad.sen[0])==0)){
            system("cls");
            printf("Bem-vindo %s!\nPressione ENTER Para Continuar!\n", ad.user[0]);
            getchar();
            tela_ad();
            break;
        }
        //esse else if serve para verificar se as informações inseridas estão corretas.
        else if((strcmp(rec_user, ad.user[1])==0&&strcmp(rec_sen, ad.sen[1])==0)){
            system("cls");
            printf("Bem-vindo %s!\nPressione ENTER Para Continuar!\n", ad.user[1]);
            getchar();
            tela_ad();
            break;
        }
        else{
            printf("\nUsuário e/ou Senha Incorreto(s)!");
            getchar();
            tela_init();
            break;
        }
    }
}
}

```

```

void red_senha(){
    int i;
    char search_user[100];
    int search_cpf;
    char nov_sen[100];
    char conf_sen[100];
    printf("REDEFINIR SENHA\n\n");
    printf("Informe seu Usuário: ");
    gets(search_user);
    printf("Informe seu CPF: ");
    scanf("%d", &search_cpf);
    setbuf(stdin, NULL);
    //esse for serve para fazer uma verificação de cadastros para a redefinição de
    senha.
    for(i=0;i<linha;i++){
        //Esse if compara o Usuario e CPF inseridos com os do codigo fonte.
        if((strcmp(ad.user[0], search_user)==0&&ad.cpf[0]==search_cpf)){
            printf("\nDigite sua Nova Senha: ");
            gets(nov_sen);
            printf("Confirme sua Nova Senha: ");
            gets(conf_sen);
            //esse while Continua rodando até a senha inserida for igual a senha de
            confirmação.
            while(strcmp(nov_sen, conf_sen)!=0){
                printf("\nAs Senhas não coincidem, Por Favor Repita o Processo!\n\n");
                printf("\nDigite sua Nova Senha: ");
                gets(nov_sen);
                printf("Confirme sua Nova Senha: ");
                gets(conf_sen);
            }
            strcpy(ad.sen[0], conf_sen);
            printf("\nSenha Alterada com Sucesso! Pressione ENTER para voltar a Tela
            Inicial!");
            getchar();
            tela_init();
        }
        //Esse if compara o Usuario e cpf inseridos com os do codigo fonte.
        else if((strcmp(ad.user[1], search_user)==0&&ad.cpf[1]==search_cpf)){
            printf("\nDigite sua Nova Senha: ");
            gets(nov_sen);
            printf("Confirme sua Nova Senha: ");
            gets(conf_sen);
            //esse while continua rodando até a senha inserida for igual a senha de
            confirmação.
            while(strcmp(nov_sen, conf_sen)!=0){
                printf("\nAs Senhas não coincidem, Por Favor Repita o Processo!\n\n");
                printf("\nDigite sua Nova Senha: ");
                gets(nov_sen);
                printf("Confirme sua Nova Senha: ");
                gets(conf_sen);
            }
        }
    }
}

```

```

    }
    strcpy(ad.sen[1], conf_sen);
    printf("\nSenha Alterada com Sucesso! Pressione ENTER para voltar a Tela
Inicial!");
    getchar();
    tela_init();
}
else{
    printf("\nUsuário não encontrado!");
    getchar();
    tela_init();
    break;
}
}
}

```

```

void tela_init(){
    int c;
    //Esse do serve para repetir a tela inicial enquanto o usuario não digitar 0 ou
    qualquer opção.
    do{
        setbuf(stdin, NULL);
        system("cls");
        printf("TELA INICIAL\n\n");
        printf("1 - Login\n\n");
        printf("2 - Redefinir Senha\n\n");
        printf("0 - Sair\n\n");
        printf("Sua Opção: ");
        scanf("%d", &c);
        //esse switch serve para selecionar as opções de telas existentes.
        switch(c){
        case 1:
            system("cls");
            setbuf(stdin, NULL);
            login();
            break;
        case 2:
            system("cls");
            setbuf(stdin, NULL);
            red_senha();
            break;
        case 0:
            system("cls");
            printf("Programa encerrado!\n");
            exit(EXIT_SUCCESS);
            break;
        default:
            system("cls");
            printf("Opção inválida!\n");
            getch();

```



```
        break;
    }
}while(c!=0);
}
```