**UNIVERSIDADE PAULISTA – UNIP EAD**

**Projeto Integrado Multidisciplinar IV**

**Curso Superior de Tecnologia em**

**Análise e Desenvolvimento de Sistemas**

Bruna Regina Nunes dos Santos RA: 0606942

Diogo Cavalcanti Dos Santos RA: 0601375

Gabriel Maia de Andrade RA: 0608185

Paula Helena Câmara RA: 0605092

Natã Rodrigues Frois RA: 0601016

**SISTEMA EM LINGUAGEM C PARA CADASTRAR PACIENTES COM COVID-19**

São Paulo – SP

**2022**

Bruna Regina Nunes dos Santos RA: 0606942

Diogo Cavalcanti Dos Santos RA: 0601375

Gabriel Maia de Andrade RA: 0608185

Paula Helena Câmara RA: 0605092

Natã Rodrigues Frois RA: 0601016

SISTEMA EM LINGUAGEM C PARA CADASTRAR PACIENTES COM COVID-19

Projeto Integrado Multidisciplinar IV

em Análise e Desenvolvimento de Sistemas

Projeto Integrado Multidisciplinar IV, apresentado a Universidade Paulista UNIP, como exigência parcial para a conclusão do semestre no curso Superior de Tecnologia em Análise e Desenvolvimento de Sistemas.

Prof. Orientadora: Dra. Vanessa Lessa

São Paulo ­- SP

**2022**

**UNIVERSIDADE PAULISTA – UNIP EAD**

**Folha De Aprovação**

Bruna Regina Nunes dos Santos RA: 0606942

Diogo Cavalcanti Dos Santos RA: 0601375

Gabriel Maia de Andrade RA: 0608185

Paula Helena Câmara RA: 0605092

Natã Rodrigues Frois RA: 0601016

Projeto Integrado Multidisciplinar IV, apresentado a Universidade Paulista UNIP, como exigência parcial para a conclusão do semestre no curso Superior de Tecnologia em Análise e Desenvolvimento de Sistemas.

**Projeto Multidisciplinar PIM IV: SISTEMA EM LINGUAGEM C PARA CADASTRAR PACIENTES COM COVID-19**

Aprovado em:

BANCA EXAMINADORA

Prof. Nome do Professor

BANCA EXAMINADORA

Prof. Nome do Professor

BANCA EXAMINADORA

Prof. Nome do Professor

São Paulo - SP

**2022**

**RESUMO**

Passamos por um momento aterrorizante, depois que descobrimos que uma doença de nível mundial estava por vir. Não havia preparo e muito menos vacinas para essa nova comorbidade. Foram milhões de mortes, que deixaram todos nós em pânico sem saber como combater a COVID 19 naquele momento. Felizmente, vacinas foram criadas, e houve muitos investimentos na tecnologia para atender a população mundial, assim salvando muitas vidas. O nosso projeto tem como finalidade colocar em prática os aprendizados que tivemos sobre a linguagem C durante este semestre. Vamos desenvolver um sistema em C, utilizando o CODE::BLOCKS, onde será possível colocar as informações sobre pacientes contaminados e que precisam de um acompanhamento cuidadoso, ainda mais se tiverem comorbidades.

PALAVRAS-CHAVE: COVID 19, LINGUAGEM C, CODE::BLOCKS

***ABSTRACT***

We went through a terrifying time after we found out that a world-class disease was coming. There was no preparation, let alone vaccines, for this new comorbidity. There were millions of deaths, which left all of us in a panic without knowing how to fight COVID 19 at that moment. Fortunately, vaccines were created, and there were many investments in technology to serve the world's population, thus saving many lives. Our project aims to put into practice what we learned about the C language during this semester. We are going to develop a system in C, using CODE::BLOCKS, where it will be possible to place information about infected patients who need careful monitoring, even more so if they have comorbidities.

KEYWORDS: COVID 19, Language C, CODE::BLOCKS

**SUMÁRIO**

[**1 Introdução 6**](#_Toc120125384)

[**2 Definição História e Evolução da Engenharia de Software 7**](#_Toc120125385)

[**3 Papel do Software 9**](#_Toc120125386)

[**4 Características do Software 10**](#_Toc120125387)

[**5 Processo de Software, Infraestrutura de Processo 11**](#_Toc120125388)

[**6 Ciclo de vida 12**](#_Toc120125389)

[**6.1 Codificar e consertar (*Code-and-fix*) 13**](#_Toc120125390)

[**7 Linguagem e Técnicas de Programação 15**](#_Toc120125391)

[**7.1 Origem Da Linguagem C 15**](#_Toc120125392)

[**7.2 Variáveis em ponto flutuante – float, double e void 15**](#_Toc120125393)

[**7.3 Variáveis 16**](#_Toc120125394)

[**7.4 Características da Linguagem C 17**](#_Toc120125395)

[**8 Desenvolvimento Do Sistema 18**](#_Toc120125396)

[**8.1 Construção do projeto 18**](#_Toc120125397)

[**8.2 Tela de cadastro e informações do diagnóstico 19**](#_Toc120125398)

[**8.3 Comprovando que o nosso sistema funciona e executa os dados 21**](#_Toc120125399)

[**8.4 Variáveis Usadas 22**](#_Toc120125400)

[**8.5 Funções Usadas 22**](#_Toc120125401)

[**8.6 Estruturas Usadas 22**](#_Toc120125402)

[**8.7 Bibliotecas Usadas no Projeto 22**](#_Toc120125403)

[**9 Manual do Usuário 23**](#_Toc120125404)

[**10 Conclusão. 24**](#_Toc120125405)

[**11 Referências 25**](#_Toc120125406)

# 1 Introdução

Neste projeto proposto pela Universidade Paulista UNIP, iremos desenvolver um sistema em C utilizando o software Code::Blocks, que será utilizado para cadastramento dos pacientes com COVID 19, que precisam ser monitorados e acompanhados de forma prática e rápida. Com o auxílio da disciplina Linguagem e Técnicas de Programação e Engenharia de Software I, onde aprendemos os conceitos sobre a linguagem C, vamos colocar esses aprendizados em prática desenvolvendo o Projeto Multidisciplinar IV. Com a pandemia, o mundo precisou realizar inovações em todos os setores, principalmente na saúde, o que acarretou o desenvolvimento de softwares para atender à alta demanda de pessoas enfermas. Todos os países exigem tecnologia e equipamentos para o combate à doença, e conforme fomos aprendendo sobre esta doença, foram feitas novas inovações no campo da saúde. Vamos criar um programa para que seja feito o controle de pessoas que foram contaminadas por essa doença. Primeiramente, iremos criar o programa para alimentar com os dados do paciente, calculando à idade e mostrando se há alguma comorbidade. Para que seja feito um acompanhamento ainda mais cuidadoso.

# 2 Definição História e Evolução da Engenharia de Software

Primeiramente, segue definição de Engenharia de software segundo Sommerville (2003).

“Definido por um conjunto de componentes abstratos de software (estruturas de dados e algoritmos) descritos na forma de procedimentos, funções, módulos, objetos ou agentes interconectados entre si, compondo, assim, a arquitetura de software, e que devem ser executados em sistemas computacionais.”

Historicamente o termo Engenharia de Software foi definido pelo professor Friedrich Bauer, em 1960, ano em que trabalhava como Professor na Universidade de Tecnologia de Munique, Alemanha, começou a ser usado de maneira oficial em 1968 na Conferência sobre Engenharia de Software da OTAN Segundo o professor Bauer, “Engenharia de Software é a criação e a utilização de sólidos princípios da engenharia a fim de se obter software de maneira econômica, que seja confiável e que trabalhe eficientemente em computadores reais”. Nos anos 40 os primeiros usuários de computadores escreviam código de máquina, ou seja, instruções executadas pelos computadores à mão. As primeiras ferramentas de software, tais como, montadores de código e interpretadores, foram criados nos anos 50 com o objetivo de melhorar a qualidade do código escrito e a produtividade dos programadores, surgindo, assim, a primeira geração de compiladores que aperfeiçoavam o código de máquina. Nos anos 60 essas ferramentas evoluíram ainda mais permitindo maior qualidade nos programas, e produtividade mais alta por parte dos programadores. Nesta época o conceito de engenharia de software começou a ser discutido tendo como propósito aplicar os conceitos de engenharia ao desenvolvimento de sistemas de software complexos. A partir de então, ferramentas colaborativas e repositórios de código apareceram na década de 1970 e 80 os computadores pessoais e as estações de trabalho tornaram-se comuns. Os profissionais de desenvolvimento de software também viram surgir nesta época a primeira linguagem de programação orientada a objetos comercial, denominada Smalltalk.

A programação orientada a objetos e os processos ágeis, como XP (*Extreme Programming* - Programação Extrema), na qual equipes pequenas ou médiasdesenvolvem software cujos requisitos não são claros ou que mudam constantemente,obtiveram uma maior aceitação nos anos 90. Ainda nos anos90 os computadores de maior processamento tiveram uma queda acentuadade preços. A complexidade das aplicações de software cresceu mais como avanço da tecnologia, a Internet e os dispositivos móveis permitiram queo software se tornasse cada vez mais disponível. A partir de 2000 surgiu o *código* gerenciado e plataformas interpretadas, tais como, Java, NET, Ruby,Python e PHP, proporcionaram facilidades na criação de aplicações de software. A Partir, da aí o termo Engenharia de Software é amplamente utilizado para se referir a área da informática cujo objetivo é descrever, desenvolver e promover a manutenção em sistemas de software, ou seja, programas de computador, aplicando tecnologias e boas práticas consideradas vindas das disciplinas de ciência da computação e gerência de projetos, entre outras, levando, assim, a produtividade e qualidade. A Engenharia de software se preocupa com os aspectos da construção de um sistema de software, desde o início de sua especificação até a manutenção, após ter sido posto em operação. Os engenheiros fazem uso de teorias, métodos e ferramentas para tornar os produtos úteis, procurando soluções para problemas, mesmo que não existam teorias aplicáveis e métodos de auxílio. Não somente os aspectostécnicos são importantes, as atividades de gerenciamento de projetos e desenvolvimento de ferramentas, os métodos e as teorias de apoio à produçãode software também são questões importantes dentro da engenharia de software**.**

# 

# 3 Papel do Software

Quanto mais os produtos de software são criados, mais eles facilitam a vida das pessoas. Terminais de autoatendimento , e-mail e WhatsApp, aparelhos eletroeletrônicos (televisores e fornos de micro-ondas) podem receber as mais diversas programações realizadas pelos usuários através de um controle remoto ou em um painel digital, carros que realizam várias verificações no sistema elétrico e de injeção de combustível e emitem sinais alertando sobre o não uso do cinto de segurança, aviões capazes de realizar muitas de suas operações, praticamente, sem a intervenção humana, aplicações de software que interpretam e leem textos auxiliando portadores de deficiência visual são apenas alguns exemplos de como o software está inserido na vida cotidiana. Quanto mais a tecnologia evolui, quanto mais a humanidade se globaliza, quanto mais atividades as pessoas procuram realizar, maior é o impacto do software na sociedade e na cultura. O software exerce papel importante na vida do ser humano ajudando, auxiliando, conduzindo e executando as mais diversas tarefas com maior eficiência e eficácia. Assim, os profissionais de desenvolvimento de software procuram criar tecnologias que permitem a construção de softwares com maior produtividade e qualidade. O software atualmente pode exercer dois papéis, ele pode ser um produto ou ele é um meio para a entrega do produto. Como produto, é possível ter acesso ao potencial do computador no uso de alguma ferramenta ou rede de computadores. Independente da capacidade computacional do equipamento em que o software está instalado, seja em um telefone celular, em um equipamento doméstico, nos carros, nos aviões, em um computador pessoal ou em um computador de grande porte, o software age sobre a informação, automatizando, gerando, adquirindo, modificando, exibindo ou transmitindo essa informação. Como veículo de entrega de um produto, o software é utilizado no controle do computador (sistemas operacionais), na comunicação (redes de computadores) e na criação de outros softwares (ferramentas e ambientes de desenvolvimento de software).

# 4 Características do Software

Para entender as principais características de um software, é necessário fazer um paralelo como quando um produto qualquer é construído várias etapas se seguem: análise, projeto, desenvolvimento e teste. Uma vez que estas etapas tenham sido seguidas um produto passa a existir na forma física. Como exemplo, tem-se a construção de uma casa, uma vez que o engenheiro civil tenha conhecimento do local no qual a casa será erguida e os desejos do futuro morador, ele realiza uma análise para saber quais os materiais mais adequados para a construção, então constrói um esboço da casa na forma de projeto. Uma vez que o projeto esteja pronto, o mestre de obras é encarregado de orientar os pedreiros na construção da casa, ao mesmo tempo em que realiza verificações e testes para confirmar que a construção atende aos desejos do morador e as especificações do projeto. Assim que a construção é terminada, tem-se um produto físico, algo palpável. Quando o assunto é software, a ideia de construção é semelhante a qualquer outro produto, porém, com diferentes abordagens. O software faz parte de um sistema lógico e não de um sistema físico, por conta disso é necessário um processo de construção de software.

Principais características de um software.

* Surge de um processo de desenvolvimento e não manufatura.
* Não apresenta desgaste com passar do tempo.
* Pode surgir a partir de componentes pré-existentes, no entanto a maior parte das aplicações de software ainda é desenvolvida sob demanda, de acordo com as necessidades do usuário.

# 5 Processo de Software, Infraestrutura de Processo

O processo de software é um esboço, uma estrutura inicial para as tarefas necessárias à construção de um software de alta qualidade. Sendo assim, cabe a seguinte indagação: processo de software é sinônimo de engenharia de software? Para esta questão há duas respostas, dependendo de como a ela é percebida. Tanto o processo de software quanto a engenharia de software descrevem como a construção de um software deve ser realizada. Porém, a engenharia de software vai além, ela acrescenta tecnologias (métodos técnicos e ferramentas automatizadas) que formam o processo. Os métodos abrangem um conjunto de tarefas necessárias à construção de um software. Estas tarefas são análise de requisitos, projeto, construção de programas, testes e validações e manutenção. As ferramentas, que podem ser automatizadas ou semiautomatizadas, apoiam o processo e os métodos. Se a informação criada por uma ferramenta puder ser usada por outra se diz que a engenharia de software é auxiliada pelo computador ou, do inglês, CASE (*Computer Aided Software Engineering*). Uma ferramenta CASE pode ser apenas um software que inclua editores de projeto e dicionários de dados, por exemplo, ou uma combinação de software, hardware e um repositório de informações a respeito de análise, projeto, construção e testes. O processo de software é caracterizado como tendo uma estrutura comum de processo, aplicado a projetos de qualquer porte e complexidade; um conjunto de tarefas, que permite a aplicação das atividades de estrutura às características do projeto e às necessidades da equipe de projeto; e, por fim, atividades guarda-chuva, que são independentes de qualquer atividade de estrutura e acontecem durante todo o projeto.

# 6 Ciclo de vida

O ser humano passa por um processo durante existência, um ciclo de vidaque se inicia com o nascimento, passa pela adolescência, pela juventude eentra na fase adulta e, finalmente, a velhice nos alcança, deixando claro queo fim está se aproximando. Isto acontece com tudo e todos. Os seres vivosnascem, crescem e morrem; os seres inanimados (sem vida) são criados (pelanatureza ou pelo homem), utilizados, se desgastam e se tornam inúteis. Umditado bem conhecido e que expressa essa verdade diz: “Na vida a únicacerteza é a morte”. O desenvolvimento de software passa por processo semelhante. Ele é concebido, projetado, codificado e passa a ser utilizado para o fim a que se propõe. Ao contrário do ser humano, um software não se desgasta, mas pode sofrer modificações para que possa continuar atendendo aos seus usuários. Um software passa por etapas distintas que podem ser resumidas em: *análise, projeto, codificação, teste e uso.* O ciclo de vida é utilizado para descrever o que acontece em cada uma destas etapas. O modelo de ciclo de vida é importante, pois com ele é possível determinar as etapas existentes, a ordem das atividades a serem desenvolvidas e quais os critérios a serem adotados para transição entre as etapas. Os modelos de ciclo de vida vão desde o simples codificar e consertar (do inglês *Code-and-fix*) até modelos mais complexos, como o espiral. Cada modelo possui características que o torna útil para determinada situação ou projeto de software, bem como, desvantagens inerentes ao próprio modelo. A seguir, tem-se uma descrição de cada um dos segue alguns modelos de ciclo de vida mais aplicados no desenvolvimento de aplicações de software, pois se constituem em modelos para sistemas práticos.

# 6.1 Codificar e consertar (*Code-and-fix*)

Nos primórdios do desenvolvimento de software o processo se concentrava nos programas e, normalmente, era realizada por uma pequena equipe ou, muitas vezes, por uma única pessoa. O processo é iniciado a partir de uma ideia geral do que se pretende construir e utiliza uma combinação de projeto e programação informal. Este modelo também ficou conhecido como desenvolvimento artesanal ou *ad-hoc* e consistia em realizar a programação e, à medida que as inconsistências e erros eram encontrados, promoviam-se os acertos, sendo estes passos repetidos até que o projeto fosse concluído. Um dos modelos mais antigos e ainda utilizados, também conhecido como Modelo Cascata ou *Waterfall*. Consiste em um conjunto de fases que são executadas sequencialmente uma após a outra. A fase seguinte somente pode ser iniciada se a fase atual for completamente concluída. Por isso, ao final de cada fase uma revisão é realizada para se saber se a fase foi completada a contendo. Se houver alguma falha na fase atual, o projeto permanecerá nela até que os problemas sejam resolvidos.

**Análise e definição de requisitos:** nesta etapa, diversas entrevistas são de funcionalidades e suas restrições.

**Projeto de sistemas e de software:** o processo de projeto de sistemas estabelece uma arquitetura do sistema geral seja de hardware ou de software, enquanto o projeto de software se preocupa com a identificação e a descrição dos requisitos fundamentais do sistema de software

**Implementação e testes de unidades:** uma vez que o projeto de software seja codificado, ele passe a ser compreendido por um conjunto de programas ou unidades de código. Os testes têm o objetivo de certificar cada unidade de código de acordo com sua especificação;

**Integração e teste de sistemas:** esta é a etapa em que as unidades de código são integradas, o sistema é testado como completo e finalizado para garantir que ele atenda aos requisitos levantados junto aos usuários.

**Operação e manutenção:** ao término da fase de testes o sistema é implantado e posto em operação. Esta etapa pode ser longa, pois ela envolve em descobrir e corrigir falhas que não foram verificadas nas etapas anteriores.

**6.2 Protótipos**

Protótipos são divididos em 4 (quatro) categorias: ilustrativo, no qual somente as telas são projetadas dando ao usuário uma ideia de como será a interface (“cara” do programa) final; simulado, que simplesmente simula o acesso ao sistema de persistência, possivelmente uma base de dados; funcional, neste protótipo são implementados apenas um subconjunto de todas as funcionalidades requeridas pelo software; evolucionário, este protótipo começa com uma versão menor do software que crescerá a medida que as próximas etapas forem concluídas. O processo inicia-se com a definição dos requisitos, sendo que os objetivos do software são definidos e identificados, bem como identificadas aquelas áreas que merecem um maior refinamento. Uma vez levantados os requisitos, um projeto inicial é construído e apresentado ao usuário, que realiza a avaliação do protótipo. Desta forma, o usuário interage com o processo de desenvolvimento ajudando a refinar os requisitos e adquirindo uma maior confiança no software. À medida que os protótipos são construídos, uma base de código é produzida. Assim, os próximos protótipos podem reusar o código já escrito. A reutilização de software é um conceito muito importante e se baseia não somente na reutilização de código escrito anteriormente, mas, também, na experiência adquirida na construção da base de código. Em Sommerville (2003), verifica-se a existência de 3 (três) problemas do ponto de vista da engenharia e de gerenciamento para este modelo: O processo não é visível:se o software é construído muito rapidamente não há como medir o progresso do desenvolvimento os sistemas são frequentemente mal estruturados acrescentar modificações pode se tornar difícil e de alto custo à medida que o software cresce a exigência de ferramentas e técnicas especiais:ferramentas e técnicas que auxiliam na produtividade permitindo um desenvolvimento mais rápido podem não ser compatíveis com outras ferramentas ou técnicas, bem como o número de pessoas especializadas com conhecimento suficiente nestas ferramentas ou técnicas podem não ser suficientes.

# 7 Linguagem e Técnicas de Programação

# 7.1 Origem Da Linguagem C

A linguagem C foi criada por Dennis M Ritchie e Ken Thompson no laboratório Bell em 1972, que foi baseada na linguagem B de Thompson que era uma evolução da antiga linguagem BCPL. B foi nomeada como a primeira linguagem e C foi a segunda, é muito provável que a próxima linguagem se chamará P.C é uma linguagem vitoriosa na programação de qualquer tipo de sistema, um exemplo disso, é o sistema operacional UNIX, que foi baseado nessa linguagem. A linguagem C ganhou grande notoriedade, por ser uma linguagem muito poderosa, portátil e flexível. Esse tipo de programação foi criado para que o usuário possa planejar programas estruturados e modelares, além de trazer uma execução rápida e compacta. A linguagem C é considerada amiga do programador, pois eles podem ser desenvolvidos em partes separadas por pessoas distintas e depois unidas em um produto final. (MIZRAHI, 2008, p.2) Há cinco tipos básicos de dados em C: caractere, inteiro, ponto flutuante, ponto flutuante de precisão dupla e sem valor (char, Int, float, double e Void). Todos os outros tipos de dados em C são baseados em um desses tipos mencionados. O tamanho e a faixa variam de acordo com o tipo de processador e com implementação do compilador C. (SCHILDT, 1996, p.17) Os tipos char e Int armazenam números inteiros. O tipo chair possui o tamanho de um byte. O tipo Int tem o tamanho da palavra da máquina, ou seja, em ambiente de 16 bits, o tipo Int terá 2 bytes de tamanho e em ambientes 32 bits, o tipo Int terá 4 bytes de tamanho. (MIZRAHI, 2008, p.16)

# 7.2 Variáveis em ponto flutuante – float, double e Void

As faixas dos tipos float e double é dada em dígitos de precisão. Essas grandezas dependem do método usado para representar os números em pontos flutuantes. Qualquer que seja o método, o número acaba sendo muito grande. O padrão ANSI especifica que a faixa mínima de um ponto flutuante é de 1E-37 a 1E+27. Já o tipo Void declara uma função que não retorna nenhum valor ou cria pontos genéricos. (SCHILDT, 1996, p.17)

Figura 1 – Tabela explicando sobre as variáveis

Tabela

Descrição gerada automaticamente

Fonte: SCHILDT, H. 1996.

# 7.3 Variáveis

Uma variável é uma posição nomeada de memórias, que é usada para guardar um valor que pode ser alterado pelo sistema. Todas as variáveis C precisam ser declaradas antes de serem utilizadas. A forma geral de uma declaração é:

*tipo lista\_de\_variáveis*;

Aqui *tipo* deve ser um tipo de dado válido em C mais qualquer outros modificadores; e *lista\_de\_variáveis,* pode consistir em um ou mais nomes de identificadores separados por uma vírgula. (SCHILDT, 1996, p.20) De acordo com MIZRAHI (2008), as variáveis são um aspecto fundamental para qualquer linguagem de computador. Uma variável em C é um espaço de memória reservado para armazenar um certo tipo de dado e tendo um nome para representá-lo. Uma variável é um espaço de memória que pode ter, a cada tempo, valores diferentes.

# 7.4 Características da Linguagem C

Essa linguagem foi desenvolvida por Dennis Ritchie em 1972, e é considerada uma linguagem de programação procedural. Foi desenvolvida como uma linguagem de programação de sistema para escrever um sistema operacional. Os principais recursos incluem acesso de baixo nível a memória, estilo limpo e um conjunto simples de palavras-chaves. (ACERVOLIMA, s.d.)

Principais recursos são:

* Portabilidade
* Modularidade
* Recursos de baixo nível
* Simplicidade

A seguir iremos detalhar melhor cada um destes tópicos.

**Portabilidade**: pode ser compilada em diferentes arquiteturas, seja de software ou hardware. Isto pode parecer algo simples, mas há linguagens que não funcionam assim.

**Modularidade**: Um programa C é dividido em vários blocos de programação diferentes, neste caso, o que foi escrito uma função, não irá afetar as outras.

**Baixo nível:** Linguagem eficiente, com muitos recursos para controlar a memória da máquina.

**Simplicidade**: Programação fácil com regras estabelecidas, o compilador avisa sempre que ocorrer um erro de sintaxe. (CANALTECH, s.d.)

# 8 Desenvolvimento Do Sistema

# 8.1 Construção do projeto

A seguir iremos apresentar como realizamos o desenvolvimento do nosso projeto no Code Blocks utilizando a linguagem C.

**Apresentando a tela de login**

Figura 2 – Apresentando a tela de login

**Interface gráfica do usuário, Texto

Descrição gerada automaticamente**

Fonte: Elaborado pelo autor (2022)

Nesta imagem podemos ver a nossa tela de login sendo efetuado com sucesso, solicitando login e senha. E após isso, já podemos inserir o nome do paciente.

# 8.2 Tela de cadastro e informações do diagnóstico

Figura 3 – Tela de cadastro

**Interface gráfica do usuário, Texto, Aplicativo

Descrição gerada automaticamente**

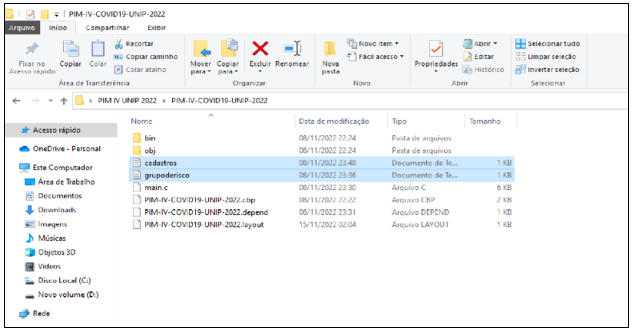
Fonte: Elaborado pelo autor (2022)

Nesta tela podemos visualizar as informações que serão necessárias para o cadastro do paciente. Primeiramente é solicitado os dados do endereço, data do diagnóstico e comorbidades, para que o cadastro seja concluído com sucesso. Assim que é informado que o paciente possui alguma doença, automaticamente o sistema identifica o indivíduo como pertencente ao grupo de risco.

**Enviando as informações para a secretária da saúde**

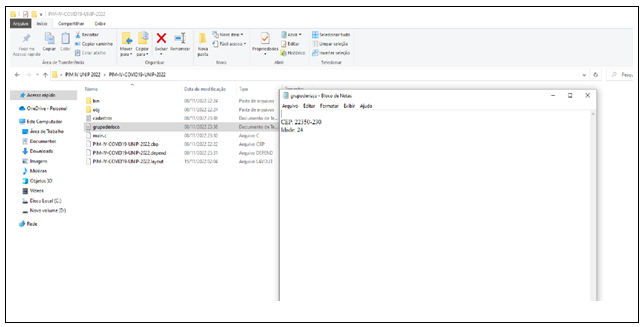
Ao preencher a o cadastro o programa avalia se o paciente está em um grupo de risco, caso sim, o sistema elabora um arquivo de texto com as informações do indivíduo e salva na pasta em que está o programa. Exemplo:

Figura 4 - Demonstrando o Arquivo



Fonte: Elaborado pelo autor (2022)

Figura 5 – Demonstrando o Arquivo



Fonte: Elaborado pelo autor (2022)

# 8.3 Comprovando que o nosso sistema funciona e executa os dados

Nas imagens abaixo, podemos constatar que o funcionamento do nosso sistema acontece perfeitamente e sem falhas técnicas.

Figura 6 – funcionamento do sistema

Interface gráfica do usuário, Texto, Aplicativo, Email

Descrição gerada automaticamente

Fonte: Elaborado pelo autor (2022)

Figura 7 – funcionamento do sistema

Interface gráfica do usuário, Texto, Aplicativo

Descrição gerada automaticamente

Fonte: Elaborado pelo autor (2022)

# 

# 8.4 Variáveis Usadas

* Char: Responsável por guardar um valor inteiro de um conjunto de caracteres.
* Int: O Int é uma variável usada para armazenar números inteiros.
* Time\_t: Variável usada para valores como data e hora em uma única variável.

# 8.5 Funções Usadas

* Printf: Em uma biblioteca de rotinas da Linguagem C, temos a função Printf responsável em nos mostrar os valores na tela de qualquer tipo de dado.
* Scanf: Vem do arquivo cabeçalho stdio.h, que permite a leitura de dados.
* Void: comando que nos permite fazer funções que não retornam nada e que não tem parâmetros.
* Struct: Essa função agrupa variáveis sob um mesmo tipo de dado e é conhecida como registros, armazenar dados de uma mesma entidade, permite que seja feito com uma única variável.

# 8.6 Estruturas Usadas

* If: É uma estrutura de decisão, ou seja, ela soma dois números inteiros e mostra o resultado na tela, se o resultado for maior que 20, por exemplo, ela nos informa na tela.
* Else: dentro da instrução If, o código que estiver dentro dos parênteses for verdadeiro, ele será executado, mas se o que estiver dentro do parêntese do If for falso, o código dentro das chaves da instrução Else será executado no lugar, logo o resultado será falso.

# 8.7 Bibliotecas Usadas no Projeto

* #include <stdio.h>
* #include <stdio.h>
* #include <string.h>
* #include <time.h>
* #define SIZE 200

# 

# 9 Manual do Usuário

Para encontrar o sistema desenvolvido, basta o usuário procurar pelo arquivo **PIM-IV-COVID19-UNIP-2022**, e clicar em cima do arquivo duas vezes. Assim ele irá iniciar na tela de login.

**Login**

Na tela de login irá aparecer os campos vazios solicitando login e senha, e precisam ser preenchidos com os dados abaixo:

Login: Gabriel Maia

Senha: 1234

Após preencher com essas informações, o sistema pode ser acessado.

**Consultar cadastro**

Acessando o sistema, podemos visualizar os campos com data de nascimento, telefone, endereço completo com CEP, e-mail, data do diagnóstico e se há comorbidades.

**Sair**

Para finalizar o programa basta que apertamos a tecla Enter do teclado e o sistema se fechará sozinho.

# 10 Conclusão.

Este Projeto Integrado Multidisciplinar teve o propósito de demonstrar alguns pontos importantes na concepção e desenvolvimento de software por meio de linguagem de programação, no caso a Linguagem C. Nos dias atuais, os softwares estão cada vez mais importantes e necessário nas vidas das pessoas. Métodos, técnicas e ferramentas de construção são fundamentais para elevar sua qualidade e consequentemente deixar o cliente final ou usuário satisfeito. Softwares ou programas são usados com a proposta de solução de problemas tanto na parte de interface do usuário com o computador ou um determinado equipamento como também no dia a dia de uma maneira geral. No caso deste Projeto os conhecimentos envolvidos tanto na engenharia de software como na linguagem de programação foram utilizados para resolver um problema a falta de um sistema para armazenar as informações dos pacientes, estratégias tomadas diante do crescimento da Pandemia de Covid -19 pois assim é possível ter controle dos pacientes de forma mais precisa e com responsabilidade e qualidade. É um ponto de vista que a tecnologia pode ajudar no bem-estar físico do ser humano. Caro leitor(a), este Projeto Integrado Multidisciplinar teve o objetivo de demonstrar de maneira acadêmica os aspectos envolvidos no desenvolvimento de um produto de software, o conhecimento envolvido necessário em linguagem de programação e engenharia de software de maneira resumida e objetiva foram aplicadas várias horas de trabalho árduo visando agregar conhecimento e qualidade ao material, por conta disso um olhar analítico construtivo é o ideal na leitura do material.

# 

# 11 Referências

**DEMARCO**, tom. análise estruturada e especificada de sistemas. rio de janeiro: campus, 1989.

**FREITAS** Romualdo Rubens. análise e projeto de software rede E-tec -brasil.

**FURLAN,** josé Davi. modelagem de objetos através da uml – the Unified Modeling Language. São Paulo

**LARMAN, CRAIG**. utilizando uml e padrões – uma introdução a análise e ao projeto orientado a objeto. 3 ed. porto alegre: Bookman, 2007.

**MIZRAHI,** v, v. treinamento em linguagem c. são Paulo: Pearson Prentice hall, 2008.

**SOMMERVILLE**, Ian. engenharia de software. 6 ed. são Paulo: Addison Wesley, 2003.

**SCHILDT,** h. c completo e total. são Paulo: Makron books, 1996.

**SEM AUTOR**: características da linguagem de programação c. acervo lima, 2022. disponível em: <https://acervolima.com/caracteristicas-da-linguagem-de-programacao-c/>. acesso em 09/11/2022 às 20h45

**SEM AUTOR**: C: a linguagem de programação que está em tudo o que você usa. canal tech, 2014. disponível em <https://canaltech.com.br/software/c-a-linguagem-de-programacao-que-esta-em-tudo-o-que-voce-usa-19512/>. acesso em 09/11/2022 às 20h48