**UNIVERSIDADE PAULISTA – UNIP EaD**

**Projeto Integrado Multidisciplinar**

**Curso Superior de Tecnologia em**

**Análise e Desenvolvimento de Sistemas**

**ANA JÚLIA MONTEIRO FERREIRA – 2239285**

**CÉSAR AUGUSTO FERREIRA VENÂNCIO – 2206637**

**HUDSON EDUARDO DE AGUIAR – 2220950**

**LEANDRO DE MOURA FÉ – 2228701**

**LUÍS FABIANO SOARES SMARRA – 2211916**

**MÁRCIO GONÇALVES DA SILVA - 2217479**

**ELABORAÇÃO DE UM SISTEMA DE CADASTRO DE PACIENTES COM COVID-19 COM A LINGUAGEM C**

Santo André III

**2022**

**ANA JÚLIA MONTEIRO FERREIRA – 2239285**

**CÉSAR AUGUSTO FERREIRA VENÂNCIO – 2206637**

**HUDSON EDUARDO DE AGUIAR – 2220950**

**LEANDRO DE MOURA FÉ – 2228701**

**LUÍS FABIANO SOARES SMARRA – 2211916**

**MÁRCIO GONÇALVES DA SILVA - 2217479**

ELABORAÇÃO DE UM SISTEMA DE CADASTRO DE PACIENTES COM COVID-19 COM A LINGUAGEM C

Projeto Integrado Multidisciplinar em

Análise e Desenvolvimento de Projetos

Projeto Integrado Multidisciplinar para obtenção do título de tecnólogo em Análise e Desenvolvimento de Sistemas, apresentado à Universidade Paulista – UNIP EaD.

 Orientador(a): Profa. Dra. Vanessa Lessa

Santo André III

**2022**

Dedicamos este trabalho acadêmico aos nossos familiares e a todas as pessoas que ainda acreditam que o conhecimento é uma das formas de mudar o mundo!

AGRADECIMENTOS

Agradecemos a Deus, acima de tudo, pelo dom da vida, pela sabedoria e tudo que tem nos proporcionado a cada momento em nossa vida e em nossa trajetória. Agradecemos também a nossos familiares pelo apoio incansável, aos professores e todo corpo docente.

"Quanto mais avança a tecnologia, mais respeitado será o Ser que se mantiver Humano. Uma máquina consegue fazer o trabalho de 50 homens ordinários. Nenhuma máquina consegue fazer o trabalho de um homem extraordinário."

(Elbert Hubbard, escritor).

#### RESUMO

O presente Projeto Integrado Multidisciplinar tem como tema apresentar um sistema em linguagem C que será utilizado pelos hospitais para cadastrar os pacientes que forem diagnosticados com Covid-19 e que carecem de um acompanhamento e monitoramento para que essa informação possa ser enviada para a central da Secretaria da Saúde. Especificamente, busca-se: Desenvolver e aplicar os conhecimentos adquiridos nas disciplinas de Linguagem e Técnicas de Programação e Engenharia de Software I; Fomentar o hábito de executar atividades envolvendo múltiplas disciplinas; Desenvolver a capacidade de identificar necessidades e propor soluções técnicas; Elencar, argumentar e justificar sobre metodologias referentes ao desenvolvimento de um sistema para auxiliar a área da saúde; Aplicar as normas ABNT para a produção de trabalhos acadêmicos. O projeto foi construído utilizando o modelo híbrido: cascata na implementação de código em linguagem C com a ferramenta Code::Blocks e o incremental durante o processo com o auxílio da ferramenta Git para o versionamento a cada desenvolvimento de novas funcionalidades. Um manual do usuário que implementa instruções de instalação, compilação, execução, entre outras, está presente, explicando-as passo a passo. Resultou-se em um software simples, entretanto que servirá para múltiplas funcionalidades na área da saúde, como por exemplo, a coleta de dados estatísticos dos pacientes com covid por região.

Palavras-chave: Covid-19; programação; linguagem C.

#### ABSTRACT

This Multidisciplinary Integrated Project has the theme of presenting a system in C language that will be used by hospitals to register patients who are diagnosed with Covid-19 and who need follow-up and monitoring so that this information can be sent to the central Secretariat of the Health. Specifically, it seeks to: develop and apply the knowledge acquired in the disciplines Language and Techniques of Programming and Software Engineering I; Foster the habit of carrying out activities involving multiple disciplines; Develop the ability to identify needs and propose technical solutions; List, argue and justify methodologies related to the development of a system to assist the health area; Apply ABNT standards for the production of academic papers. The project has been built using the hybrid model: cascading code implementation in C language with the Code::Blocks tool and incremental during the process with the help of the Git tool for versioning with each development of new features. An user manual that implements installation, compilation, execution instructions, among others, is present, explaining them step by step. It resulted in a simple software, however that will serve for multiple functionalities in the health area, such as, for example, the collection of statistical data of patients with covid by region.

Key-words: Covid-19; programming; C language.

#### LISTA DE FIGURAS

[Figura 1 - Processo da elaboração do modelo cascata 10](#_Toc120137328)

[Figura 2 - Forma com que a função printf se relaciona ao arquivo-cabeçalho stdio.h 12](#_Toc120137329)

[Figura 3 - Fluxograma do projeto 16](#_Toc120137330)

[Figura 4 - Como efetuar o download da ferramenta Code::Blocks 19](#_Toc120137331)

[Figura 5 - Tela de associação do Code::Blocks com as extensões das linguagens C e C++ 20](#_Toc120137332)

[Figura 6 - Como abrir um projeto existente no Code::Blocks 21](#_Toc120137333)

[Figura 7 - Como compilar o programa 21](#_Toc120137334)

[Figura 8 - Como compilar e executar o programa 21](#_Toc120137335)

[Figura 9 - Exemplo de falha de validação do sistema 22](#_Toc120137336)

[Figura 10 - Desconformidade na inserção do nome do paciente 23](#_Toc120137337)

[Figura 11 - Possíveis erros de validação do campo telefone 24](#_Toc120137338)

[Figura 12 - Confirmação dos dados inseridos 27](#_Toc120137339)

[Figura 13 - Pergunta de reinicio de ciclo 28](#_Toc120137340)

**SUMARIO**

[RESUMO 5](#_Toc120365607)

[ABSTRACT 6](#_Toc120365608)

[LISTA DE FIGURAS 7](#_Toc120365609)

[1. INTRODUÇÃO 9](#_Toc120365610)

[2. MODELO DE CICLO DE VIDA DO SOFTWARE 10](#_Toc120365611)

[3. LINGUAGEM C 12](#_Toc120365612)

[3.1. Funções e bibliotecas 12](#_Toc120365613)

[3.2. Estruturas de Repetição 12](#_Toc120365614)

[4. Levantamento de requisitos 14](#_Toc120365615)

[4.1. Requisitos funcionais e não funcionais do programa de registro de pacientes de Covid19 14](#_Toc120365616)

[4.1.1. Metodologia de gerenciamento de projeto 15](#_Toc120365617)

[5. FLUXOGRAMA 16](#_Toc120365618)

[5.1. Ferramentas 17](#_Toc120365619)

[5.1.1. CodeBlocks 17](#_Toc120365620)

[5.1.2. Git 17](#_Toc120365621)

[6. MANUAL DO USUÁRIO 19](#_Toc120365622)

[6.1. Instalação do Code::Blocks 19](#_Toc120365623)

[6.2. Abrindo e compilando o programa de registro de pacientes com covid19 no Code::Blocks. 20](#_Toc120365624)

[6.3. Funcionamento Do Programa 21](#_Toc120365625)

[6.4. Resgatando os Arquivos Salvos 28](#_Toc120365626)

[7. CONSIDERAÇÕES FINAIS 29](#_Toc120365627)

[REFERÊNCIAS 30](#_Toc120365628)

# INTRODUÇÃO

O presente Projeto Integrado Multidisciplinar tem como tema apresentar um sistema em Linguagem C que será utilizado pelos hospitais para cadastrar os pacientes que forem diagnosticados com Covid-19 e que carecem de um acompanhamento e monitoramento para que essa informação possa ser enviada para a central da Secretaria da Saúde. Especificamente, busca-se:

* Desenvolver e aplicar os conhecimentos adquiridos nas disciplinas de Linguagem e Técnicas de Programação e Engenharia de Software I;
* Fomentar o hábito de executar atividades envolvendo múltiplas disciplinas;
* Desenvolver a capacidade de identificar necessidades e propor soluções técnicas;
* Elencar, argumentar e justificar sobre metodologias referentes ao desenvolvimento de um sistema para auxiliar a área da saúde;
* Aplicar as normas ABNT para a produção de trabalhos acadêmicos.

A metodologia utilizada neste projeto será, portanto, a linguagem em programação em C. Este sistema que será desenvolvido gera arquivos de texto com informações pessoais dos pacientes como: existência ou não de comorbidades e dados pessoais como nome, localização e e-mail. Os profissionais acessam através da realização de *login* e senha, onde as informações serão salvas e podem ser acessadas a qualquer momento, além disso, ao final do desenvolvimento o programa será capaz de calcular a idade e verificar se o paciente possui alguma comorbidade e se pertence ao grupo de risco (maiores de 65 anos) para que a informação seja enviada para a central da Secretaria da Saúde da cidade.

O projeto segue com a divisão em capítulos, onde: o capítulo 02 trata do modelo de ciclo de vida do software, o capítulo 03 da linguagem de programação em C para desenvolver o tema proposto, o capítulo 04 do levantamento de requisitos, o capítulo 05 do fluxograma, o capítulo 06 do manual do usuário, com as considerações finais do trabalho no capítulo 07. Todo o decorrer deste documento está conforme o estabelecido pela norma da ABNT.

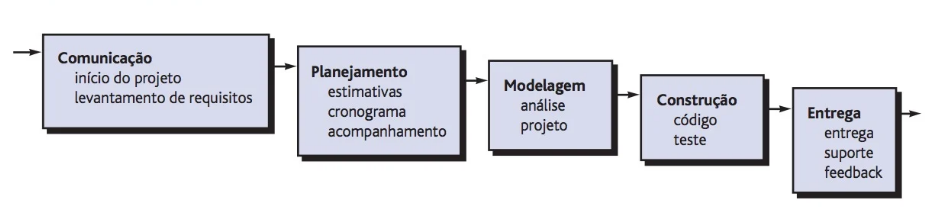
# **MODELO DE CICLO DE VIDA DO SOFTWARE**

Os softwares possuem um ciclo de vida que são compostos por etapas, sendo elas: a percepção de uma necessidade, o desenvolvimento, a operação, e seu aprimoramento ou retirada de operação (PAULA FILHO, 2019). Para o seu desenvolvimento existem vários modelos, sendo o mais clássico o cascata.

Quando os requisitos de um problema são bem compreendidos, definidos, e razoavelmente estáveis, o modelo cascata pode se enquadrar como um dos modelos mais adequados. As atividades de desenvolvimento são realizadas de forma sistemática, cada fase deve ser completada antes que inicie a próxima, não havendo sobreposições, o processo de desenvolvimento é tratado, portanto, como sequencial (PRESSMAN, 2016).

Como é observado na Figura 1o processo da elaboração do modelo cascata segue uma metodologia sequencial, composto por etapas que são: comunicação, planejamento, modelagem, construção e entrega.

Figura - Processo da elaboração do modelo cascata



Fonte: (PRESSMAN, 2016).

Existem algumas variantes do modelo cascata tradicional, que procuram sanar algum dos seus problemas, um deles é o modelo incremental.

Pressman (2016) também afirma que o modelo incremental combina elementos dos fluxos de processos lineares e paralelos. O desenvolvimento incremental é baseado na ideia de formalizar um programa inicial, apresentá-lo ao cliente, e ir incrementando modificações para atender aos requisitos que foram levantados, até que o sistema completo esteja pronto. Aplicando sequencia linear de forma escalonada.

Essa abordagem é a mais comum utilizada atualmente para desenvolver sistemas e aplicativos em geral. E é melhor que a abordagem em cascata, por resolver problemas do usuário mais rapidamente, além da quantidade de documentação ser menor.

# LINGUAGEM C

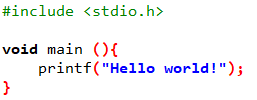
## Funções e bibliotecas

*Costa (2017) define que a standard library* é um conjunto de bibliotecas para o desenvolvimento de programas em linguagem C, suas principais tarefas são: executar manipulação de *strings*, entrada e saída *(input/output*) e gestão de memória.

Adicionalmente às funções *standard*, o programador pode definir suas próprias funções para desempenhar tarefas específicas que contém argumentos de entrada e retorno de saída. A chamada das funções é realizada através da chamada de seu nome seguido por parênteses contendo seus parâmetros.

O programa C executa a função primária que é nomeada de *main()*, ela é o ponto de partida para executar o programa, pois irá controlar e direcionar as outras funções do programa, então implementa as funções antes definidas (COSTA, 2017).

Figura - Forma com que a função printf se relaciona ao arquivo-cabeçalho stdio.h



Fonte: Elaborado pelos autores (2022)

As bibliotecas-padrão utilizadas para o desenvolvimento do *software* foram:

* *Conio.h*: utilizada por disponibilizar as funções getch e putch;
* *Math.h*: dispõe as funções de cálculo exponencial;
* *Stdio.h*: insere as funções de entrada e saída de dados;
* *String.h*: viabiliza a manipulação de cadeia de caracteres.

## Estruturas de Repetição

A instrução *while* é usada quando não se sabe quantas vezes um determinado bloco de instruções precisa ser repetido. Com ele, a execução das instruções vai continuar até que uma condição seja falsa. (MANZANO, 2013).

O *do/while* tem quase o mesmo funcionamento que o *while*, a diferença é que com o uso dele teremos os comandos executados ao menos uma única vez (MANZANO, 2013).

A estrutura de repetição *for* é utilizada quando sabemos a quantidade de vezes que um bloco de códigos de ser executado pois a sintaxe é composta de três expressões separadas de ponto e vírgula que determinam a inicialização de uma variável de controle, condição de parada e incremento da variável. (MANZANO, 2013).

# Levantamento de requisitos

Conforme os conceitos de engenharia de software aplicados a comunicação, os autores do trabalho realizaram o levantamento dos requisitos funcionais e não funcionais para que a elaboração do programa fosse precisa em relação as orientações fornecidas.

Conforme Paula Filho (2000, p. 13), os requisitos funcionais são as funcionalidades do sistema, a forma com que o sistema responde as interações com o usuário, já os não funcionais qualificam/ restringe alguns aspectos do funcionamento do *software*.

## Requisitos funcionais e não funcionais do programa de registro de pacientes de Covid19

As orientações fornecidas para o desenvolvimento desse trabalho propuseram a elaboração de um programa criado em linguagem C que será utilizado pela área da saúde para realizar o cadastro de pacientes diagnosticados com COVID19 para acompanhamento. O sistema deve ser capaz de realizar o registro dos pacientes em um arquivo e separar os pacientes que se caracterizam como sendo grupo de risco.

Os requisitos funcionais e não funcionais desse projeto são:

* Sistema de *login* que valide os campos de usuário e senha para garantir que o programa está executado por uma pessoa autorizada;
* Formulário com os campos nome, CPF, telefone, endereço (rua, número, bairro, cidade, estado e CEP), data de nascimento, e-mail, data do diagnóstico e comorbidades que deve ser salvo em um arquivo de texto;
* Validação e separação dos pacientes com mais de 65 anos e/ou que possuam comorbidades;
* Verificação dos dados imputados para que não seja inserido informações inválidas / não esperadas;
* Capacidade de registrar mais de um paciente por *login* feito;
* Programa desenvolvido em linguagem C através do compilador CodeBlocks;

### Metodologia de gerenciamento de projeto

Conforme Jory (2018, p.26 – 27), os métodos de gerenciamento de projetos tidos como tradicionais se caracterizam por certos atributos, como a definição do escopo antes do início do projeto, firme determinação de ciclo de vida, prazos e custos, a padronização dos processos e documentação extensiva do andamento da realização. As metodologias tradicionais não permitem a flexibilidade ou mudanças de escopo durante o projeto, já que nesse método o cliente não tem papel ativo durante.

Jory (2018, p.32 – 33) afirma que para satisfazer as necessidades de um novo momento no desenvolvimento de *software*, um grupo se reuniu para criar o Manifesto Ágil que definiu os princípios e valores que as lideranças de equipes de desenvolvimento de sistemas deveriam adotar, sem ter em conta a metodologia de ciclo de vida utilizada. Esses métodos e valores, visam a flexibilidade, a relação e comunicação entre a equipe mais humana, a colaboração com cliente, e o foco no *software* e não em sua documentação.

Para o desenvolvimento desse trabalho, optou-se por metodologia hibrida, uma vez que o escopo do projeto já estava bem definido e não haveriam mudanças por parte do cliente, mas a métrica aplicada analisar o andamento da realização foi a PDCA (*action* – agir, *plan* – planejar, *do* – fazer e *check* – checar), que geralmente é aplicada em metodologias ágeis.

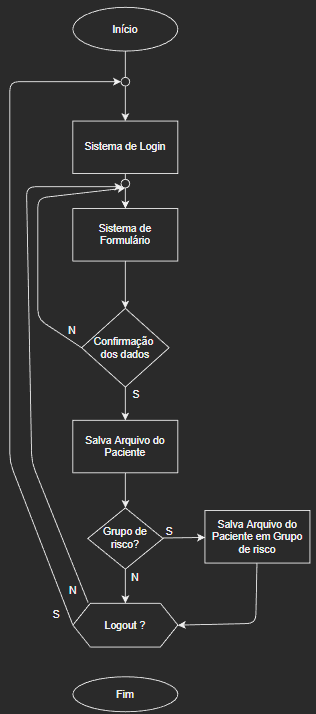
Jory (2018, p. 42) descreve e explica o funcionamento da métrica PDCA como “coletar os dados, analisar e transformar os dados em informação, aplicar melhorias no processo com base na informação obtida e monitorar os dados afetados após aplicação das melhorias”.

# FLUXOGRAMA

O fluxograma é uma das ferramentas utilizadas para representar um determinado processo, ilustrando através de símbolos e setas a sequência de execução de uma determinada atividade.

O fluxograma apresentado na Figura 3 a seguir ilustra a execução do programa desenvolvido em sistema C, com finalidade de acompanhar as pessoas infectadas pelo Corona Vírus.

Figura - Fluxograma do projeto



Fonte: Elaborado pelos autores (2022)

A representação gráfica tem início no sistema de *login*, onde os profissionais de saúde devem inserir suas credenciais para obter acesso ao sistema, após um determinado paciente receber o diagnóstico positivo para a Covid 19.

Ao obter êxito no processo anterior, o profissional de saúde é direcionado para o sistema de formulário, onde deve inserir os dados do paciente infectado e outras informações como a data do diagnóstico e se o paciente possui alguma comorbidade (doença preexistente).

Após dessa etapa o fluxograma apresenta o processo de confirmação de dados, onde o operador deverá confirmar ou não as informações inseridas até aqui, em caso de negativa o sistema retornará para o processo anterior visando a correção das informações inseridas. Já em caso de positiva, a ilustração avança o processo de salvamento em arquivo de texto com as informações do paciente para possíveis consultas posteriores.

Após esse processo o software automaticamente calcula a idade do paciente e verifica se o paciente é do grupo de risco ou não, em caso de positiva o sistema salvará o arquivo do paciente em grupo de risco e irá para tela de *logout*, já em caso de negativa o sistema apresentará diretamente a tela de *logout*, onde o atendimento será finalizado e o software retornará para o sistema de formulário, para o próximo atendimento.

## Ferramentas

Para auxiliar no desenvolvimento do *software* foram utilizadas as ferramentas: interface de desenvolvimento Code::Blocks e versionamento do projeto utilizando Git.

### CodeBlocks

Para criação do *software* de registros de pacientes com Covid 19, optou-se por utilizar o programa CodeBlocks.

Codeblocks é um ambiente de desenvolvimento integrado totalmente gratuito e extensível para linguagem C e C++, que fornece as ferramentas de apoio para a criação de *softwares*.

### Git

Para o desenvolvimento do código proposto para este projeto a ferramenta Git foi utilizada, uma ferramenta de controle de versões de software.

O versionamento do programa foi armazenado utilizando o Github. Plataforma da Microsoft desde 2018, o GitHub é uma ferramenta responsável pelo controle e organização para revisão de projetos. Criada principalmente para o setor de tecnologia, atua disponibilizando as versões dos projetos, além de permite que múltiplos usuários trabalhem simultaneamente em uma mesma solução (RAMOS, c2022).

A cada versão nova, também chamada de *commit*, é realizado um salvamento em repositório do Github com acesso a todos os colaboradores do grupo para eventuais testes e correções. Caso haja algum problema com uma nova versão, pode-se voltar para a versão anterior e assim não ter que refazer o projeto desde o começo.

# MANUAL DO USUÁRIO

Essa seção foi desenvolvida com o propósito de auxiliar o usuário antes, durante e depois da execução do programa tratado neste trabalho.

## Instalação do Code::Blocks

Inicialmente o usuário deve acessar o *site* da empresa Code::Blocks, através do URL <https://www.codeblocks.org/downloads/> e baixar a aplicação por meio da seleção de *‘Download the binary release’*, conforme Figura 4 evidência*.*

Figura - Como efetuar o download da ferramenta Code::Blocks



Fonte: Elaborado pelos autores (2022)

Existem várias versões disponibilizadas, é recomendado que a escolha seja feita de acordo com a máquina e o intuito da utilização do *software* em relação ao uso do compilador.

Primeiramente, o utilizador deve selecionar a opção de acordo com o sistema operacional (Windows*,* Linuxou MAC OS) instalado em sua máquina. Posteriormente, as opções precisam ser filtradas conforme o tipo de sistema operacional (32 ou 64 bits), no caso da plataforma Windows as cinco primeiras opções são para computadores de 64 bits e o restante para 32. Em seguida, o usuário deve se atentar se a versão selecionada já disponibiliza o compilador, que é responsável por realizar a tradução do código em linguagem C para linguagem de máquina, utilizando como exemplo o sistema Windows seria a opção ‘codeblocks-20.03mingw-setup.exe’. Qualquer um dos sites disponibilizados realizam a inicialização do *download*.

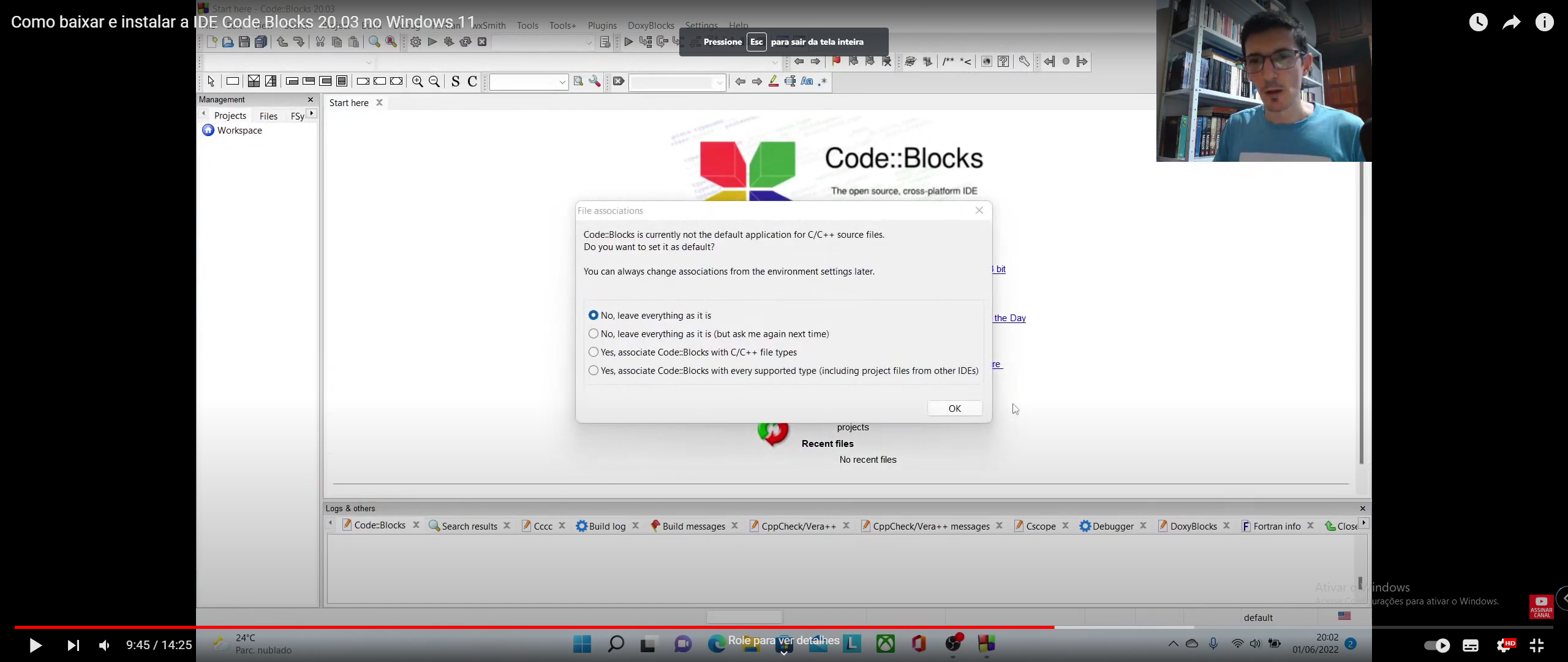
A instalação do *software* será iniciada assim que o arquivo executável for aberto, a partir desse ponto o usuário deve seguir com uma instalação convencional, seguindo as etapas conforme se apresentam.

Após a instalação, o Code::Blocks detecta automaticamente o compilador GNU GCC que foi instalado juntamente com o programa. A confirmação deve ser feita definindo o compilador como padrão, clicando em *Set as default* e em ok.

## Abrindo e compilando o programa de registro de pacientes com covid19 no Code::Blocks.

Ao inicializar o Code::Blocks pela primeira vez, o programa irá perguntará se o usuário deseja definir o *software* como a aplicação padrão para arquivos e recursos da linguagem C e C++, conforme a Figura 5. O usuário pode selecionar a terceira opção, que autoriza o programa a se tornar o padrão.

Figura - Tela de associação do Code::Blocks com as extensões das linguagens C e C++



Fonte: Elaborado pelos autores (2022)

Em seguida, o usuário deve selecionar a opção ‘*Open an existing project*’, conforme grifo na Figura 6 para que seja possível compilar o arquivo.

Figura - Como abrir um projeto existente no Code::Blocks

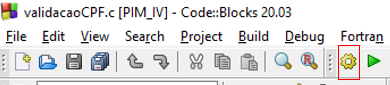


Fonte: Elaborado pelos autores (2022)

A pasta que contém os arquivos do programa deve ser acessada e o arquivo PIM\_IV do tipo ‘project file’ (.cbp) deve ser selecionado.

Para compilar o arquivo clique no ícone conforme a Figura 7 indica.

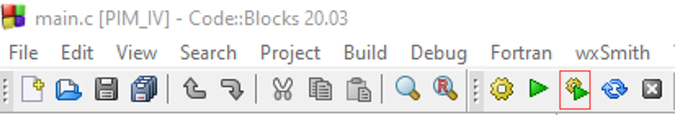
Figura - Como compilar o programa



Fonte: Elaborado pelos autores (2022)

Caso o usuário prefira compilar e já executar o programa, o ícone ressaltado pela Figura 8 deve ser selecionado.

Figura - Como compilar e executar o programa



Fonte: Elaborado pelos autores (2022)

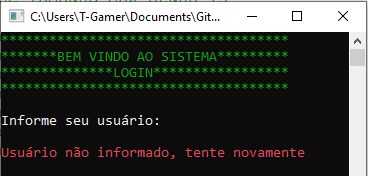
## Funcionamento Do Programa

Para a correta execução do programa de registro de pacientes com Covid19 é necessário que exista uma pasta chamada arquivos no mesmo nível da chamada para o executável localizado em bin/Release/. Para eliminar possíveis execuções em uma pasta diferente, é recomendada a execução por meio do arquivo PIM\_IV.bat.

O *software* permite que o usuário realize o *login* e preencha um formulário com os campos nome, CPF, telefone, endereço (rua, número, bairro, cidade, estado e CEP), data de nascimento, e-mail, data do diagnóstico e comorbidades que será salvo automaticamente após a confirmação das informações. O programa também verifica se o paciente está no grupo de risco e separa os arquivos em uma outra pasta para que possam ser enviados a Secretaria da Saúde da cidade.

Todos os campos a serem preenchidos pelo utilizador possuem validação, que se não forem cumpridas serão indicadas pelo texto em vermelho, conforme Figura 9.

Figura - Exemplo de falha de validação do sistema



Fonte: Elaborado pelos autores (2022)

Conforme menção anterior, primeiro o usuário realiza o *login* que é composto por um usuário e senha que devem já estar preenchidos de acordo com o correspondente no arquivo texto ‘login.txt’ localizado na pasta arquivos/entrada.

Usuário/senha definidos para utilização, respectivamente: saudesp/covid.

Abaixo estão os erros que o utilizador pode enfrentar durante o *login*:

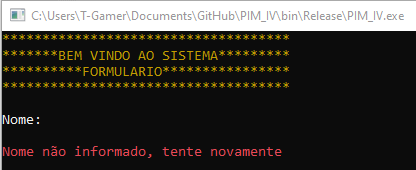
* Erro ao tentar abrir arquivo, contate o administrador: o arquivo que contém os acessos não foi encontrado;
* Usuário não informado, tente novamente: campo usuário não preenchido;
* Abortando programa: quando tentam copiar a senha;
* Usuário/Senha inválidos, tente novamente: o que foi preenchido nos campos usuário e senha não corresponde aos acessos na pasta arquivos/entrada.

O primeiro campo a ser preenchido pelo usuário é o CPF, pois com ele a aplicação já verifica se já existe um arquivo que possua a mesma informação, evitando duplicidade. O campo CPF deve ser preenchido somente com números e precisa possuir digito verificador válido. Esse campo também possui validações que caso não sejam respeitadas apresentaram um erro na tela do usuário, conforme descrição a seguir:

* CPF com tamanho inválido, tente novamente: o valor inserido não respeita a quantidade de caracteres que um CPF possui, 11 dígitos sendo 9 números e 2 verificadores;
* CPF já cadastrado, tente novamente: já existe um arquivo com o CPF inserido;
* CPF inválido, apenas números com digito verificador válido, tente novamente: CPF não possui dígitos verificadores válidos ou não foi inserido somente números.

Em seguida, o usuário deve preencher o campo 'Nome', que não pode ser deixado vazio nem ser preenchido com números, caso o dado inserido seja um erro, o terminal apresentará mensagem de desconformidade, conforme a Figura 10 demonstra.

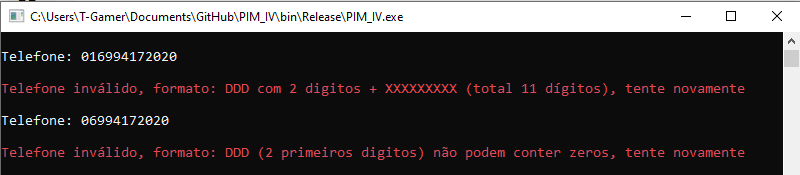
Figura - Desconformidade na inserção do nome do paciente



Fonte: Elaborado pelos autores (2022)

Após o preenchimento do nome, o campo telefone será apresentado, e deve ser satisfeito através da inserção de um telefone celular válido com tamanho de 11 caracteres e composto por números. Conforme Figura 11, os possíveis erros que o usuário pode enfrentar são:

Figura - Possíveis erros de validação do campo telefone



Fonte: Elaborado pelos autores (2022)

O endereço é composto pelos campos rua, número, bairro, cidade, estado e CEP. Cada um deles possuem validações, abaixo tem-se a explicação de cada uma delas:

* Rua:
  + Rua não informada, tente novamente: quando não é inserido nenhuma informação.
* Número:
  + Número inválido, somente números positivos até 999999, tente novamente: o valor imputado não é um número, ou não é positivo ou ainda ultrapassa a quantidade máxima de dígitos esperados.
* Bairro:
  + Bairro não informado, tente novamente: quando usuário tenta prosseguir sem inserir nenhum valor.
* Cidade:
  + Cidade não informada, tente novamente: nenhuma informação foi imputada.
* Estado:
  + Estado inválido, formato com 2 dígitos dentre os estados brasileiros, tente novamente: o dado inserido não possui o tamanho correto, ou não está na lista de estados brasileiros.
* CEP:
  + CEP inválido, apenas números, tente novamente: o CEP inserido não possui a quantidade de caracteres corretas ou foi preenchida por números.

A seguir, tem-se o campo data de nascimento que precisa ser preenchido corretamente, caso contrário o usuário poderá sofrer com os seguintes apontamentos:

* Data inválida, a data informada não pode ser superior a data atual, tente novamente;
* O ano informado também não pode ser 150 anos antes do ano atual, tente novamente;
* Mês inválido, tente novamente;
* Dia informado não pode ser superior ao número de dias que o mês pode conter, tente novamente;
* Data inválida, formato: ddmmaaaa, apenas números positivos, tente novamente;

Com a data de nascimento, o programa verifica se o paciente possui mais de 65 anos e, por consequência, pertence ao grupo de risco.

O próximo campo a ser preenchido se refere ao e-mail do paciente. Assim como Holovach (2020) explica, existem diversas regras para que um e-mail seja adequado, como não ter caracteres especiais ocupando a primeira posição ou parte que antecede o símbolo arroba (‘@’), ou só ser possível inserir letras e caracteres especiais na parte anterior à arroba (‘@’) respeitando o que foi dito anteriormente, ou só possuir letras do alfabeto após o ponto (‘.’) que indica o domínio do e-mail. A seguir estão os possíveis erros que o usuário pode enfrentar caso o dado preenchido não seja válido:

* E-mail inválido antes do @. É permitido texto começado ou terminado com letras/números;
* Entre elas pode conter caracteres especiais como: { #, !, %, ', &, +, \*, -, /, =, ?, ^, `, ., {, |, }, ~, $ };
* E-mail inválido, ou ‘@’ não encontrado ou não possui mais caracteres disponíveis após o ‘@’. Tente novamente;
* E-mail inválido no texto após o '@'. É permitido apenas letras. Tente novamente;
* E-mail inválido, ponto '.' não encontrado no texto após o @, tente novamente;
* E-mail inválido, não encontrado texto após o '.'. Tente novamente;
* E-mail inválido no texto após o ponto '.'. É permitido apenas letras, opcionalmente pode conter um segundo ponto '.'. Tente novamente;
* E-mail inválido no texto após o segundo ponto '.'. É permitido apenas letras. Tente novamente;
* E-mail inválido, formato: texto@texto.texto(.textoAlternativo), tente novamente.

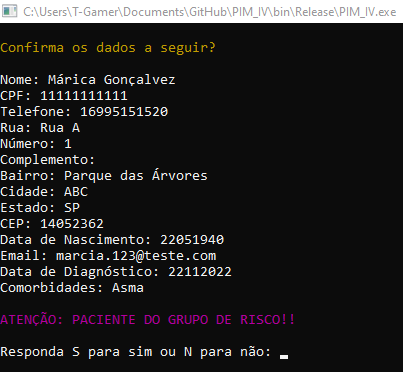
A seguir, o campo data de diagnóstico deve ser preenchido. A informação inserida deve ser uma data com ano, mês e dia válidos, não superior a data atual e necessita ser maior que a data de nascimento do paciente. Os apontamentos que o programa informa ao usuário caso o valor preenchido não de acordo com preceitos levantados anteriormente são:

* Data inválida, a data informada não pode ser superior a data atual, tente novamente;
* O ano informado também não pode ser 150 anos antes do ano atual, tente novamente;
* Mês inválido, tente novamente;
* Dia informado não pode ser superior ao número de dias que o mês pode conter, tente novamente;
* A data de diagnóstico não pode ser antes da data de nascimento, tente novamente;
* Data inválida, formato: ddmmaaaa, apenas números positivos, tente novamente.

O último campo que o usuário deve inserir é de comorbidades. Seu objetivo é registrar disfunções que podem agravar o quadro de um paciente que possui Covid19.

Conforme Figura 12, após o preenchimento de todos os campos, as informações inseridas serão mostradas ao usuário para que sejam confirmadas. Caso o usuário deseje corrigir algum dado deve ser digitar ‘N’ para não realizar a confirmação, caso contrário, se a intenção é aprovar o registro, ‘S’ deve ser inserido.

Figura - Confirmação dos dados inseridos



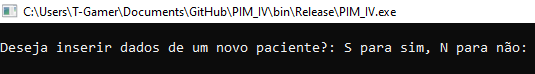
Fonte: Elaborado pelos autores (2022)

Assim que as informações são confirmadas, uma mensagem em azul informará ao usuário que o arquivo de texto com o registro do paciente está sendo salvo. Se for concluído a mensagem “Salvo arquivo do paciente com sucesso!”, mas se o sistema falhar, o *software* apresentará uma mensagem de erro. Caso o problema aconteça, o usuário deve contatar o administrador.

Ademais, na hipótese de o paciente pertencer ao grupo de risco, além de armazenar no computador um arquivo com todas as informações, o programa também salva um outro registro contendo somente o CEP e a idade do paciente para que depois possa ser encaminhado para a Secretaria de Saúde da cidade. Se o programa gravar com sucesso o novo arquivo texto, a mensagem “Salvo arquivo do paciente do grupo de risco com sucesso!” aparecerá, caso contrário o apontamento “Erro ao salvar arquivo do paciente do grupo de risco, contate o administrador” surgirá na tela do usuário. Caso o problema aconteça, o usuário deve contatar o administrador.

Assim como a Figura 13 demonstra, após salvar as informações, a aplicação pergunta ao usuário se ele deseja registrar um novo paciente.

Figura - Pergunta de reinicio de ciclo



Fonte: Elaborado pelos autores (2022)

Caso seja inserido a letra ‘S’ para sim, o programa volta ao início do formulário, se o valor inserido for ‘N’, o sistema finaliza o *login* do usuário com a mensagem “Deslogando...”.

## Resgatando os Arquivos Salvos

Conforme dito anteriormente, todas as informações do paciente, sem ter em conta se o enfermo pertence ao grupo de risco ou não, são salvas em um arquivo texto que fica disponível dentro da pasta “arquivos/saída/pacientes”. Os arquivos são nomeados automaticamente com o CPF da pessoa.

Para recuperar os arquivos do grupo de risco, o usuário deve adentrar na pasta “arquivos/saída/grupo\_de\_risco”. Assim como os registros completos, esses arquivos textos são intitulados conforme o CPF informado durante o preenchimento do formulário.

# CONSIDERAÇÕES FINAIS

Nesse projeto foi proposto um software de uso na área da saúde com o objetivo de gerenciar a entrada de pacientes com covid-19 desenvolvido com a utilização da linguagem C.

Com o escopo previamente definido, foi feita a escolha do modelo do método que seria utilizado, chegou-se à conclusão de que a melhor escolha seria um sistema híbrido levando em consideração os requisitos impostos para o desenvolvimento, de forma que o possibilitou fazer seguindo o modelo cascata, mas de forma incremental, evoluindo modulo a modulo a fim de atender os requisitos que foram levantados.

O desenvolvimento foi realizado através de ferramentas como o Code::blocks que contribuiu para a elaboração de todo o código fonte do programa em linguagem C. Outra ferramenta essencial utilizada foi o Git, armazenando cada etapa do trabalho em *commit*.

O levantamento dos requisitos funcionais e não funcionais possibilitou determinar os métodos utilizados para elaboração de programa de registros, com acesso por *login* com senha, e formulários para cadastro de dados específicos dos pacientes, com suas respectivas validações em cada campo.

Embora seja um software simples, o desenvolvimento dele será muito útil, tendo em vista que com ele, a unidade médica poderá efetuar a triagem dos pacientes com maior efetividade. Classificando-os de acordo com aspectos como gênero, idade ou comorbidades. Também pode ser usado os dados de endereço para fazer um possível mapeamento estatístico da propagação da enfermidade.

#### REFERÊNCIAS

CODE::BLOCKS: the free C/C++ and Fortran IDE. **Code::Block**, c2020. Disponível em: <https://www.codeblocks.org/>. Acesso em: 10 nov. 2022.

COSTA, Carlos Jorge Almeida. Aspectos de Programação Modular em C. 2017. 62 f. Dissertação (Relatório científico para instituição nacional) - Instituto Politécnico de Viseu, Viseu, 2017. Disponível em <https://repositorio.ipv.pt/bitstream/10400.19/6340/1/b\_LicaoAspetosProgramacaoModularC.pdf>. Acesso 24 nov 2022.

HOLOVACH, Helen. **Como é um formato válido de endereço de e-mail**. Site Snovio. Disponível em: <https://snov.io/knowledgebase/br/como-e-um-formato-valido-de-endereco-de-E-mail/>. Acesso em: 26 out 2022.

JORY, Natan. **Métodos Híbridos para Gerenciamento de Projetos de Desenvolvimento de Software**. São José: Instituto Federal De Santa Catarina, 2018. Disponível em: <https://wiki.sj.ifsc.edu.br/wiki/images/9/9a/TCC290\_Natan\_Martins\_Jory.pdf>. Acesso em: 15 nov. 2022.

MANZANO, José Augusto N. G. **Estudo dirigido de linguagem C**. 17. ed. São Paulo: Editora Érica, 2013.

PAULA FILHO, Wilson de Pádua. **Engenharia de Software**. 04. Ed. Rio de Janeiro: Grupo GEN, 2019. Disponível em: <https://integrada.minhabiblioteca.com.br/#/books/9788521636724/>. Acesso em: 19 nov 2022.

PRESSMAN, Roger; MAXIM, Bruce. **Engenharia de Software**. 08. Ed. Porto Alegre: Grupo A, 2016. Disponível em: <https://integrada.minhabiblioteca.com.br/#/books/9788580555349/>. Acesso em: 19 nov 2022.

RAMOS, Guilherme. O que é o GitHub? Veja para que serve a 'rede social de programadores'. **Techtudo**. Disponível em: http: <https://www.techtudo.com.br/listas/2021/05/o-que-e-o-github-veja-para-que-serve-a-rede-social-de-programadores.ghtml>. Acesso em: 21 nov 2022.

SOMMERVILLE, Ian. **Engenharia de Software**, 9ª Edição. Pearson. São Paulo, Brasil, 2011. Disponível em: <https://www.facom.ufu.br/~william/Disciplinas%202018-2/BSI-GSI030-EngenhariaSoftware/Livro/engenhariaSoftwareSommerville.pdf>. Acessado em : 19 nov 2022.