**2. Linguagem C e Metodologia**

2.1 Utilizando a linguagem C

A construção dessa aplicação, será feita toda utilizando a linguagem C que foi uma linguagem criada em 1972 por Dennis Ritchie e Ken Thompson no laboratório Bell com um foco em desenvolvimento de sistemas operacionais como por exemplo o sistema UNIX, e possui a vantagem imensa para produzir sistemas operacionais, gerenciadores de bancos de dados, planilhas eletrônicas e é uma linguagem de fácil legibilidade e documentada conforme (1). Assim como boa parte das linguagens de programação como o C, um compilador é utilizado para traduzir a linguagem de programação que está mais próximo do desenvolvedor para a linguagem de máquina, que é como o computador entende os comados. O compilador lê as instruções que foram codificadas no programa, analisa se existem erros na sintaxe das instruções e converte essas instruções em linguagem de máquina e depois segue para a próxima instrução até que as instruções acabem e em seguida o compilador transforma o código fonte em um arquivo com extensão .exe, ou seja um executável que roda no sistema operacional conforme mostrado por (2). Para facilitar o desenvolvimento do software, existem editores de códigos que facilitam o processo de criação, codificação, compilação, manutenção e uso de outras ferramentas. São conhecidos como IDE (Integrated Development Environment) ou Ambientes Integrado de Desenvolvimento. Para a construção desse software, será utilizado a IDE Code Blocks, visto que esse editor já possui um compilador e agilizará o processo de desenvolvimento da aplicação. Essa IDE contém atalhos para indentação que ajuda na visualização e organização do código e um console que exibe mensagens de erro caso seja cometido algum erro na codificação. Voltando a linguagem C, ela é conhecida por ser uma linguagem procedural, ou seja, a lógica costuma estar separada como atributos e comportamentos. Sabendo disso, antes de desenvolver o algoritmo que irá processar a regra de negócio que foi proposta a essa aplicação, primeiramente será definido uma metodologia para o processo de construção da aplicação.

2.2 Metodologia aplicada \*Jéssica

A metodologia de software utiliza um conjunto coerente e coordenado de abordagens que podem ser usadas para criar um sistema de processamento de dados. Seu objetivo é definir o papel de cada envolvido, diretamente ou indiretamente, no desenvolvimento do software, fornecendo um processo dinâmico e interativo, visando à qualidade e produtividade dos projetos. Para a garantia de sucesso e mais eficiência no desenvolvimento do projeto, o responsável pela equipe deve conhecer as diferentes metodologias existentes e optar por aquela que melhor se adequa para o seu caso. A metodologia possui suas vantagens e desvantagens, cada qual se encaixando em situações distintas, por isso há a necessidade de conhecer os objetivos, prazos, metas e orçamentos do projeto, para a determinação da melhor abordagem a ser utilizada, com base na natureza do projeto e do produto a ser desenvolvido. Para este projeto, será adotado a metodologia em cascata. Criado em 1970 por Winston W. Royce, o modelo em cascata é o mais antigo dos processos, possui esse nome por ser um modelo sequencial que visa um fluir constante do processo de desenvolvimento de software, para que possam ser realizadas essas etapas sequenciais, o método prevê que o projeto só avance a partir do momento que uma etapa é finalizada. Ao total são cinco etapas, sendo elas, o levantamento de requisitos, o projeto, a codificação, os testes e a implementação e manutenção.

Diagrama

Descrição gerada automaticamente

[**https://pt.wikipedia.org/wiki/Modelo\_em\_cascata**](https://pt.wikipedia.org/wiki/Modelo_em_cascata)

Requisitos: nesta fase é feito um levantamento das necessidades, objetivos e expectativas dos usuários, com base nestes dados, são definidas as funcionalidades, elementos de hardware e software necessários, requisitos de negócio e o que deverá ser feito no desenvolvimento do software.

Projeto(design): é realizado um planejamento da aplicação que será construída, envolvendo a criação de um cronograma, com estimativas do término de cada etapa, definição da arquitetura do software, do banco de dados e das características visuais, com base nos requisitos, montagem do time desenvolvedor etc.

Codificação (implementação): nesta etapa, o software é codificado, através de linguagens de programação, de acordo com as especificações e requisitos do projeto.

Testes: fase voltada para testes e verificação da aplicação, para a garantia de que cumpre com as especiações e seu objetivo. É verificado também se há algum erro, para que seja corrigido antes de prosseguir para a próxima etapa.

Implementação e Manutenção: o sistema é implantado para que o cliente veja o resultado, caso seja necessária alguma alteração no sistema relacionada a erros, identificação de necessidades não vistas anteriormente, evolução do negócio, ou evolução tecnológica, o software deverá passar por uma manutenção, que pode ser feita com o modelo em cascata. Sendo assim, essa fase é responsável pela correção de erros ou melhorias do sistema.

É sugerido seu uso em projetos com requisitos bem definidos e de pequena duração, e ao utilizar este modelo, podemos encontrar algumas dificuldades, como em projetos reais que nem sempre são sequenciais, sendo necessária mudanças que podem ocasionar problemas à medida que o projeto prossegue. Também é necessário que o cliente estabeleça todas as suas necessidades logo no início do projeto, visto que, o produto somente é visível no final de todo o ciclo. Por outro lado, o método pode possuir um acordo de risco logo no início do processo, e por ter todos os detalhes, é mais fácil de gerenciar e planejar.

**3. Levantamento de Requisitos**

Sendo uma das partes mais importantes, o levantamento de requisitos será a etapa que irá ditar como a construção da aplicação será feita. Nessa etapa, conseguimos extrair do cliente as suas necessidades e expectativas em relação ao produto e quais funcionalidades que ele deseja que o sistema tenha. Basicamente a demanda do cliente exige um software capaz de cadastrar e enviar para a secretaria da saúde pacientes contaminados com o covid-19. A aplicação deve conter inicialmente uma tela de login, no qual o usuário entra com suas credenciais de login e senha, e seja redirecionado para o menu principal da aplicação, onde esse menu deve trazer a opção de cadastrar paciente juntamente com os seus dados, e enviar para a secretaria da saúde, pacientes já cadastrados que tenham mais de sessenta e cinco anos e que tenha algum agravante em sua saúde, como diabete, hipertensão ,etc. Também é importante que exista uma opção de o usuário consultar quais pacientes já foram cadastrados anteriormente. De forma resumida, a aplicação terá uma tela de login, um menu para a seleção das funcionalidades, a funcionalidade de cadastro de pacientes, consulta de pacientes e envio de pacientes para a secretaria da saúde. Vale ressaltar que ao enviar pacientes que atendam os requisitos citados anteriormente, deve-se registrar em um arquivo, o CEP e a idade do paciente ao enviar para secretaria da saúde. O cadastro dos pacientes ficará em outro arquivo, para fins de consulta e envio para a secretaria. O programa só poderá enviar pacientes que atendam os requisitos e que já estejam previamente cadastrados no sistema.

**4. Projetando**

Seguindo o a metodologia cascata, após a etapa de levantamento de requisitos, já se possui basicamente todas as informações necessárias para a construção da aplicação. Nessa etapa será definido etapas como o cronograma, tarefas, estimativa de prazos, desenvolvedores, e arquitetura da aplicação. Começando com o cronograma, a aplicação a ser desenvolvida é de pequeno porte, ou seja, pode ser entregue em um mês. Na primeira semana foi analisado os requisitos necessários para a construção e foram criadas as tarefas e distribuídas entre os desenvolvedores responsáveis. E segunda e terceira semana foram dedicadas a construção da aplicação, aonde a segunda semana foi dado a ênfase a modelagem da dados, a arquitetura do software, a IDE que será utilizada, levantamento de ferramentas necessárias para complementar o desenvolvimento, e a codificação em si, e a terceira semana foi focada em testes da aplicação por parte de desenvolvedores. Na última semana, a aplicação é entregue ao cliente para que ele possa testar todas as funcionalidades e ver se os requisitos desejados são atendidos, e caso surja algum mal funcionamento da aplicação, o projeto volta para a fase de desenvolvimento e sejam feitas as devidas correções e testes e entregando para o cliente validar. Caso o cliente valide, o projeto é finalizado.

**5. A Implementação**

5.1 Bibliotecas

As bibliotecas são arquivos que contém código pronto feito por outros desenvolvedores e podem ser utilizadas em nosso projeto conforme explica (3). Normalmente para se importar uma biblioteca padrão fornecida pelos compiladores C, são feitas no início do arquivo utilizando o caractere cerquilha (#) seguida da palavra reservada “include”, espaço e entre os sinais de menor e maior, declara-se o nome da biblioteca desejada. Esse comando colocado no cabeçalho do arquivo, indica para o compilador que essa biblioteca deve ser incluída para que o programa funcione.

Texto

Descrição gerada automaticamente

As bibliotecas contêm muitos códigos prontos, isso proporciona uma agilidade maior no desenvolvimento da aplicação e está menos propenso a erros visto que devemos focar em como nossa codificação está sendo desenvolvida, e não no funcionamento dos métodos e atributos de uma biblioteca específica, visto que essa responsabilidade fica a cargo da biblioteca processar os dados e não do desenvolvedor que a utiliza. Algumas bibliotecas conhecidas serão utilizadas em nosso projeto como a biblioteca stdio.h (Standart Input/Output) que contém métodos prontos que fazem a manipulação de entrada e saída de dados da aplicação e realizando escrita e leitura de arquivos. Também será utilizado a biblioteca stdbool.h (Standart Boolean) que permitira o uso de variáveis do tipo bool que são valores lógicos (1 ou 0, True ou False).

--ADICIONAR PESQUISA QUE FALTA

5.2 Estruturas

No desenvolvimento do projeto, será necessário lidar com a transação de dados que serão armazenados na memória do computador. (4) explica que existem alguns tipos primitivos que a linguagem C fornece para armazenar determinados tipos de dados. Temos as variáveis do tipo int que armazena números inteiros; o tipo float que armazena números de ponto flutuante que tem casas decimais; o tipo double que também armazena números de ponto flutuante, porém com mais precisão que o tipo float mas ocupa mais espaço de memória; e o tipo char que armazena um caractere alfanumérico. Porém, para armazenar informações que pertencem a um mesmo contexto como por exemplo, os dados de um paciente, trabalhar com variáveis “soltas” no código, além de tornar o código pouco legível, fica de difícil manutenção e não é uma boa prática de programação. É mais viável centralizar essas variáveis em estruturas ou structs. As structs que é uma abreviação para structure ou estrutura em português, é uma estrutura de dados que permite em seu corpo declarações de variáveis de diferentes tipos que podem ser acessadas pelo mesmo ponteiro. Nela podemos armazenar diversos tipos de dados como dados de cadastro como explica (5). Será necessário modelar algumas structs para a manipulação dos dados na aplicação. Com essas structs bem definidas, ficará melhor para futuras manutenções já que se houver a necessidade de mudar o tipo de uma variável, adicionar ou remover, fica tudo centralizado em uma struct. Alguns dados são cruciais para que possamos identificar o paciente no momento de fazer o cadastro, como: Nome, CPF, CEP, Aniversário, Idade, Data do diagnóstico, se tem diabete, se é obeso, se é hipertenso, se já teve tuberculose. As datas serão no padrão dia/mês/ano, ou seja, para a data de aniversário, e para a data do diagnóstico, serão utilizadas structs que armazenarão esses dados. Para isso, foi criada uma struct chamada Date e nela contém três variáveis do tipo inteiro que são: day, month e year que irão armazenar as informações de dia, mês e ano respectivamente.

Texto

Descrição gerada automaticamente

Em seguida foi criada uma struct chamada Patient que irá centralizar as variáveis que conterá os dados de cadastro do paciente. Visto que um nome contém vários caracteres, uma variável do tipo char, não atende a esse contexto visto que esse tipo só armazena na memória do computador, um caractere alfanumérico e precisamos armazenar vários caracteres. Nesse caso será utilizado um vetor ou um array, que é uma estrutura de dados que guarda e organiza elementos em espaço fixo de memória e esses elementos podem ser acessados por meio de um índice, que indica qual posição se encontra o elemento dentro da coleção conforme mostra (6). Dessa forma, foi criado um vetor de char reservando cem posições para armazenar o nome do paciente na variável Name, para armazenar CEP e CPF foi utilizado variáveis do tipo inteiro. Para armazenar aniversário (\_BirthDate) e data de diagnóstico (\_diagnosisDate), foi utilizado a struct date. Para armazenar informações com respeito as comorbidades dos pacientes, foram utilizados variáveis do tipo lógico ou booleano que podem armazenar somente verdadeiro ou falso (0 ou 1) e por isso foi colocado no cabeçalho do arquivo #include <stdbool> para que esses tipos lógicos fossem utilizados na codificação. A estrutura de paciente foi estruturada conforme a imagem a seguir:

Texto

Descrição gerada automaticamente

5.3 Métodos

As funções ou métodos, é um conjunto de código ou instruções que tem o propósito de entregar uma tarefa em particular e é agrupada numa unidade e pode ser invocada pelo nome designado a função como mostra (7). Funções podem ou não receber parâmetros que são passados entre parênteses após o nome da função e esses dados são processados nas instruções da função. As funções podem processar dados e retornar ou não dados para a pilha na qual foi chamada. Assim sendo, a lógica da aplicação foi segmentada em sete métodos: o método main, que é o primeiro método a ser chamado quando a aplicação é iniciada, todo o código dentro desse método é executado e depois finaliza a aplicação.; Login() é o método que será responsável para verificar se as credencias do cliente são válidas ao tentar acessar o sistema; SystemOptions() é responsável em exibir o menu principal da aplicação para o cliente, no qual é possível selecionar quais funcionalidades do software deseja-se usar; BuildPatient() é o método responsável por montar a struct com os dados capturados do teclado do cliente e armazenar nas devidas variáveis dentro da struct Patient; AddPatient() é responsável por escrever em um arquivo .txt os dados digitados pelo cliente; ValidateNumber() é responsável por validar se os dados digitados pelo cliente são números; ValidateString() é responsável por validar se os dados digitados pelo cliente são letras; GetPatient() é o método que lê o arquivo .txt que se encontra na raiz do projeto, arquivo esse que armazena uma lista de pacientes cadastrados e exibe na saída do console para o cliente visualizar todos os pacientes cadastrados; SendPatient() é responsável em ler o arquivo que contém os pacientes, analisar se algum paciente tem mais de sessenta e cinco anos e se tem alguma comorbidade . A seguir será detalhado a lógica de cada método, em como foi construído o algoritmo e técnicas de validações para tratamento de dados e retorno de dados desses métodos.

5.3.1 Main()

Método que retorna um inteiro. Esse método é o primeiro a ser chamado quando a aplicação é iniciada e todo o código que estiver dentro dele é executado e quando não houver mais instruções, a aplicação é encerrada. Essa função irá fazer a chamada de duas funções que é a função de fazer login no sistema e outra função para acessar o menu principal da aplicação conforme a figura a seguir:

Texto

Descrição gerada automaticamente

5.3.2 Login()

O método Login() é invocado assim que a aplicação é iniciada, esse método tem por responsabilidade solicitar o usuário e senha do cliente, captar essa entrada de dados e verificar se são credenciais válidas e redirecionar para o menu raiz onde contém as principais funcionalidades do software. Foi declarado um vetor de char chamado userScan que irá armazenar o usuário e outro vetor de char chamado pwScan para armazenar a senha digitada. Também foi declarado uma variável de controle chamada login\_efetuado do tipo inteiro que será responsável para fazer a autenticação do usuário no sistema. Para manter o usuário “preso” na tela de login do sistema até que suas credenciais sejam aceitas e ele seja redirecionado para a tela principal do sistema, é necessário o uso de um laço de repetição ou loop.” Laços são comandos usados sempre que uma ou mais instruções tiverem de ser repetidas enquanto uma certa condição estiver sendo satisfeita” (7). Dentre os laços existentes, o que melhor atende a essa lógica é o while onde, enquanto uma certa condição não for satisfeita, o loop será feito.

Um loop while é declarado com a condição de que enquanto o valor da variável login\_efetuado for igual a zero ou false, o bloco repetirá a lógica a seguir. Para fazer exibição das mensagens para o usuário, foi utilizado uma função pertencente a biblioteca stdio.h que é a função printf(), que exibe na tela do usuário alguma informação e recebe como primeiro parâmetro uma cadeia de caracteres ou também conhecido como string que será escrito em aspas duplas, passando o especificador de formato que começa com o caractere % seguido do formato do tipo dado que será exibido. Como será exibido uma mensagem para o usuário, será utilizado o especificador de acesso “%s”, que formata o dado que será passado nos parâmetros seguintes para um formato de string. Para a entrada de dados ou ler o que o usuário digitou, foi utilizado outra função da biblioteca stdio.h que é a função scanf(), que consegue fazer a leitura da entrada de dados do usuário através do teclado. Essa função recebe dois parâmetros onde o primeiro é uma string como formato do dado capturado e o segundo parâmetro, é a variável onde esse dado deve ser armazenado. Após a entrada de dados ter sido feita pelo usuário, será utilizado uma condicional IF que analisa uma expressão booleana que se retornar verdadeiro o código dentro do bloco IF é executado e se retornar falso, o bloco ELSE será executado. As credenciais aceitas para o usuário ser autenticado no sistema, foram definidas como variáveis globais após as diretivas #include, no qual essas variáveis podem ser acessadas em qualquer escopo da aplicação. Para isso foi declarado um vetor de char chamado \_userName inicializado com o valor “admin” e outro vetor de char chamado \_passWord inicializado com o valor de “1234567”. Para realizar a comparação dos dados que usuário digitou com as credenciais previamente estabelecidas no escopo do arquivo, foi utilizado uma função da biblioteca string.h chamada strcmp() (string compare) que recebe por parâmetro dois vetores de char e retorna um valor inteiro onde, zero significa que os dois vetores são iguais, e -1 ou 1 que os vetores não são iguais conforme (7). Dessa forma dentro da expressão IF, é feito a comparação dos dados de usuário e usando o operador lógico && que significa “E” e comparando as senhas. Como está sendo usado o operador lógico “E” ou &&, ambos as expressões precisam ser verdadeiras, ou seja, ambas as expressões precisam retornar o valor zero para o bloco IF ser executado. Caso o bloco IF seja executado a função system passando como parâmetro o argumento “cls” para limpar as mensagens do console e a variável login\_efetuado é setada com o valor 1 e o loop é encerrado. Caso as comparações na condicional IF retornem falso, o bloco ELSE é executado e nesse bloco é utilizado a função printf para exibir ao usuário que suas credenciais não são válidas e o loop recomeça solicitando usuário e senha. Uma breve explicação sobre a função system que executa um comando interno de cadeia de caracteres no sistema operacional. Ela localiza o interpretador de comandos que costuma ser o CMD.EXE e executa o comando passado como argumento que é uma string segundo (8). O comando “CLS” é utilizado para apagar ou limpar as informações mostradas na saída do console. A imagem a seguir demonstra como foi codificado o método Login:

Interface gráfica do usuário, Texto, Email

Descrição gerada automaticamente

5.3.3 SystemOptions

Método sem retorno (void) responsável em exibir as principais funcionalidades do software. Aqui o cliente pode selecionar através de números qual funcionalidade o cliente deseja usar. A opção 1 permite cadastro de pacientes; A opção 2 permite visualizar os pacientes cadastrados; A opção 3 permite enviar os pacientes previamente cadastrados para a secretaria da saúde; A opção 4 permite a saída do menu, finalizando o programa. Foi utilizado a função printf para exibir a mensagem das opções disponíveis e o scanf para captar qual opção o cliente digitou e armazena na variável conditional do tipo inteiro. Em seguida essa variável é analisada numa estrutura condicional chamada switch para determinar qual fluxo a aplicação seguirá onde: no caso de ter digitado o valor 1 o método BuldPatient é chamado; No caso 2 o método GetPatients é chamado; no caso 3 o método SendPatients é chamado e no caso 4 nenhuma lógica é executada. Toda essa lógica se encontra dentro de um do/while. Caso o valor que cliente digitou seja maior que 0 e menos que 4, o loop é encerrado e aplicação é encerada. Dentro de cada case dentro do switch, contém a instrução break que encerra “sai” do bloco switch após a chamada da função, indicando que o switch foi encerrado. A imagem a seguir mostra a estrutura dessa função.

Interface gráfica do usuário, Texto, Aplicativo

Descrição gerada automaticamente

5.3.4 ValidationNumber()

Esta função captura o caractere digitado pelo usuário, se o caractere for um número, a função exibe o número digitado na tela, armazena no vetor de caracteres e incrementa o índice do vetor, caso o caractere não seja um número, a função não exibe o caractere na tela. A função também permite o uso da tecla de BACKSPACE decrementando o índice do vetor de caracteres e apaga da tela o caractere que havia sido digitado pelo usuário. A função roda em loop até que a tecla ENTER do teclado seja pressionada e retorna os números digitados pelo usuário.

5.3.5 ValidationString()

Esta função captura o caractere digitado pelo usuário, se o caractere for uma letra, a função exibe a letra digitada na tela, armazena no vetor de caracteres e incrementa o índice do vetor, caso o caractere não seja uma letra, a função não exibe o caractere na tela. A função também permite o uso da tecla de BACKSPACE decrementando o índice do vetor de caracteres e apaga da tela o caractere que havia sido digitado pelo usuário. A função roda em loop até que a tecla ENTER do teclado seja pressionada e retorna as letras digitadas pelo usuário.

5.3.6 BuildPatient

Função sem retorno responsável por captar a entrada de dados por parte do usuário e armazenar os valores nas variáveis dentro da struct do tipo Patient. É criado quatro variáveis, uma do tipo Patient chamada \_patient que é uma struct para armazenar as informações do paciente, duas do tipo Date chamadas \_birthDate e \_diagnosisDate que também é uma struct para armazenar os dados relativos a datas e uma do tipo int chamada conditional inicializada com o valor 1 que servirá de controle para o loop a seguir. Após a declarações das variáveis que irão armazenar os dados do paciente, A lógica seguinte foi colocada dentre de um loop do/while porque caso o usuário deseja refazer o cadastro, ele terá a opção de refazer o cadastro e a variável conditional servirá de controle para determinar se o loop será encerrado ou não. Foi utilizado a função printf para exibir mensagens ao cliente solicitando os dados para o cadastro do cliente como nome, CPF, idade, data de aniversário etc. A função ValidateString e ValidateNumber que são funções especializadas em tratamento de dados como veremos adiante, são utilizadas para receber essas entradas de dados feito pelo cliente e ir armazenando nas variáveis respectivas da struct \_patient. Para armazenar esses dados do usuário nas respectivas variáveis dentro da struct \_patient, deve- se chamar o nome da struct que é a \_patient seguido de ponto (.) e o nome da variável no qual deseja armazenar ao dado conforme mostra a figura a seguir:

Texto

Descrição gerada automaticamente

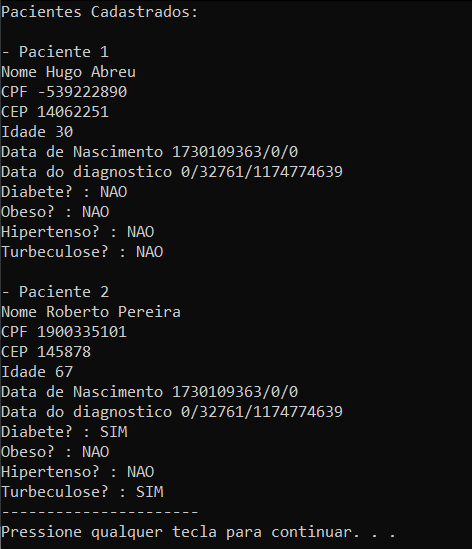
Após as solicitações dos dados do paciente para o cadastro, é exibido todas as informações que foram cadastradas utilizando a função printf e perguntando ao usuário se deseja concluir o cadastro digitando 0 ou refazendo o cadastro digitando 1. Com a função scanf é feita a captura da opção do usuário e armazenada na variável conditional. Em seguida é feito um teste nesta variável para ler qual opção o usuário escolheu através de uma expressão condicional IF. Caso tenha escolhido 0, a função CreatePatientFile é chamada e recebe dois parâmetros onde o primeiro recebe um argumento do tipo Patient e o segundo é uma string onde passa-se o caminho do arquivo que veremos adiante na explicação da função CreatePatientFile. Essa função retorna um valor booleano onde 1 indica que o processamento dentro dela deu certo ou false caso surja alguma falha de processamento. Se retornar 1, uma mensagem indicando que o paciente foi cadastrado é exibida usando printf, o loop é encerrado e o usuário é retornado para a tela principal, caso falhe outra mensagem é exibida indicando a falha. A função system é chamada passando como argumento a string “PAUSE”, que indica ao usuário que ele precisa digitar alguma tecla para continuar Agora se o usuário digitou 1, o loop recomeça e os dados de cadastros serão novamente solicitados.

5.3.7 CreatePatientFile

Método que retorna um booleano que recebe uma variável do tipo Patient por parâmetro, uma string para indicar o caminho que o arquivo será salvo, e um inteiro para indicar como será o formato que será aplicado ao salvar o arquivo. Os dados serão em um arquivo do tipo texto e para cada informação do paciente, como nome, idade etc., serão separados por vírgula e ao final serão utilizados ponto e vírgula. Cada linha nesse arquivo representa um cliente. Para escrever esses dados no arquivo, deve-se criar um ponteiro para o local do arquivo. Para isso é necessário utilizar um método da biblioteca stdio.h chamada fopen (file open) que recebe no primeiro parâmetro uma string indicando o caminho do arquivo e o segundo parâmetro que é um char para indicar o modo de acesso ao arquivo que neste caso é adicionar dados ao arquivo, então é passado o char “a” que é equivalente ao “add” e retorna um ponteiro ou endereço de memória do arquivo. Como o método CreatePatientFile recebe por parâmetro um vetor de char chamado folder que é o endereço do arquivo, esse parâmetro é passado para a função fopen diretamente. A função fopen() de forma resumida, executa duas tarefas principais onde a primeira ela cria uma estrutura FILE com todas as informações que são necessárias para a aplicação e para o sistema operacional para comunicação e em seguida retorna um ponteiro FILE apontando para o local do arquivo conforme (9). Após obter o ponteiro do arquivo, é feito uma condicional em cima do parâmetro chamado format do tipo inteiro que determinará se os dados do paciente serão no todos salvos que será no arquivo Pacientes.txt ou apenas o nome e o CEP no arquivo PaceintesEnviados.txt, que conterá todos os pacientes com mais de sessenta e cinco anos e tem algum agravante na saúde. Para a escrita dos dados do paciente, será utilizado o método fprintf() da biblioteca stdio.h que grava as informações de modo formatado bem semelhante a função printf() mas o ponteiro do arquivo é passado como primeiro parâmetro, o parâmetro seguinte é passado a string de formatação e nos parâmetros seguintes são passados os respectivos dados como mostra (10) . Após a escrita dos dados no arquivo, é necessário fechar o arquivo utilizando a função fclose(). (11) explica que os caracteres são armazenados primeiramente num espaço de memória chamado buffer e quando esse buffer lota, ele despejado no disco físico do computador. A função fclose() força que o buffer seja despejado no disco físico e libera as comunicações usadas para que o arquivo fico disponível. Após o fechamento do arquivo é retornado o valor true indicando que os dados foram salvos no arquivo

5.3.8 GetPatient

Método sem retorno responsável em ler o arquivo Pacientes.txt e exibir na saída o nome de todos os pacientes cadastrados. O método fopen é acionado recebendo no primeiro parâmetro o diretório do arquivo Pacientes.txt e o segundo parâmetro é um char “r” (read) que significa que será feita uma leitura do arquivo. O retorno desse método é um ponteiro do arquivo que é armazenado na variável file do tipo FILE. Logo a seguir é feito um teste através de uma condicional IF para testar se o ponteiro é nulo. Caso o teste seja verdadeiro, indicando que o ponteiro está nulo, uma mensagem é exibida para o cliente informando para o usuário que o arquivo não foi encontrado. Caso existam pacientes previamente cadastrados, é necessário utilizar a função fgets (File Get String) que recebe um vetor de char onde será salvo a string lida, um inteiro indicando a quantidade máximas de caracteres que serão lidos e de onde será lido o arquivo conforme explicado por (11). Então foi criado um vetor de char com dez mil posições, uma variável do tipo inteiro chamado count que servirá de contador para os pacientes. O vetor de char é passado como parâmetro par ao método fgets(), é passado valor de dez mil para leitura dos caracteres, e a varíavel que contém o ponteiro do arquivo. Esse método foi colocado num loop While para repetir esse loop sempre que o retorno de fgets seja diferente de nulo. Dentro do loop foi criado uma variável do tipo Patient que é mesma struct utilizada pra armazenar os dados do paciente. Ao ler o arquivo é necessário segmentar a string do arquivo lido pelas vírgulas, visto que é o formato que os pacientes foram salvos, todas as informações separadas por vírgula. Para isso foi utilizado a função strtok (string token) que “quebra” a string em tokens ou caracteres em específicos conforme (12). Essa função recebe o vetor de char onde contém a string com os dados do arquivo como primeiro argumento e o char que representa em qual caractere que será quebrado, que no contexto atual, é o caractere de vírgula e retorna um ponteiro de char onde armazenará o endereço de memória dessa nova string. Cada laço dentro do loop representa uma linha do arquivo que será quebrada onde houver as vírgulas e assim é possível ir armazenando dentro das variáveis respeitando a sequência em como foi gravado as informações na ordem que foram salvas antes. No fim do loop, é utilizado a função printf() para ir exibindo as informações lidas do arquivo na saída do console conforme a figura a seguir :



5.3.9 SendPatient()

Método sem retorno que funciona de forma muito semelhante ao método GetPatient() onde se utiliza a função fopen() para abrir a conexão com o arquivo, a função fgets() e um loop While para ler linha a linha do arquivo. Entretanto, a cada linha lida que representa um paciente cadastrado, é feito uma condicional IF com os dados do paciente verificando se o paciente é maior que 65 anos e contém alguma comorbidade a função CreatePatientFile é chamada passando a Struct de pacientes, o endereço do novo arquivo que se chama “PacientesEnviados.txt” e inteiro 2 que indica que será salvo apenas o nome e o CEP do paciente que será enviado para a secretaria da saúde. Em seguida é exibido na tela os dados de todos os pacientes que forma enviados para a secretaria da saúde. Se nenhum paciente se encaixar nos requisitos, uma mensagem é retornada ao usuário que nenhum paciente cadastrado se encaixa nas requisições de serem enviados ao ministério da saúde.

**6. Testes \***

\*Matheus

**7.Implementação e manutenção**

\*Matheus

**8. Conclusão**

\*Hugo