1. Introdução

Este trabalho tem como objetivo a criação e a apresentação de um sistema, composto por uma API Java e por um banco de dados relacional, para monitoramento de casos de covid-19, sendo feito o cadastro de pacientes, para que os dados guardados sejam acessíveis mais facilmente e o cruzamento de dados seja possível para que, a tomada de decisões em relação ao combate à pandemia seja feita de forma mais rápida e organizada.

1. Proposta inicial do sistema

O objetivo principal era criar um sistema para o monitoramento de casos de covid-19, para isso foi feito o cadastramento de usuários que podem ser pacientes, gerentes de empresa e/ou médicos, catalogando dados dos pacientes e empresas para gerar relatórios os quais estariam disponíveis para os médicos e alguns também estariam disponíveis para os gerentes de empresas.

* 1. Etapas do projeto

A primeira etapa para o desenvolvimento do sistema foi o desenvolvimento do modelo relacional do banco de dados. Feito isso, era preciso desenvolver a API em Java após o desenvolvimento da API foi desenvolvido a modelagem UML seguindo a estrutura criada no Java, por fim criamos scripts SQL para testes e scripts de “*backup*” para o banco de dados.

1. Modelo relacional

Para o desenvolvimento do modelo relacional utilizamos o software Visual Paradigm Community e levamos em conta os seguintes requisitos:=

* É necessário cadastrar as empresas com CNPJ e demais dados. O paciente está associado a uma empresa;
* Cadastrar os dados em geral, gerar relatórios;
* Para cada paciente/usuário deve ser aplicado um questionário para abordar desde dados pessoais como peso, a histórico de comorbidades e sintomas do momento;
* É necessário informar a empresa a qual o paciente está vinculado;
* Também devem ser registrados os tipos de usuário, sejam eles administradores, líderes, pacientes etc.;
* Para cada usuário devem ser coletadas informações de geolocalização;
* Febre, tosse, falta de ar, dor no corpo, dor de garganta, calafrio, dor muscular, congestão nasal e coriza são sintomas que devem ser informados no aplicativo, com detalhamento de intensidade (pouco, moderado ou constante), sempre que alterações forem observadas;

COLOCA AS FOTOS DO DIAGRAMA E EXPLICA ELAS RESUMIDAMENTE (fala das unicidades, nulidades e bota uma conclusão)

1. API Java

Uma API nada mais é do que uma ponte de comunicação entre usuário e banco de dados e no nosso caso utilizamos a linguagem Java e o framework SpringBoot para o desenvolvimento da API.

Em Java desenvolvemos as classes seguindo o modelo relacional e utilizamos o Java Persistence para criar o banco de dados diretamente pelo Java sem a necessidade de um script de criação, contudo o Java Persistence não nos fornecia todo o suporte necessário então para complementação utilizamos o Liquibase que ficou responsável pela adição dos comentários nas colunas de cada tabela, pela criação das Stored Procedures e Triggers necessárias, criação dos grupos e das permissões de acesso para cada grupo além disso para testes adicionamos ao Liquibase um script de inserção que faz com que o banco já seja criado com informações de teste dentro dele.

Utilizamos o SGBD PostgreSQL para controlar nosso banco de dados, e para fazer a comunicação do banco com o Java utilizamos a interface do JPARepository que já vem com vários comandos prontos e nos permite uma fácil e rápida criação de novos comandos.

Na parte de criação de “*endpoints”* para a comunicação web desenvolvemos todos as funções necessárias para a utilização do banco por um usuário, questões como adição de países, estados, cidades, sintomas e comorbidades ainda devem ser feitas diretamente por SQL pois estas informações não podem e não devem ser adicionadas por usuários comuns somente por administradores que controlarão o banco de dados diretamente por SQL.

Para as respostas e validação de erros que um usuário pode causar criamos algumas classes de respostas para os “endpoints” que possuem a maior chance de erros acontecerem e para ocultação e conversão de dados utilizamos classes de schema para converter a informação para a maneira que desejamos.

Para funções de atualização de dados optamos por não exigir que o usuário nos informe todos os dados novamente, apenas nos informe o dado que necessita de alteração, porém para permitir isso precisamos criar classes de preenchimento onde ocorre a comparação dos dados já inseridos com o novo dado repassado e preenche o que não foi informado.

Também adicionamos algumas funcionalidades extras como a possibilidade de exportar e importar arquivos JSON, a importação funciona apenas para usuários e para empresas, e também envio de e-mails para o usuário quando ele se cadastra altera dados ou exclui seu cadastro.

Para a geração de relatórios feita pelo sistema criamos classes modelo onde os dados vindos do banco são mapeados para dentro dessas classes e assim podem ser exibidos ao usuário.

COLOCA AS FOTOS DO Java (explica resumidamente e bota uma conclusão)

1. Modelagem UML

Assim como na modelagem relacional utilizamos o Visual Paradigm para o desenvolvimento da modelagem UML que nada mais é que colocar toda a estrutura do projeto em formato de diagramas para auxiliar no desenvolvimento.

Os diagramas UML no nosso caso foram desenvolvidos por último pois para desenvolver a engenharia de um software o engenheiro precisa conhecer os recursos disponíveis e como usá-los e nós estávamos testando os recursos pela primeira vez diretamente no trabalho então a modelagem precisou esperar até nós entendermos como o sistema iria funcionar.

Na modelagem UML foram desenvolvidos uma serie de diagramas de casos de uso que expressas as utilizações que o sistema terá e o quem será o ator responsável por aquela utilização, forma desenvolvidos também diagramas de sequência que expressam a sequência que ocorrerá para a utilização de cada caso de uso, também foram desenvolvidos diagramas de atividade onde você expressa cada atividade que será realizada em cada caso de uso e as ordens em que as atividades ocorrerão, foram desenvolvidos diagramas de máquina estado que mostram o passo a passo para a execução de um caso de uso e a situação de cada parte do programa em cada momento do caso de uso e por fim foi feito o diagrama de classes que representa as classes da API desenvolvida em Java.

Foram desenvolvidos também fluxos de caso de uso que representam o passo a passo da utilização de cada caso de uso e suas possíveis alternativas caso algo fora do normal aconteça.

COLOCA AS FOTOS DOS DIAGRAMAS (explica resumidamente e bota uma conclusão)

1. Backup

O backup de um banco de dados nada mais é do que salvar os dados contidos naquele banco para caso ocorra algo com o banco você possa restaurá-lo sem problemas, porém quando se trata de um banco de dados que tem os dados em constante transformação é necessário ter uma certa periodicidade na criação desse backup.

Como o nosso banco trata de informações de saúde optamos por definir uma regra de backup mais severa, definimos que o backup será feito de 30 em 30 minutos, mas para não ficar muito pesado para o servidor optamos também do tipo do backup ser do tipo texto, esse backup é mais leve para ser feito e ocupa menos espaço em disco, visto que um backup a cada 30 minutos acaba gerando um grande volume de arquivos.

COLOCA AS FOTOS DOS arquivos bat (explica resumidamente e bota uma conclusão)

1. Utilização do sistema

Como não foi desenvolvido uma interface para o sistema a sua utilização pode ser feita por softwares de comunicação web como o Postman, ou diretamente por linguagem SQL porém se você utilizar diretamente por SQL você perderá algumas funcionalidades que estão disponíveis somente através da API Java.

Queremos ressaltar que pelo fato de não ter uma interface para utilizar você sempre precisará informar um código que referencie você ou seja o seu código de paciente, o seu CPF, o seu CRM ou o CPNJ da sua empresa.

COLOCA AS FOTOS DO POSTMAN (explica resumidamente)

* 1. Cadastro de usuários, médicos, empresas e pacientes

O cadastro de usuários e feita em uma tabela que necessita das seguintes informações: CPF, nome, senha, e-mail, data de nascimento, sexo, telefone, CNPJ da empresa que trabalha, cep, rua, número da residência, código da cidade onde mora.

O cadastro de médicos ocorre na mesma tabela usuário, porém para o devido referenciamento de que aquele usuário é um médico, há uma tabela onde o CPF do usuário é referenciado com o CRM do médico.

O cadastro de empresas necessita que o gerente dela esteja cadastrado no sistema assim ele poderá cadastrar a própria empresa fornecendo os seguintes dados: CNPJ, nome da empresa, nome fantasia da empresa, telefone principal da empresa, telefone alternativo da empresa, e-mail da empresa, ramo da empresa, código da cidade, cep, número da residência e rua, CPF do usuário gerente da empresa.

O cadastro de pacientes assim como empresa e médico exige que o usuário esteja cadastrado no sistema para que o referenciamento de usuário-paciente possa ocorrer, estando cadastrado no sistema o usuário pode cadastrar-se como paciente informando os seguintes dados: CPF do usuário, peso, situação e se é ou não do grupo de risco.

* 1. Cadastro de monitoramentos, testes de covid, vacinas e vinculação com comorbidades

O cadastro de monitoramentos é feito pelo próprio paciente ou pelo médico, onde ele informa a data do monitoramento se quiser pois a data padrão é a data do cadastro, a intensidade do sintoma que está sentindo, o código do sintoma que está sentindo e o código do paciente.

O cadastro de testes de covid também é feito pelo próprio paciente basta ele informar a data do teste se quiser pois a data padrão é a data do cadastro, o resultado do teste e o código do paciente.

O cadastro de vacinas só pode ser feito por médicos e para cadastrar a vacina basta o médico informar o código do paciente que vai tomar a vacina, a data da vacina, a dose da vacina, a fabricante da vacina se quiser e o CRM do médico.

A vinculação com comorbidade é feita pelo próprio paciente ele só precisa informar o código da comorbidade que deseja vincular e o seu código de paciente.

* 1. Envio de solicitações

O envio de solicitações é feito pelo usuário e para enviar uma solicitação basta informar o seu CPF e escrever a solicitação.

1. Regras de utilização

Devido ao banco de dados ser relacional e também ao fato da própria organização do sistema foram criadas regras para a utilização, novamente quero ressaltar que algumas dessas regras não seriam necessárias caso tivesse interface, pois ela conseguiria barrar o erro no momento do uso.

* 1. Regras gerais
* O e-mail não pode ultrapassar 30 caracteres;
* O e-mail não pode ser igual a outro já cadastrado;
* O código da cidade informado deve existir no banco de dados;
* Os nomes não podem ultrapassar 100 caracteres;
* Os telefones não podem ultrapassar 20 caracteres;
* As datas devem ser no formato “AAA-MM-DD”;
* A senha não pode ultrapassar 20 caracteres;
* A rua informada não pode ultrapassar 60 caracteres;
* O número da residência informado não pode ultrapassar 10 caracteres;
  1. Regras de usuário
* O CPF não pode ser igual ao de outro usuário já cadastrado;
* O sexo só pode ser Masculino ou Feminino, isto é, “M” ou “F” respectivamente;
* O CNPJ da empresa que trabalha deve existir no banco de dados.
  1. Regras de empresa
* O nome fantasia é opcional;
* O telefone alternativo é opcional;
* O ramo da empresa não pode ultrapassar 60 caracteres;
* O CPF do usuário dono da empresa deve existir no banco de dados;
  1. Regras de paciente
* O CPF do usuário deve existir no banco de dados;
* O usuário não pode se cadastrar duas vezes como paciente;
* O peso não pode ultrapassar 999,99 Kg;
* A situação do paciente deve ser uma das seguintes opções:
* ISOLAMENTO;
* INTERNADO;
* BEM.;
* O paciente pode ou não estar no grupo de risco, isto é, “S” ou “N” respectivamente;
  1. Regras de teste de covid
* O código do paciente deve existir no banco de dados;
* A data do teste é opcional;
* O resultado do teste só pode ser positivo ou negativo, isto é, “P” ou “N” respectivamente;
  1. Regras de monitoramentos
* A intensidade do sintoma só pode ser “S” – sem sintomas, “P” – pouco, “M” – moderada e “C” – constante;
* O código do sintoma deve existir no banco de dados;
* O código do paciente deve existir no banco de dados;
  1. Regras de vinculação de comorbidades
* O código da comorbidade deve existir no banco de dados;
* O código do paciente deve existir no banco de dados;
* Não pode ser cadastrada comorbidade repetida no mesmo paciente;
  1. Regras de vacina
* A dose da vacina só pode ser a primeira ou a segunda, isto é, 1 ou 2 respectivamente;
* A fabricante da vacina é opcional;
* O código do paciente deve existir no banco de dados;
* O CRM do médico deve existir no banco de dados;
* Não pode ser cadastrado doses repetidas no mesmo paciente;
  1. Regras de solicitação
* O CPF do usuário informado deve existir no banco de dados;
* A descrição da solicitação não pode estar vazia;

1. Conclusão

Tire suas próprias conclusões