

Universidade do Oeste de Santa Catarina - UNOESC

Trabalho Final: PandemicStats

São Miguel Do Oeste/SC

2021

UNIVERSIDADE DO OESTE DE SANTA CATARINA - UNOESC
CAMPUS SÃO MIGUEL DO OESTE

Banco de Dados: PandemicStats

Trabalho apresentado às Unidades Curriculares de Banco de Dados II, Programação II e Engenharia de Software I do Curso de Ciência da Computação da Universidade do Oeste de Santa Catarina - UNOESC, Campus São Miguel do Oeste/SC.

Professores Orientadores: Roberson Junior Fernandes Alves, Otilia Donato Barbosa e Franciele Carla Petry.

Alunos: Denis Felipe Grezele, Eduardo Mateus da Costa, César Augusto Schuck Klunk, Murilo Ferrari Angeli, Ivanilson Schwingel

São Miguel Do Oeste/SC

2021

ÍNDICE DE ILUSTRAÇÕES

Figura 1:Diagrama geral.....	7
Figura 2:Tabelas Paciente, Empresa e Usuário.....	9
Figura 3:Tabelas de Monitoramento	10
Figura 4: Tabela Usuário	11
Figura 5: Implementação em Java	13
Figura 6: Diagrama de Caso de uso geral.....	14
Figura 7: Diagrama de Estado de Máquina - Cadastro	15
Figura 8:Diagrama de Classes.....	16
Figura 9: Diagrama de Atividades funções da Empresa.....	16
Figura 10: Diagrama de Sequência da Empresa.....	17
Figura 11: Implementação das tabelas em SQL.....	18
Figura 12 Configuração de Backup e Restore.....	19
Figura 13: Dicionário de Dados	24

Sumário

ÍNDICE DE ILUSTRAÇÕES.....	3
1. INTRODUÇÃO	5
2. PROPOSTA INICIAL DO BANCO DE DADOS.....	6
3. MODELO RELACIONAL.....	7
4. DESENVOLVIMENTO DA API EM JAVA.....	12
5. MODELAGEM UML.....	14
6. IMPLEMENTAÇÃO EM LINGUAGEM SQL	17
7. BACKUP.....	19
8. UTILIZAÇÃO DO SISTEMA.....	19
9. REGRAS DE SOLICITAÇÃO.....	21
10. DICIONÁRIO DE DADOS DO PROJETO.....	23
11. CONCLUSÃO	25
12. REFERÊNCIAS.....	26

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.

2. PROPOSTA INICIAL DO BANCO DE DADOS

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.

2.1 ETAPAS DO PROJETO

A primeira etapa para a criação do banco de dados de monitoramento do casos de covid-19 sistema foi o desenvolvimento do modelo relacional do banco de dados para exemplificar e o organizar o funcionamento do banco de dados, fazendo as devidas associações de entidades.

A próxima etapa foi a implementação da API em Java, com isso desenvolvemos o funcionamento do sistema com suas principais funcionalidades e recursos próprios. Apresentando no final a documentação correta do código utilizando as ferramentas do JavaDOC.

Tendo concluído isso, foi necessário criar a modelagem conceitual do sistema, criando os casos de uso e seus fluxos levando em conta os requisitos pré-definidos. Além disso, também montamos os diagramas de sequência, atividades, de estado e de classes.

Feito isso, foi preciso criar scripts para implementação do banco em linguagem SQL(Structured Query Language ou Linguagem de Consulta Estruturada), e implementar cada tabela com suas devidas ligações. Fazendo isso, foram gerados dados diversos para testar o funcionamento do sistema. E após isso, foram feitas algumas pesquisas para geração de relatórios usando comandos da linguagem SQL.

Além disso, também fizemos a criação de políticas de acesso adicionando usuários e concedendo privilégios, implementação de gatilhos(triggers) para o controle de regras de integridade, a criação de regras de negócio e política de backup e restore.

Para finalizar foi gerado o dicionário de dados das tabelas do modelo relacional para ter acesso às definições e representações dos elementos, sendo descritos de forma composta e apresentados em uma lista organizada.

3. MODELO RELACIONAL

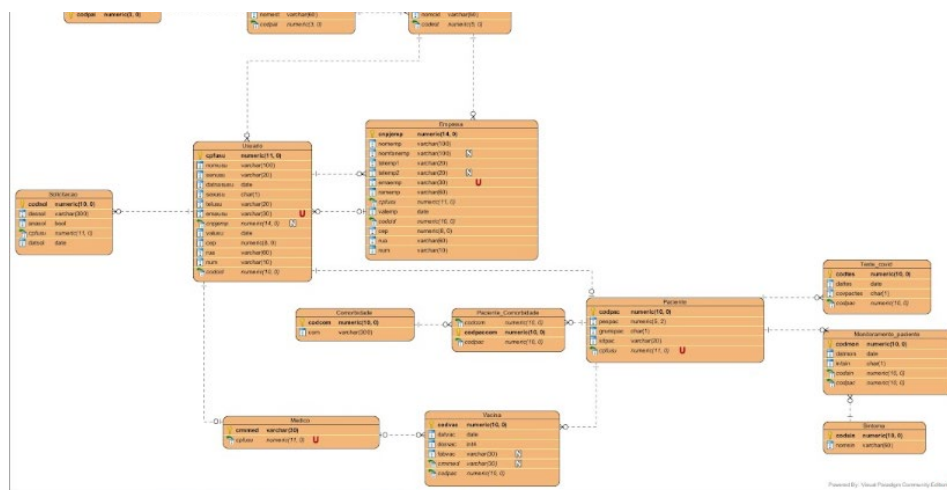
3.1 CADASTRAMENTO DE PACIENTES/USUÁRIOS E EMPRESAS

Para iniciar o projeto foi criado um modelo relacional, fazendo a construção de diversas tabelas, para que o cadastramento e cruzamento de dados se desse de forma organizada.

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;

Figura 1: Diagrama geral

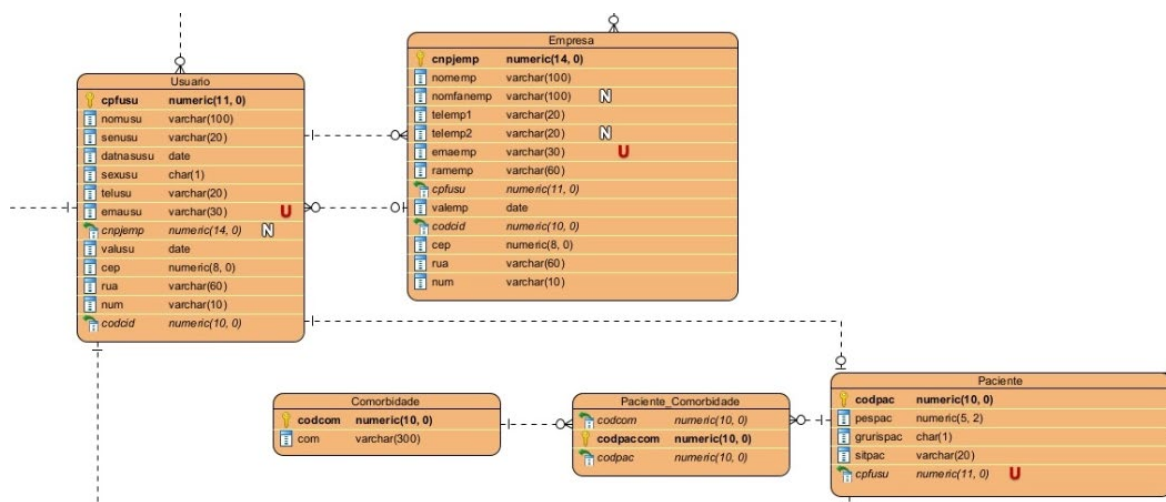


Na tabela “Paciente” são cadastradas as informações gerais dos pacientes/usuários, como, nome, cpf, se é do grupo de risco ou tem comorbidade, entre outros. Além disso, cada usuário é associado a uma empresa, para isso foi construída uma tabela “Empresa” devendo ser informada o nome da empresa seus dados gerais.

3.2 COMORBIDADES

Os pacientes cadastrados devem informar se são do grupo de risco ou possuem algum tipo de comorbidade e, caso possuam quais são elas. Essas comorbidades devem ser informadas e cada uma deve ser descrita, para que seja possível saber o nível de risco que cada paciente possui.

Figura 2: Tabelas Paciente, Empresa e Usuário



Fonte: Os Autores

3.3 CASOS SUSPEITOS

São analisados os dados dos pacientes e catalogados os casos considerados suspeitos para o vírus. Então é feito o acompanhamento dos pacientes para se ter um controle.

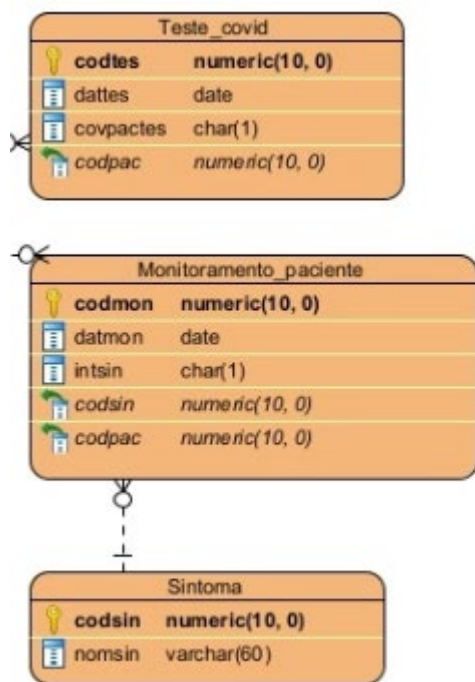
Comparando os dados e se alguns sinais mais críticos for descobertos, o sistema irá emitir alertas para que o paciente procure o serviço médico. Para serem feitos os testes e ter uma confirmação dos casos de covid-19 positivados.

3.4 SINTOMAS

Para a verificação dos sintomas dos casos suspeitos e casos confirmados os pacientes devem informar quais os sintomas estão sentindo como: febre, tosse, falta de ar, dor no corpo, dor de garganta, calafrio, dor muscular, congestão nasal e coriza. Além disso, deve ser informada qual a intensidade desses sintomas, ou seja, se for pouco,

moderado ou constante. Essas informações são analisadas nas tabelas “Monitoramento_paciente” e “Sintoma” e os dados são reenviados toda vez que houver alteração no quadro do paciente.

Figura 3:Tabelas de Monitoramento



Fonte: Os Autores

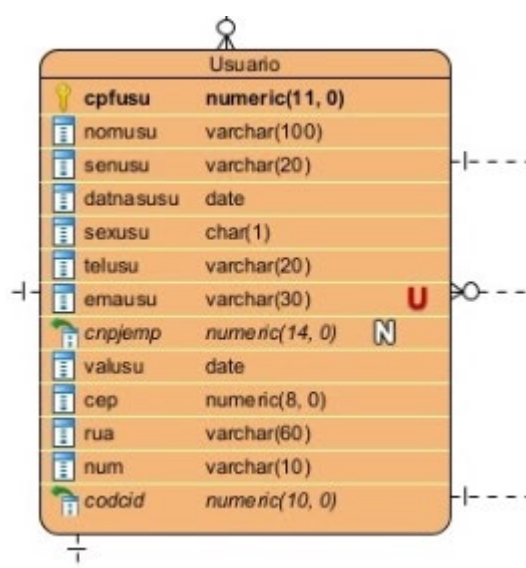
3.5 AJUDA E ORIENTAÇÕES

O sistema possui uma tabela de ajuda para trazer informações aos usuários. Essa tabela serve para que os pacientes possam tirar suas dúvidas em relação ao sistema e sobre a análise das informações.

3.6 TIPOS DE USUÁRIOS DO SISTEMA

O sistema contém a tabela “Usuario” que serve para que a pessoa se cadastre com um certo nível de acesso. Os tipos disponíveis são “Médico”, “Paciente” e “Empresa”.

Figura 4: Tabela Usuário



Fonte:

Os

Autores

4. DESENVOLVIMENTO DA API EM 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 todas 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.

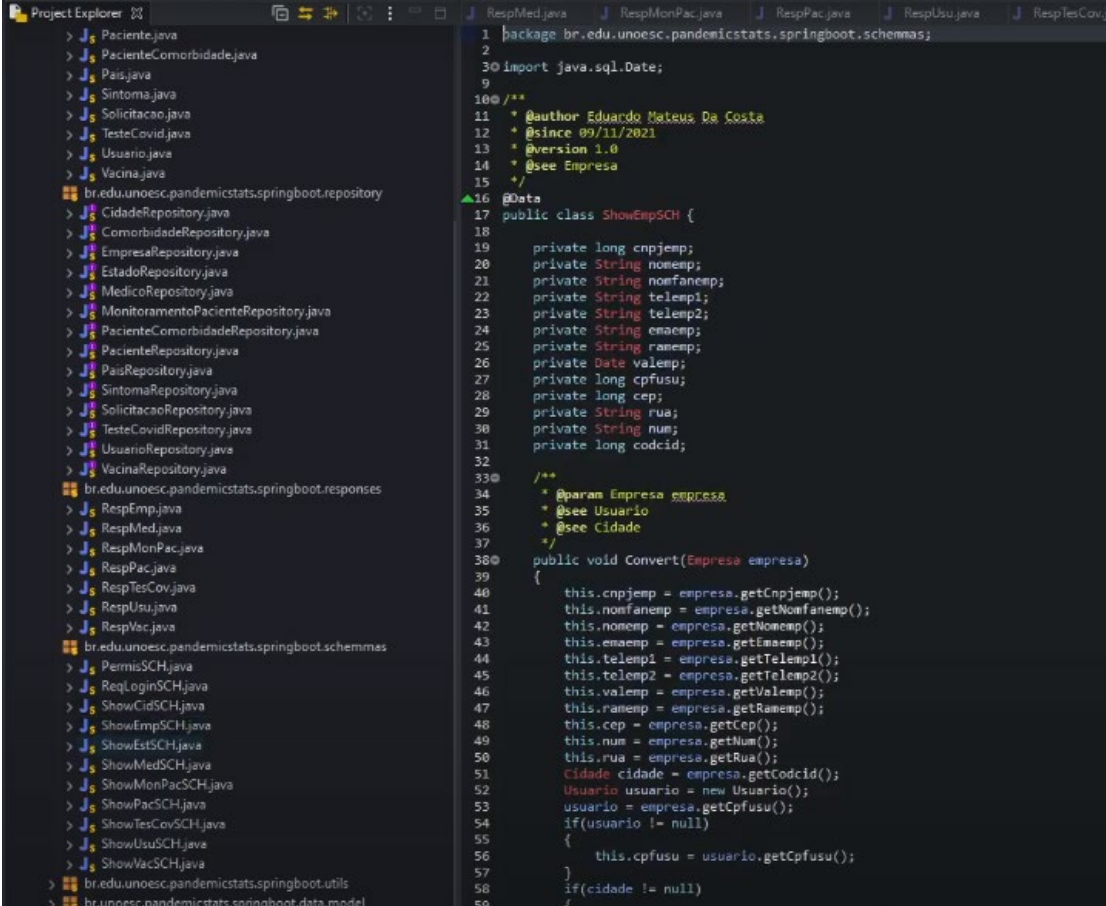
Para funções de atualização de dados optamos por não exigir que o usuário nos informe todos os dados novamente, apenas nos informe o dado que necessita de alteração, porém para permitir isso precisamos criar classes de preenchimento onde ocorre a comparação dos dados já inseridos com o novo dado repassado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

Figura 5: Implementação em Java



```
1 package br.edu.unoesc.pandemicstats.springboot.schemas;
2
3 import java.sql.Date;
4
5 /**
6  * @author Eduardo Mateus Da Costa
7  * @since 09/11/2021
8  * @version 1.0
9  * @see Empresa
10 */
11 @Data
12 public class ShowEmpSCH {
13
14     private long cnpjemp;
15     private String nomemp;
16     private String nomfanemp;
17     private String telemp1;
18     private String telemp2;
19     private String emaemp;
20     private String ramemp;
21     private Date valemp;
22     private long cpfusu;
23     private long cep;
24     private String rua;
25     private String num;
26     private long codcid;
27
28     /**
29      * @param Empresa empresa
30      * @see Usuario
31      * @see Cidade
32      */
33     public void Convert(Empresa empresa)
34     {
35         this.cnpjemp = empresa.getCnpjemp();
36         this.nomfanemp = empresa.getNomfanemp();
37         this.nomemp = empresa.getNomeemp();
38         this.emaemp = empresa.getEmaemp();
39         this.telemp1 = empresa.getTelemp1();
40         this.telemp2 = empresa.getTelemp2();
41         this.valemp = empresa.getValemp();
42         this.ramemp = empresa.getRamemp();
43         this.cep = empresa.getCep();
44         this.num = empresa.getNum();
45         this.rua = empresa.getRua();
46         Cidade cidade = empresa.getCodcid();
47         Usuario usuario = new Usuario();
48         usuario = empresa.getCpfusu();
49         if(usuario != null)
50         {
51             this.cpfusu = usuario.getCpfusu();
52         }
53         if(cidade != null)
54         {
55             //
56         }
57     }
58 }
59
```

Fonte: Os Autores

5. 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.

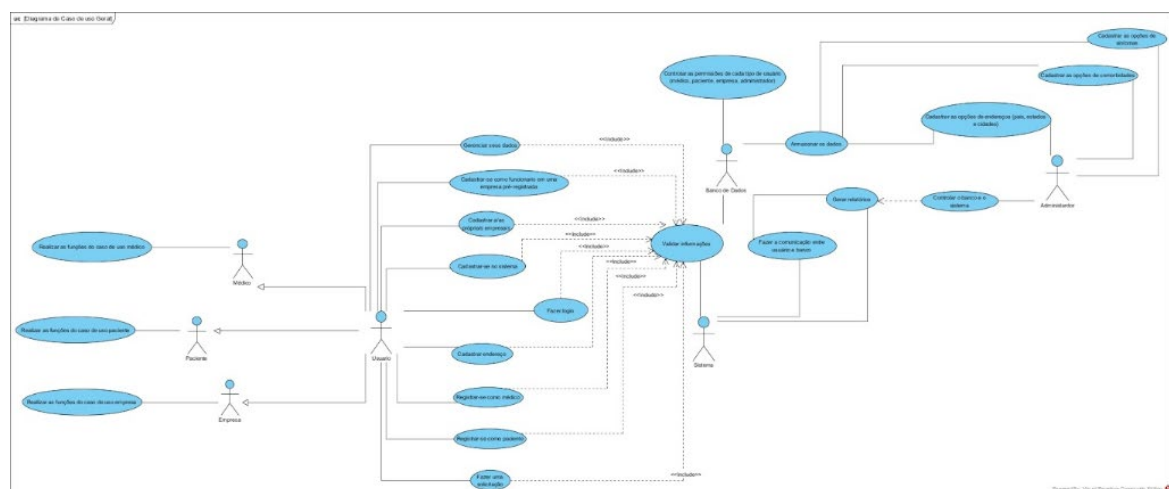
Os diagramas UML no nosso caso foram desenvolvidos por último pois para desenvolver a engenharia de um software o engenheiro precisa conhecer os recursos disponíveis e como usá-los e nós estávamos testando os recursos pela primeira vez diretamente no trabalho então a modelagem precisou esperar até nós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

Na modelagem UML foram desenvolvidos uma série de diagramas de casos de uso que expressam as utilizações que o sistema deveria ter e o quem seria o ator responsável por aquela utilização. Entre eles o de caso de uso Geral, o de Paciente, o do Médico e o da Empresa.

Foram desenvolvidos também, diagramas de sequência que expressam a sequência que ocorrerá para a utilização de cada caso de uso, diagramas de atividade onde você expressa cada atividade que será realizada em cada caso de uso e as ordens em que as atividades ocorrerão, diagramas de máquina de 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.

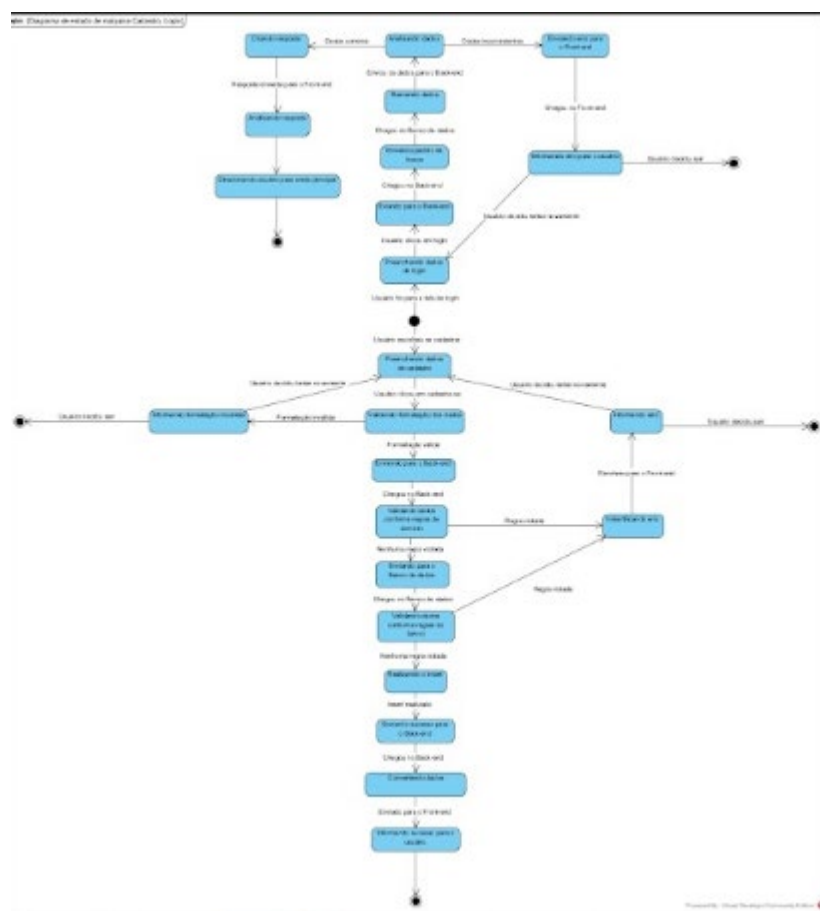
Além disso, fizemos os fluxos de caso de uso que representam o passo a passo da utilização de cada caso de uso e suas possíveis alternativas caso algo fora do normal aconteça.

Figura 6: Diagrama de Caso de uso geral



Fonte: Os Autores

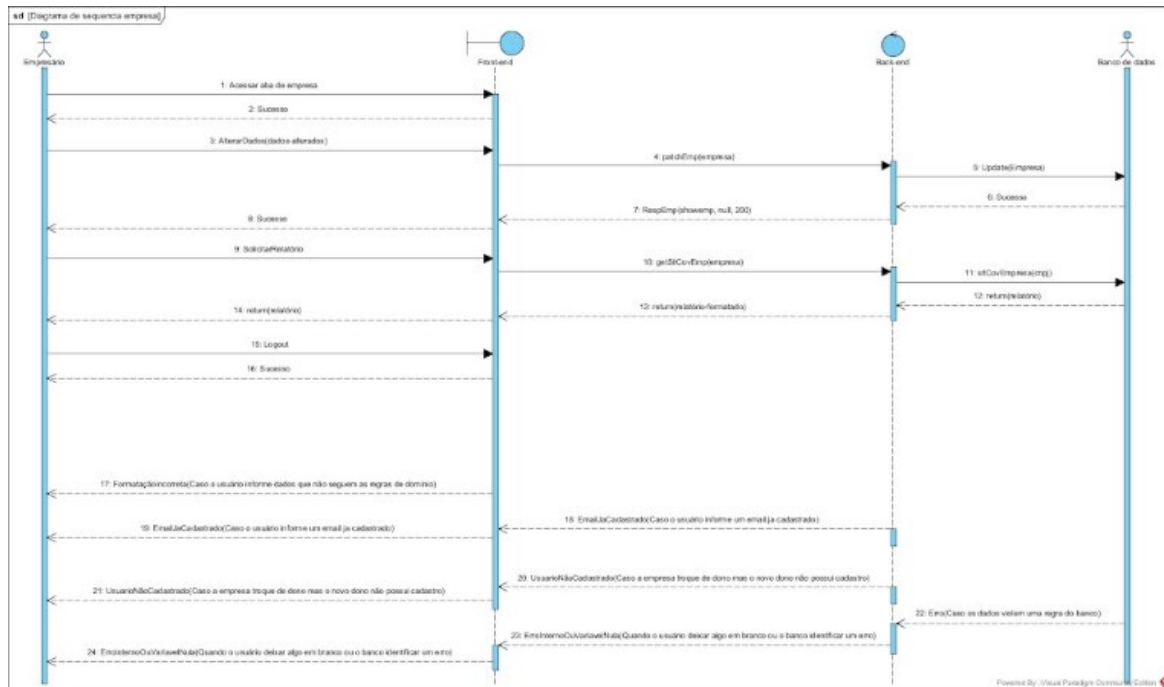
Figura 7: Diagrama de Estado de Máquina - Cadastro



Fonte: Os Autores

[illegible]

Figura 10: Diagrama de Sequência da Empresa



Fonte: Os Autores

6. IMPLEMENTAÇÃO EM LINGUAGEM SQL

Feito o Modelo Relacional, passamos para a fase de implementação das tabelas para a linguagem SQL, usando o programa DBeaver com conexão ao servidor PostgreSQL.

Cada tabela é criada e depois dela são adicionados os comentários sobre as informações de cada coluna.

Figura 11: Implementação das tabelas em SQL

```
--CRIAÇÃO
--create database pandemicstats;

create table Cidade (
    codcid numeric(10, 0) not null,
    nomcid varchar(60) not null,
    codest_codest numeric(5, 0) not null,
    primary key (codcid));

comment on table Cidade is 'Tabela de cidades';
comment on column Cidade.codcid is 'Código da cidade';
comment on column Cidade.nomcid is 'Nome da cidade.';

create table Comorbidade (
    codcom numeric(10, 0) not null,
    com    varchar(300) not null,
    primary key (codcom));

comment on table Comorbidade is 'Tabela de comorbidades';
comment on column Comorbidade.codcom is 'Código da comorbidade';
comment on column Comorbidade.com is 'Comorbidades e sua descrição se necessário.';

create table Empresa (
    nomemp    varchar(100) not null,
    cnpjemp   numeric(14, 0) not null,
    nomfanemp varchar(100),
    telemp1   varchar(20) not null,
```

Fonte:

Os

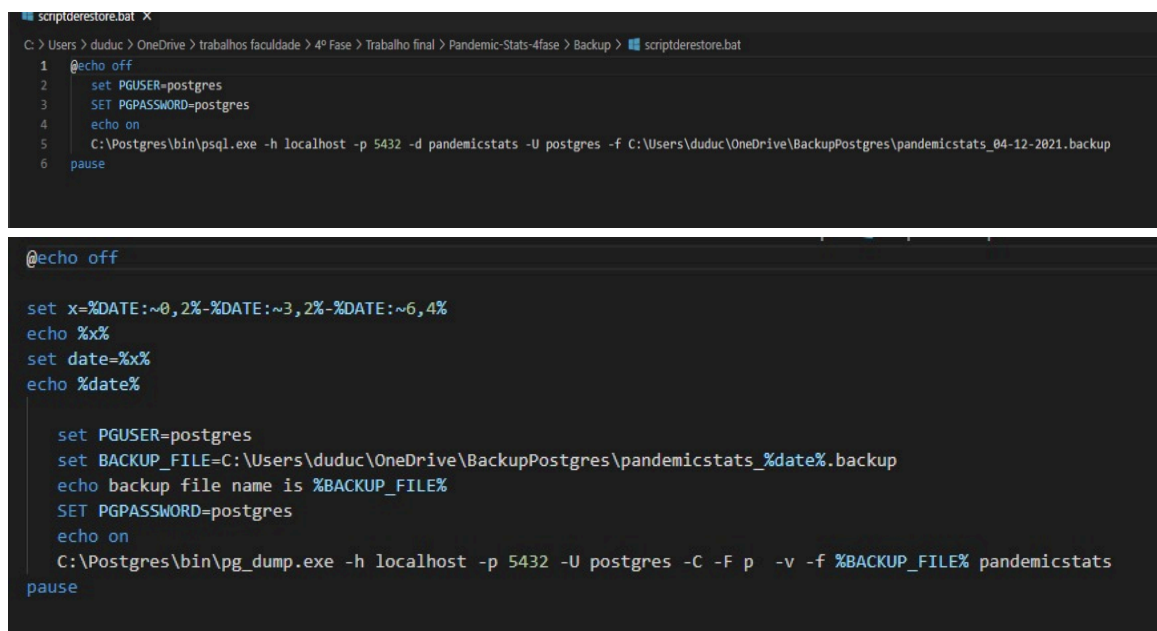
Autores

7. 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.

Figura 12 Configuração de Backup e Restore



```
scriptderestore.bat X
C:\Users\> dduc > OneDrive > trabalhos faculdade > 4º Fase > Trabalho final > Pandemic-Stats-4fase > Backup > scriptderestore.bat
1 @echo off
2 set PGUSER=postgres
3 SET PGPASSWORD=postgres
4 echo on
5 C:\Postgres\bin\psql.exe -h localhost -p 5432 -d pandemicstats -U postgres -f C:\Users\dduc\OneDrive\BackupPostgres\pandemicstats_04-12-2021.backup
6 pause

@echo off
set x=%DATE:~0,2%- %DATE:~3,2%- %DATE:~6,4%
echo %x%
set date=%x%
echo %date%

set PGUSER=postgres
set BACKUP_FILE=C:\Users\dduc\OneDrive\BackupPostgres\pandemicstats_%date%.backup
echo backup file name is %BACKUP_FILE%
SET PGPASSWORD=postgres
echo on
C:\Postgres\bin\pg_dump.exe -h localhost -p 5432 -U postgres -C -F p -v -f %BACKUP_FILE% pandemicstats
pause
```

Fonte: Os Autores

8. 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 CNPJ da sua empresa.

8.1 CADASTRO DE USUÁRIOS, MÉDICOS, EMPRESAS E PACIENTES

O cadastro de usuários é 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.

8.2 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.

8.3 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.

9. REGRAS DE SOLICITAÇÃO

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.

9.1 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;

9.2 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.

9.3 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;

9.4 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;

9.5 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;

9.6 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;

9.7 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;

9.8 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;

9.9 REGRAS DE SOLICITAÇÃO



- O CPF do usuário informado deve existir no banco de dados;
- A descrição da solicitação não pode estar vazia;

10. DICIONÁRIO DE DADOS DO PROJETO

Após todas as tabelas terem sido implementadas em SQL, foi feito o Dicionário de Dados do Modelo Relacional através do Dbeaver, que contém uma lista com as informações específicas de cada coluna em cada tabela do modelo.

Figura 13: Dicionário de Dados

1. Data Dictionary

Entity Name	Entity Description					
Column Name	Column Description	Data Type	Length	Primary Key	Nullabl e	Unique
 Cidade	Tabela de cidades					
codcid	Código da cidade	numeric	10	true	false	false
codest		numeric	5	false	false	false
nomcid	Nome da cidade.	varchar	60	false	false	false
 Comorbidade	Tabela de comorbidades					
codcom	Código da comorbidade	numeric	10	true	false	false
com	Comorbidades e sua descrição se necessário.	varchar	300	false	false	false
 Empresa	Tabela de empresas					
cep	Número do CEP.	numeric	8	false	false	false
cnpjemp	CNPJ da Empresa	numeric	14	true	false	false
codcid		numeric	10	false	false	false
cpfusu		numeric	11	false	false	false
emaemp	Email da empresa	varchar	30	false	false	true
nomemp	Nome da empresa	varchar	100	false	false	false

Fonte: Os Autores

Com o dicionário de Dados podemos reconstruir o contexto em que os dados foram coletados melhorando a análise de dados e servindo como um ponto de partida, de forma objetiva, sem ambiguidades. (HOPPEN; PRATES; SANTOS, 2017).

11. CONCLUSÃO

As investigações a respeito do estudo sobre a criação e o funcionamento de um sistema de monitoramento de casos do coronavírus demonstrou a importância e o enorme potencial que os bancos de dados tem a proporcionar, principalmente por ser uma ferramenta que atua de forma estatística para trazer quaisquer dados desejados em relação aos pacientes cadastrados, casos de covid-19 e informações de forma mais crítica em relação aos sintomas mais comuns, além disso, mostra também a importância da utilização de diagramas para a criação de um sistema, bem como a implementação em Java aliada a boas práticas de programação.

12. REFERÊNCIAS

ALVES, Roberson Junior Fernandes. **Apostila_Banco_Dados**. 2021. Não publicado. Acesso em: 25 de nov 2021.

BARBOSA, Otília Donato. **6 - Postman**. 2021. Não publicado. Acesso em: 27 nov. 2021.

BARBOSA, Otília Donato. **Parte 3 - Anotações, JavaDoc e JUnit**. 2021. Não publicado. Acesso em: 27 nov. 2021.

BARBOSA, Otília Donato. **Parte 9 - Spring Boot REST API**. 2021. Não publicado. Acesso em: 27 nov. 2021.

HOPPEN, Joni; PRATES, Wlademir Ribeiro; SANTOS, Marcos. **O que é um dicionário de Dados de Data Analytics**: importância da dicionarização. Importância da Dicionarização. 2017. Disponível em: <<https://www.aquare.la/o-que-e-um-dicionario-de-dados-de-data-analytics/>>. Acesso em: 01 jul. 2021.

PETRY, Franciele Carla. **Aula05_IntroUML**. 2021. Não publicado. Acesso em: 29 nov. 2021.

PETRY, Franciele Carla. **Aula06_DiagSequencia**. 2021. Não publicado. Acesso em: 29 nov. 2021.

PETRY, Franciele Carla. **Aula07_DiagAtividades_Estados**. 2021. Não publicado. Acesso em: 29 nov. 2021.

PETRY, Franciele Carla. **Aula08_Diagrama_Classes**. 2021. Não publicado. Acesso em: 29 nov. 2021.

SOMMERVILLE, Ian. **Engenharia de Software**. 9 ed. São Paulo: Pearson Addison-Wesley, 2011. o Capítulo 4 – Engenharia de Requisitos. Acesso em 27 nov. 2021.