

Pandemic: Sistema COVID-19

Aluno(a): André Luis

Aluno(a): Alessandro Vaiz

Aluno(a): Dieferson Romanoski

Orientador(a): Franciele Carla Petry

*Discente do Curso de Ciência da Computação
Unoesc-Campus de São Miguel do Oeste
Rua Oiapoc, 2011. São Miguel do Oeste-SC
andreluismoreirasmo@gmail.com

** Discente do Curso de Ciência da Computação
Unoesc-Campus de São Miguel do Oeste
Rua Oiapoc, 2011. São Miguel do Oeste-SC
alessandrovaiz@gmail.com

*** Discente do Curso de Ciência da Computação
Unoesc-Campus de São Miguel do Oeste
Rua Oiapoc, 2011. São Miguel do Oeste-SC
diefek-i@outlook.com

*** Mestre em Informática
Docente do Curso de Ciência da Computação
Unoesc-Campus de São Miguel do Oeste
Rua Oiapoc, 2011. São Miguel do Oeste-SC

SUMÁRIO

Introdução.....	3
Desenvolvimento do software:.....	2-8
Casos de uso e fluxos:	10-14
Diagrama de Sequência:	15
Diagrama de Classes:	16
Modelo do banco.....	17-18
Scripts do banco.....	19
Conclusão.....	23

1 Introdução

O presente trabalho foi desenvolvido junto às disciplinas de Programação IV, Banco de Dados II e Engenharia de Software II. ministrada no Curso de Ciência da Computação da UNOESC, onde ao longo do 1º Semestre do ano de 2021 vem-se trabalhando num projeto que se destina a apresentar os estudos feitos em cima do vírus COVID-19.

Apresentando um Sistema de Monitoramento de Casos e Estatísticas do Coronavírus escrito na Linguagem Flutter e Python, Utilizando-se uma base de dados em tempo real do coronavírus, o sistema faz o controle de pacientes, orientações ao COVID e estatísticas.

2 Desenvolvimento do software: Login

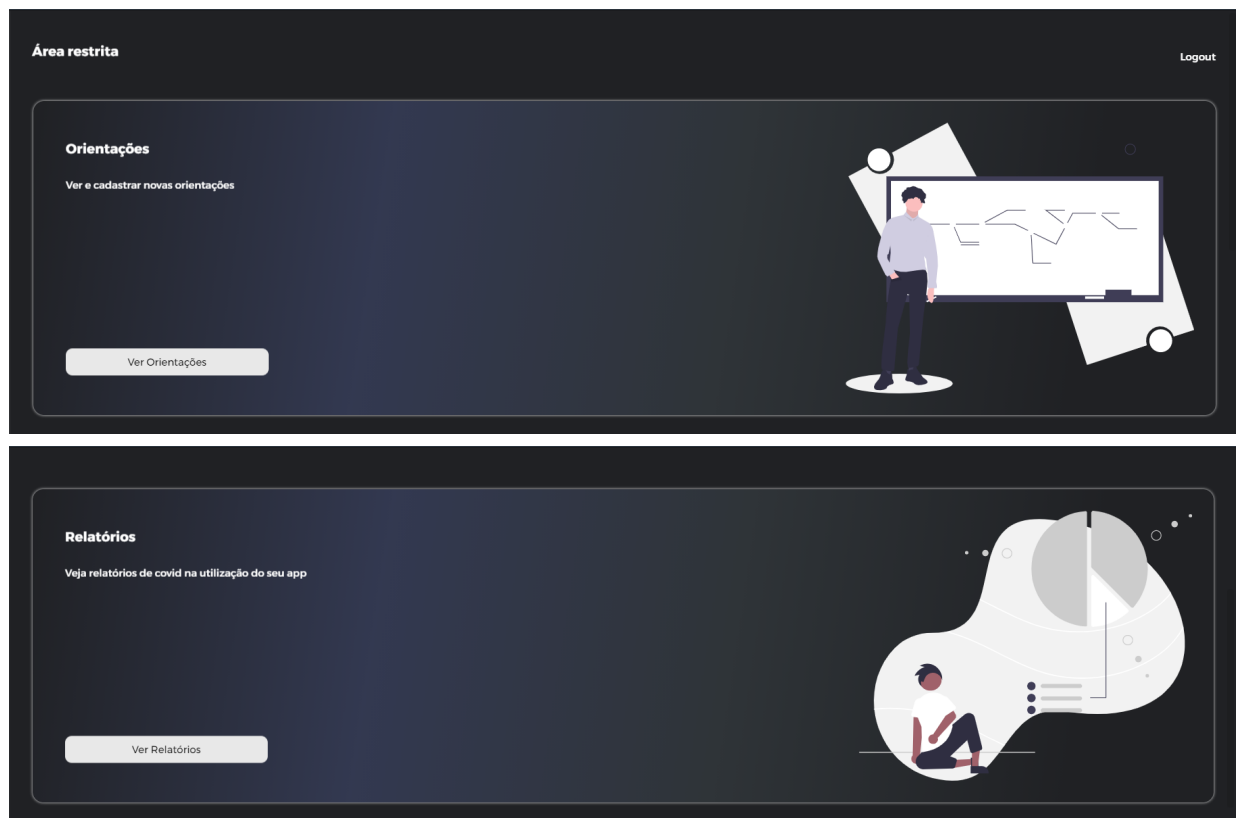
Foi desenvolvido uma área restrita, onde somente pessoas autorizadas possuem acesso, foi realizado uma forma de login, com controle de usuário e senha utilizando autenticação com o banco de dados.

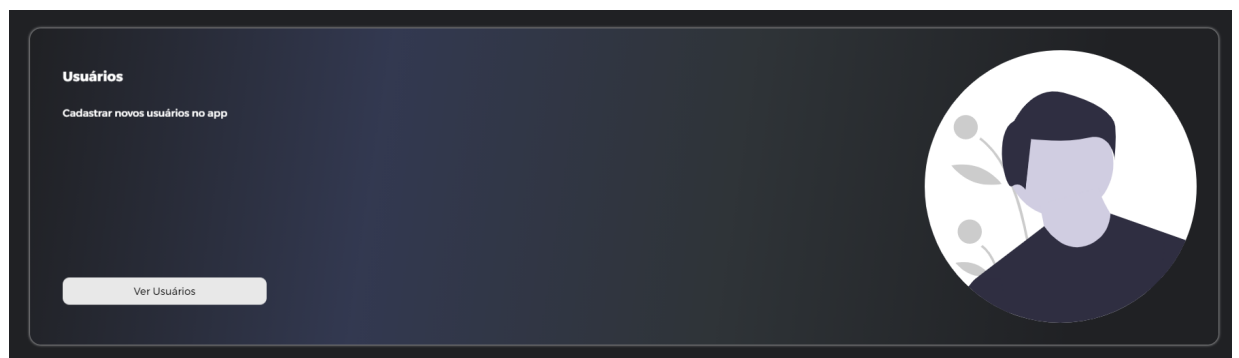
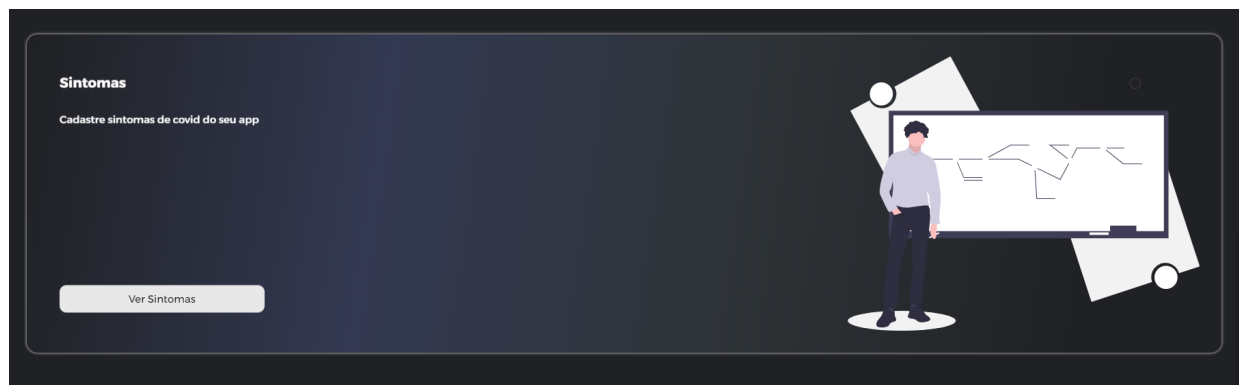


A autenticação é realizada com o backend em python, se o usuário é válido é gerado um token para o usuário pode fazer manutenções no sistema.

2.1 Desenvolvimento do software: Cadastros

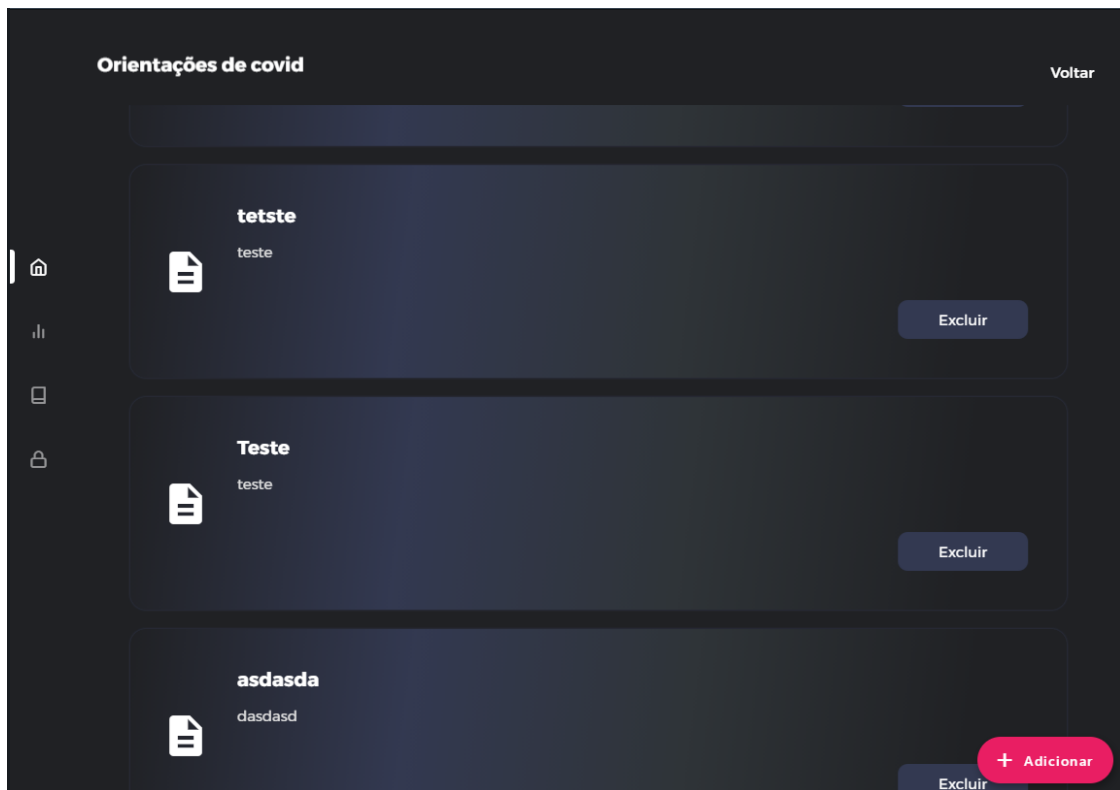
Nos cadastros, os administradores do sistema podem fazer as principais operações do sistema, com acesso às Orientações, Relatórios, Sintomas e Usuários.





2.2 Desenvolvimento do software: Orientações de covid, área administrativa

Orientações de covid, área administrativa, é onde o funcionário consegue verificar, excluir e inserir no sistema novas orientações.



2.3 Desenvolvimento do software: Cadastro de sintomas dos pacientes

Nessa tela é possível que qualquer pessoa possa informar os sintomas que estão sentindo.

The image shows a web registration form titled "Cadastro" on a dark background. The form is white and contains several input fields and a button. On the left side of the dark background, there is a vertical sidebar with four icons: a home icon, a bar chart, a document, and a lock. The form itself has a title "Cadastro" in blue. Below the title, there are two columns of input fields. The first column has fields for "Nome" (containing "João"), "Email" (containing "exemplo@gmail.com"), and "Senha" (containing "*****"). The second column has fields for "Sobrenome" (containing "Da Silva"), "Seleção de gênero" (with radio buttons for "Masculino" and "Feminino", where "Masculino" is selected), and "CPF" (containing "000.000.000-00"). Below these fields, there is a section titled "Seus sintomas" in blue. It contains a button with a heart rate icon and the text "Sintomas Selecione" followed by a right arrow. To the right of this button, the text "Suspeita Baixa" is displayed in yellow. At the bottom of the form, there is a large blue button with the text "CADASTRAR" in white.

Cadastro

Nome
João

Sobrenome
Da Silva


Email
exemplo@gmail.com

Seleção de gênero
☒ Masculino ☐ Feminino

Senha

CPF
000.000.000-00

Seus sintomas

 Sintomas Selecione >

Suspeita Baixa

CADASTRAR

2.4 Desenvolvimento do software: Relatórios e gráficos

Essa página tem o objetivo de demonstrar os dados de uma API externa que reúne dados do

mundo todo.



3 Casos de uso e fluxos: Cadastro de dados

Nome	Cadastro de Dados
Atores	Administrador
Resumo	Cadastro de dados em geral do sistema ex : orientações,usuários e sintomas.
Pré-Condições	Existir um usuário administrador previamente cadastrado
Pós-Condições	
Fluxo Principal	
<p>O usuário irá logar na área restrita e a api irá autenticá lo</p> <p>O usuário entra no módulo de cadastro e seleciona o tipo de cadastro</p> <p>O usuário cadastra as informações</p>	
Fluxo Alternativo(1) Editar cadastros	
Cadastro já existe no sistema, então não pode realizar cadastro novamente, apenas editar o cadastro já realizado	
Fluxo Alternativo(2) Excluir cadastros	
Em caso de arrependimento da criação do cadastro é possível deletá-lo	

3.1 Casos de uso e fluxos: Relatórios

Nome	Relatórios
Atores	Usuário,Sistema
Resumo	Gera relatórios e exibe estatísticas no sistema
Pré-Condições	Existir pacientes cadastrados
Pós-Condições	
Fluxo Principal - Relatórios	
<p>O usuário após logado irá acessar o menu relatórios</p> <p>Deverá selecionar o tipo de relatório desejado(por cidade ou por região) e então o sistema irá exibir uma lista com os dados</p> <p>Esse menu não será visível para pacientes</p>	
Fluxo Alternativo(1) - Estatísticas	
Ao acessar a página qualquer um irá acessar o menu estatísticas e então o sistema irá exibir os dados de covid no mundo através de uma API	

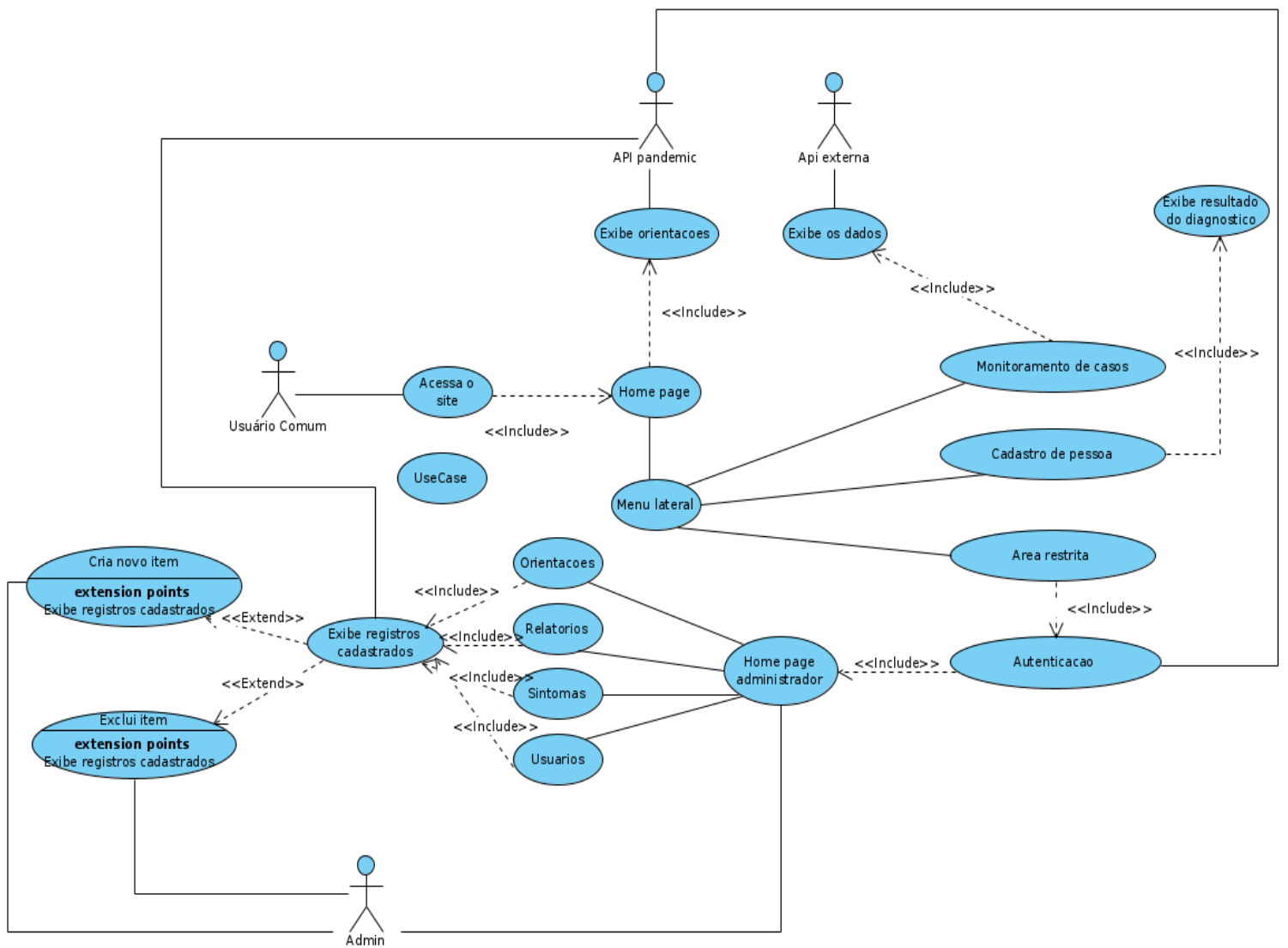
3.2 Casos de uso e fluxos: Formulário para usuário comum do site

Nome	Questionário ao paciente
Atores	Usuário comum
Resumo	<p>Irá solicitar nome e alguns outros dados da pessoa que está acessando o app</p> <p>Após isso irá pedir os sintomas e informar a pessoa como ela deve prosseguir dada a situação.</p>
Pré-Condições	
Pós-Condições	<p>Gravar o formulário preenchido pela pessoa na base de dados.</p> <p>Posteriormente esses dados serão exibidos na tela de relatórios que o administrador do sistema terá acesso.</p>
Fluxo Principal - Preencher formulário	
<p>O paciente irá acessar a página e acessar o menu de diagnóstico</p> <p>Ao informar seus sintomas o sistema irá exibir como deve prosseguir dada a situação</p> <p>As informações serão gravadas no banco e relatórios serão gerados.</p>	

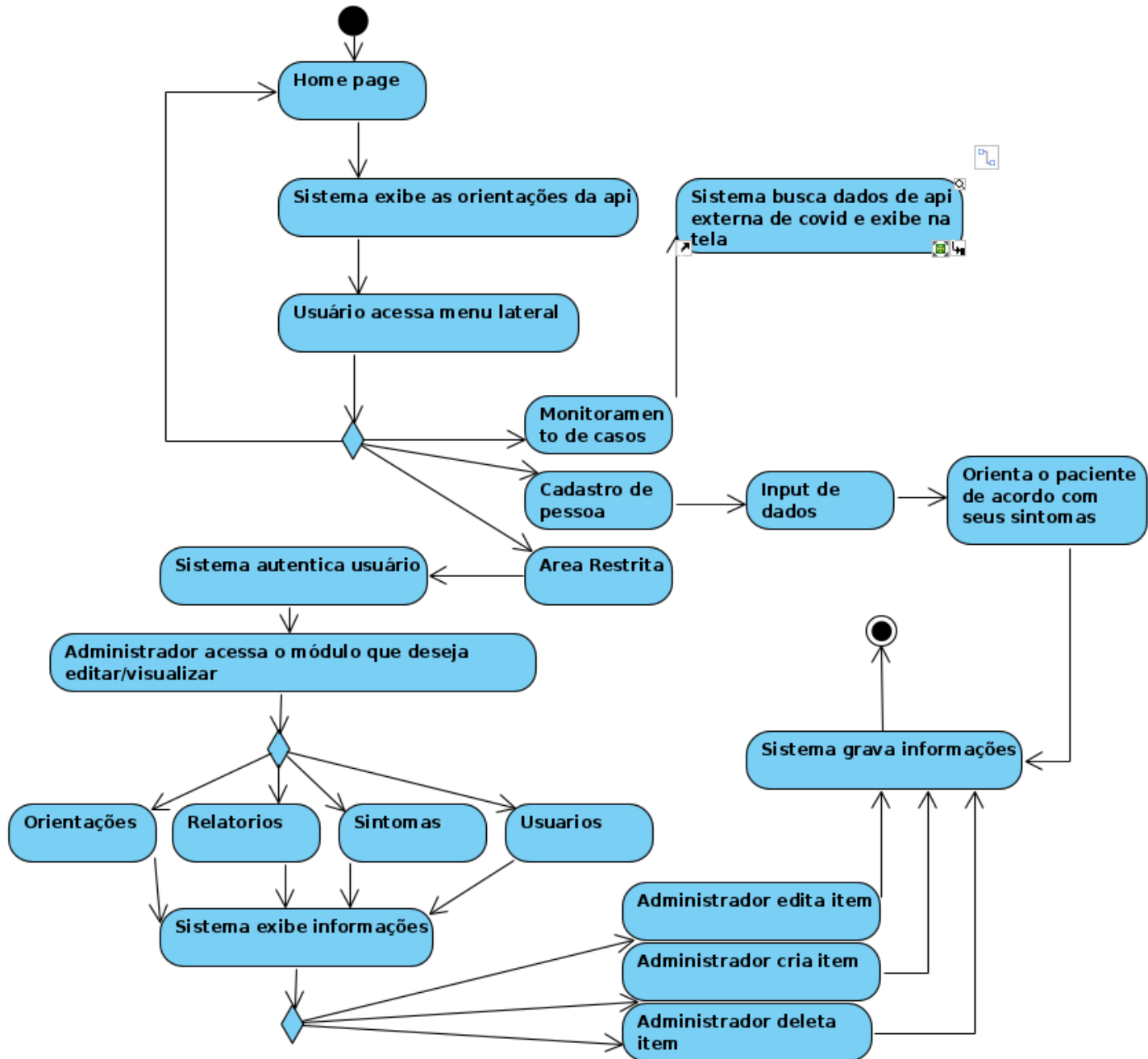
3.3 Casos de uso e fluxos: Monitoramento de dados

Nome	Monitoramento de dados
Atores	Qualquer usuário
Resumo	Consome uma api externa para exibir os dados em tempo real de covid no Brasil e Mundo
Pré-Condições	
Pós-Condições	
Fluxo Principal - Visualizar dados	
<p>Após acessar o site, acessar o submenu Monitoramento de dados</p> <p>O sistema deverá consumir a api e exibir os dados na tela.</p>	

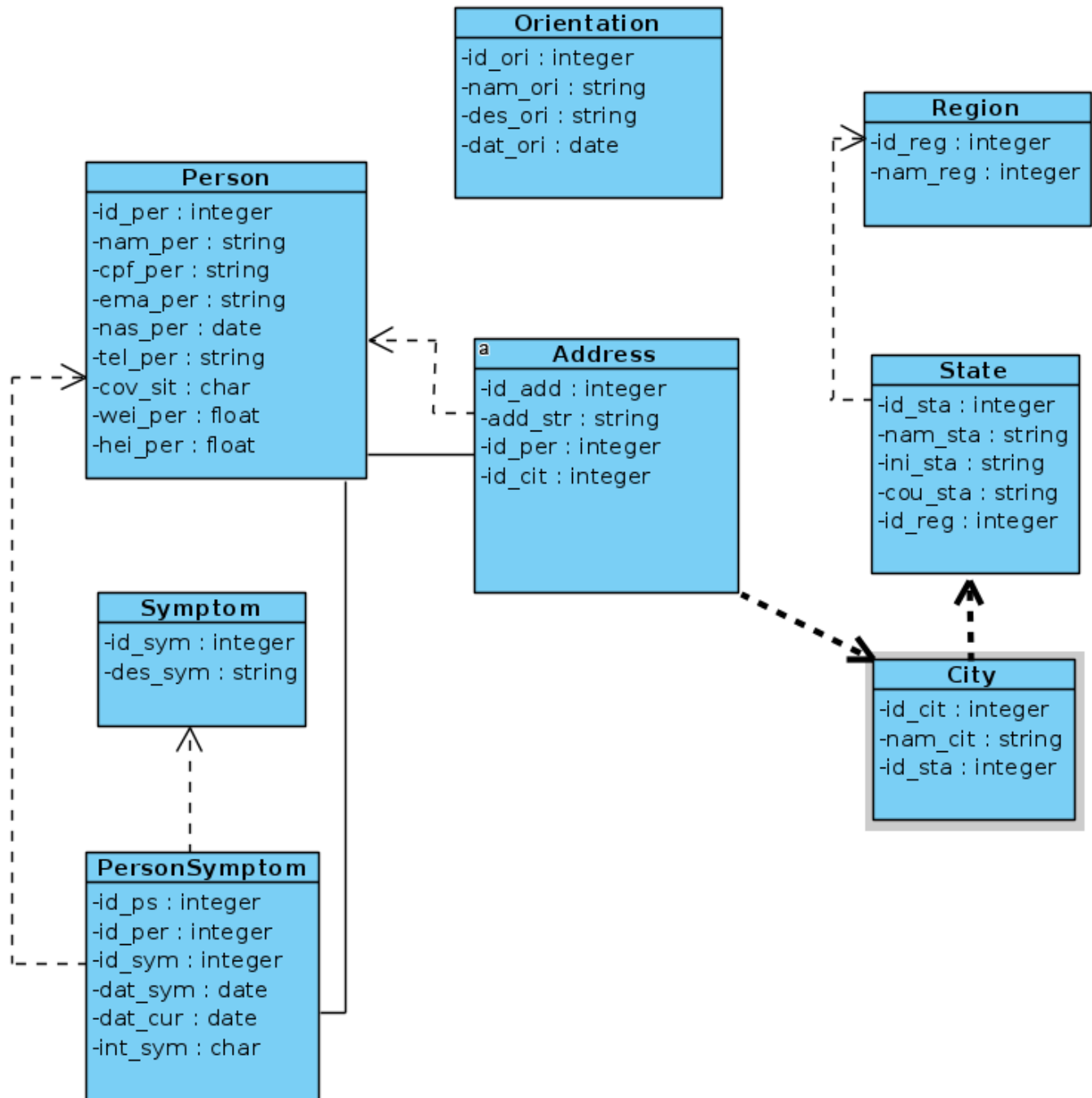
3.4 Casos de uso e fluxos: Diagrama de caso de uso



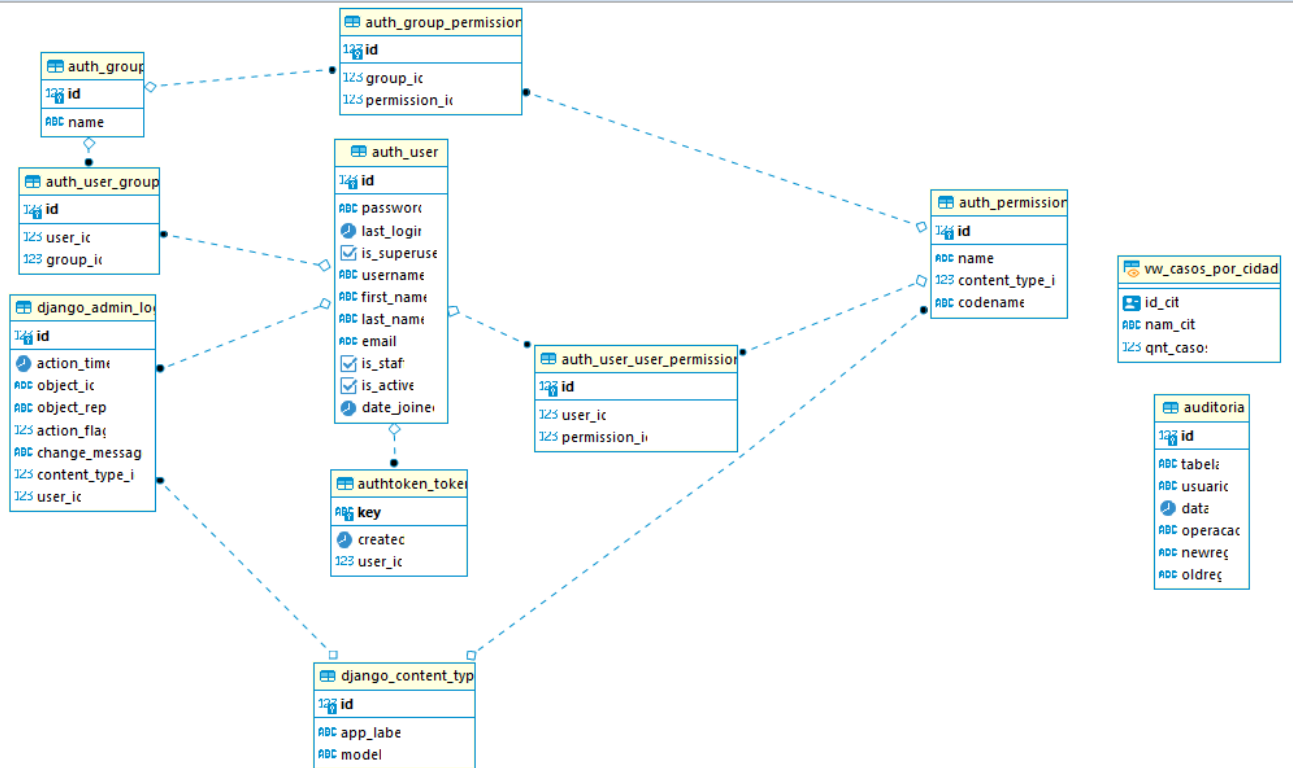
4 Diagrama de Sequência

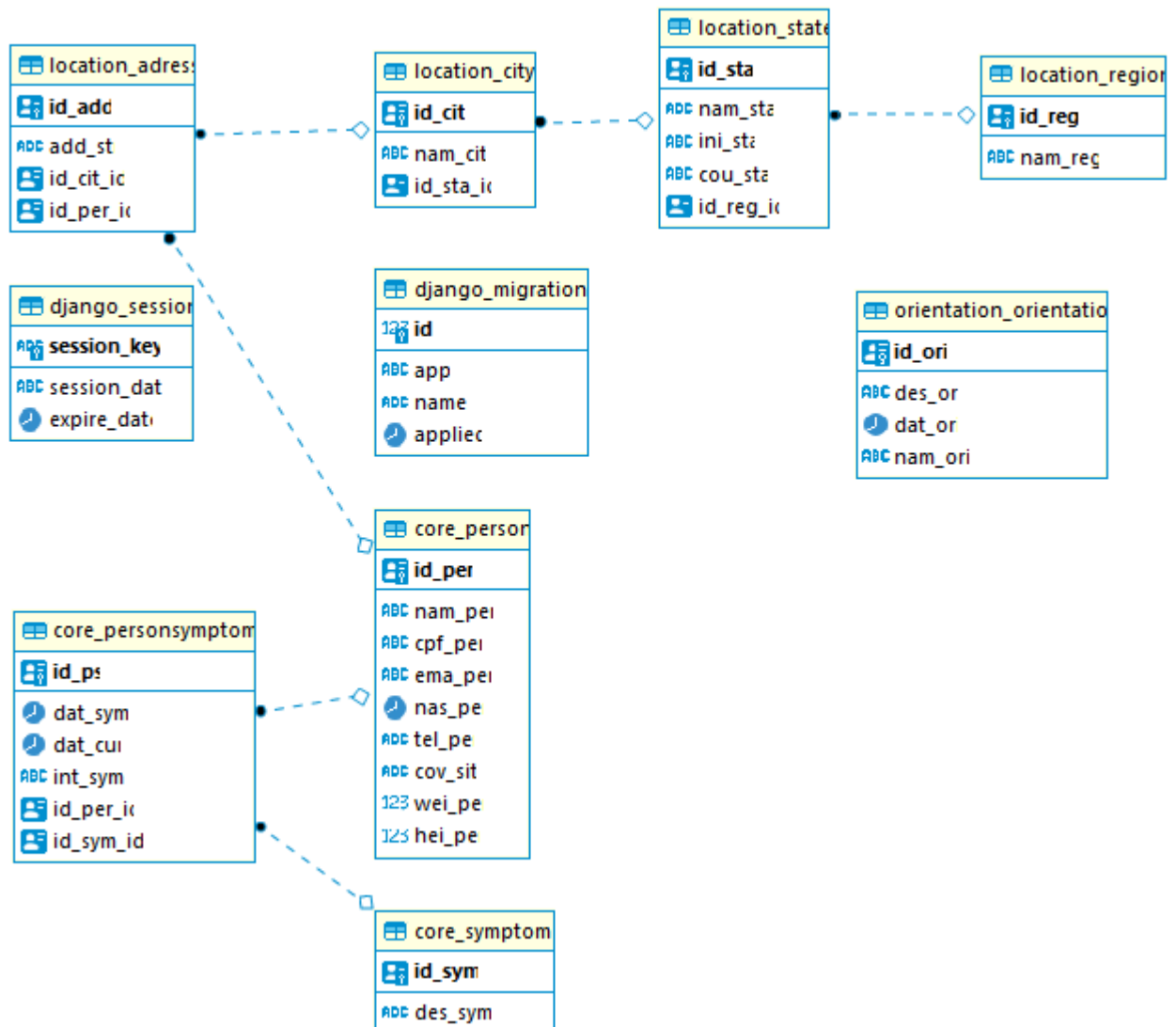


5 Diagrama de Classes



6 Modelo do banco





7 Scripts de Select

1) Relacione o código e nome de pacientes com idades ímpares, que apresentaram febre. Relacione a consulta em ordem ascendente de nome;

```
select p.id_per, p.nam_per, extract (year from AGE(CURRENT_DATE, nas_per)) as idade, s.des_sym as sintoma, ps.int_sym as intensidade
from core_person p
inner join core_personsymptom ps on p.id_per = ps.id_per_id
inner join core_symptom s on ps.id_sym_id = s.id_sym
where mod(extract(year from AGE(CURRENT_DATE, nas_per))::integer, 2) = 1
and s.des_sym = 'Febre'
order by p.nam_per
```

2) Relacione o nome do paciente, nome da cidade de residência de pacientes com mais de 60 KG e residentes nos municípios de Maravilha, Descanso, Pinhalzinho, Chapecó e Itapiranga que apresentaram sintomas e foram positivados com covid. Relacione o relatório pelo nome da cidade ascendente e o nome do paciente descendente;

```
select p.nam_per, c.nam_cit
from core_person p
left join location_address a on a.id_per_id = p.id_per
left join location_city c on c.id_cit = a.id_cit_id
where p.wei_per > 60 and p.cov_sit = 'A' and (select count(*) from core_personsymptom ps where ps.id_per_id = p.id_per) > 0
and upper(c.nam_cit) like any ('{"MARAVILHA", "DESCANSO", "PINHALZINHO", "CHAPECÓ", "ITAPIRANGA"}')
order by c.nam_cit asc, p.nam_per desc
```

3) Relacione o código da cidade, nome da cidade e quantidade de casos suspeitos de covid para cidades com mais de 20 casos. Ordene o relatório da cidade com mais casos suspeitos para a cidade com menos casos suspeitos;

```
with casos_por_cidade as (select c.id_cit, (select count(*) from core_person p
left join location_address a on a.id_per_id = p.id_per
where a.id_cit_id = c.id_cit and p.cov_sit = 'S') as qnt_casos
from location_city c)
select c.id_cit, c.nam_cit, cpd.qnt_casos
from location_city c
left join casos_por_cidade cpd on cpd.id_cit = c.id_cit
where cpd.qnt_casos > 0
order by cpd.qnt_casos desc
```

4) Relacione a idade e quantidade de casos positivos de covid por idade. Somente idades com menos de 10 casos. Ordene o relatório pela idade com mais casos para a idade com menos casos.

```
select age, qnt_casos from
(select extract(year from AGE(p.nas_per)) age, count(*) qnt_casos
from core_person p
where p.cov_sit = 'P'
group by age) foo
where qnt_casos < 10
order by qnt_casos desc
```

8 Configurações de políticas de acesso

```
CREATE ROLE "ADM_CIDADE" NOSUPERUSER NOCREATEDB NOCREATEROLE NOINHERIT LOGIN PASSWORD '12345';
GRANT SELECT ON TABLE public.location_adress TO "ADM_CIDADE";
GRANT SELECT ON TABLE public.location_city TO "ADM_CIDADE";
GRANT SELECT ON TABLE public.location_region TO "ADM_CIDADE";
GRANT SELECT ON TABLE public.location_state TO "ADM_CIDADE";
GRANT INSERT ON TABLE public.location_adress TO "ADM_CIDADE";
GRANT INSERT ON TABLE public.location_city TO "ADM_CIDADE";
GRANT INSERT ON TABLE public.location_region TO "ADM_CIDADE";
GRANT INSERT ON TABLE public.location_state TO "ADM_CIDADE";
GRANT UPDATE ON TABLE public.location_adress TO "ADM_CIDADE";
GRANT UPDATE ON TABLE public.location_city TO "ADM_CIDADE";
GRANT UPDATE ON TABLE public.location_region TO "ADM_CIDADE";
GRANT UPDATE ON TABLE public.location_state TO "ADM_CIDADE";
GRANT ALL ON TABLE public.location_adress TO "ADM_CIDADE";
GRANT ALL ON TABLE public.location_city TO "ADM_CIDADE";
GRANT ALL ON TABLE public.location_region TO "ADM_CIDADE";
GRANT ALL ON TABLE public.location_state TO "ADM_CIDADE";
REVOKE DELETE ON TABLE public.location_adress FROM "ADM_CIDADE";
REVOKE DELETE ON TABLE public.location_city FROM "ADM_CIDADE";
REVOKE DELETE ON TABLE public.location_region FROM "ADM_CIDADE";
REVOKE DELETE ON TABLE public.location_state FROM "ADM_CIDADE";
REVOKE REFERENCES ON TABLE public.location_adress FROM "ADM_CIDADE";
REVOKE REFERENCES ON TABLE public.location_city FROM "ADM_CIDADE";
REVOKE REFERENCES ON TABLE public.location_region FROM "ADM_CIDADE";
REVOKE REFERENCES ON TABLE public.location_state FROM "ADM_CIDADE";
REVOKE TRUNCATE ON TABLE public.location_adress FROM "ADM_CIDADE";
REVOKE TRUNCATE ON TABLE public.location_city FROM "ADM_CIDADE";
REVOKE TRUNCATE ON TABLE public.location_region FROM "ADM_CIDADE";
REVOKE TRUNCATE ON TABLE public.location_state FROM "ADM_CIDADE";
```

9 Views do sistema

```
create or replace view VW_CASOS_POR_CIDADE as
with casos_por_cidade as (select c.id_cit, (select count(*) from core_person p
                                left join location_address a on a.id_per_id = p.id_per
                                where a.id_cit_id = c.id_cit and p.cov_sit = 'S') as qnt_casos
                        from location_city c)
select c.id_cit, c.nam_cit, cpd.qnt_casos
from location_city c
left join casos_por_cidade cpd on cpd.id_cit = c.id_cit
where cpd.qnt_casos > 0
order by cpd.qnt_casos desc
```

10 Auditoria do sistema

```
-- criando a tabela onde serão armazenadas as informações auditadas
CREATE TABLE AUDITORIA (
    ID SERIAL NOT NULL PRIMARY KEY,
    TABELA VARCHAR(50) NOT NULL,
    USUARIO VARCHAR(50) NOT NULL,
    DATA TIMESTAMP NOT NULL,
    OPERACAO VARCHAR(1) NOT NULL, -- I - INCLUSÃO, E - EXCLUSÃO, A - ALTERAÇÃO
    NEWREG TEXT,
    OLDREG TEXT
);
```

```
-- criando a função genérica de auditoria
CREATE OR REPLACE FUNCTION ft_auditoria() RETURNS TRIGGER AS
$body$
BEGIN
-- Cria uma linha na tabela AUDITORIA para refletir a operação
-- realizada na tabela que invoca a trigger. --
IF (TG_OP = 'DELETE') THEN
    INSERT INTO auditoria(tabela, usuario, data, operacao,oldreg) SELECT
TG_RELNAME, user, current_timestamp, 'E', OLD::text;
    RETURN OLD;
ELSIF (TG_OP = 'UPDATE') THEN
    INSERT INTO auditoria(tabela, usuario, data, operacao,newreg,oldreg)
SELECT TG_RELNAME, user, current_timestamp, 'A',NEW::text,OLD::text;
    RETURN NEW;
ELSIF (TG_OP = 'INSERT') THEN
    INSERT INTO auditoria(tabela, usuario, data, operacao,newreg)
SELECT TG_RELNAME, user, current_timestamp, 'I',NEW::text;
    RETURN NEW;
END IF;
RETURN NULL; -- o resultado é ignorado uma vez que este é um gatilho AFTER
END;
$body$
LANGUAGE plpgsql;
```

```
-- auditando a tabela core_personsymptom, E core_person
CREATE TRIGGER core_personsymptom_audit AFTER INSERT OR UPDATE OR DELETE ON core_personsymptom FOR EACH ROW EXECUTE PROCEDURE ft_auditoria();
CREATE TRIGGER core_person_audit AFTER INSERT OR UPDATE OR DELETE ON core_person FOR EACH ROW EXECUTE PROCEDURE ft_auditoria();
-- agora basta executar os comandos insert, update e delete sobre estas tabelas, e as mesmas serão auditadas.
```

11 Conclusão

Neste trabalho objetivou-se o desenvolvimento de um Sistema de Monitoramento de Casos de Coronavírus, desenvolvido na linguagem Python e Flutter, juntamente com um banco de dados relacional e o uso de diagramas de classes, sequência, estado, etc.

Podemos identificar uma grande possibilidade sobre sistemas de monitoramento de doenças, podendo não só fazer o monitoramento, mas um acompanhamento geral da doença juntamente com o paciente, auxiliando com orientações e gerando relatórios completos sobre o estado da doença em si, automatizando totalmente um processo de tentativa de cura de uma determinada doença, de tal forma que não somente o coronavírus, mas sim qualquer outra doença possa vir a ser monitorada por um sistema funcional.