SCC0640 - Base de Dados

Trabalho:
Gerenciador de Pesquisas Clínicas

Docente:
Profa. Elaine Parros Machado de Sousa
Estagiário PAE:
André Moreira Souza

Bruno C. C. Navas - 13783947
Eduardo B. Luceiro - 14607621
Heloísa Pazeti - 14577991
Miguel R. Fonseca - 14682196

Dezembro - 2025

Conteúdo

1 Descrição do Problema e dos Requisitos de Dados

1.1 Ideia Geral e Motivação

O projeto tem como objetivo a criação de um ambiente para que clínicas médicas e agências de fomento possam se conectar para a produção de pesquisas científicas. Esta aplicação irá contemplar toda a comunicação entre agências e clínicas, na qual as primeiras podem manter registros de todas as pesquisas que financiam, com seus participantes, relatórios, descrições e resultados, e as segundas podem manter um registro do seu financiamento, aceitação acadêmica dos resultados ou pedidos de recursos extras. Além disso, os pesquisadores relacionados às pesquisas terão acesso à plataforma para que possam analisar o andamento de seus trabalhos, organizar seus documentos e se comunicar com clínicas e agências de fomento vinculadas.

A motivação dos desenvolvedores advém da necessidade de uma aplicação séria para essa comunicação, que requer muitas prestações de contas e segurança para manter os relatórios e resultados de forma íntegra. Diante disso, os alunos escolheram esse projeto por haver diversos detalhes importantes para uma base de dados desse gênero, além de ser uma realidade próxima à dos integrantes.

1.2 Principais Operações e Funcionalidades

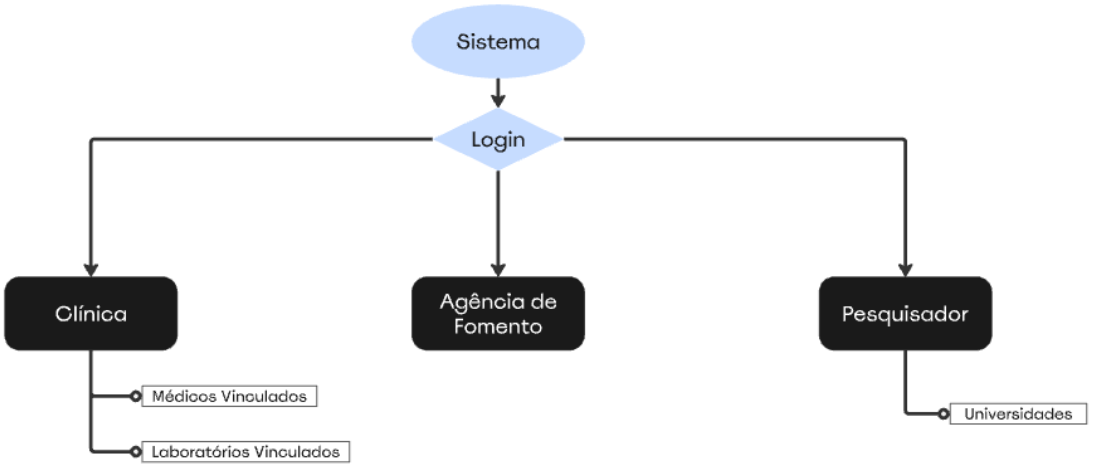

Figura 1: Diagrama do Sistema Geral. Fonte: autores.

Como pode ser notado pelo esquema da figura (1), o sistema geral irá consistir em um login ou cadastro de três tipos de usuário diferentes: Clínica, Agência de Fomento ou Pesquisador. Além dessas entidades que podem ser logadas ao sistema, elas terão outros institutos e organizações associadas, mas que não terão login no sistema, apenas terão seus dados cadastrados para a realização de algumas operações que serão explicadas adiante.

Para facilitar a compreensão das operações e funcionalidades idealizadas, é necessário ter em mente os três principais "usuários"que utilizarão a aplicação, isto é, Clínica, Agência de Fomento e Pesquisador. Assim teremos três lados da aplicação com funções diferentes.

1.2.1 Lado Clínica

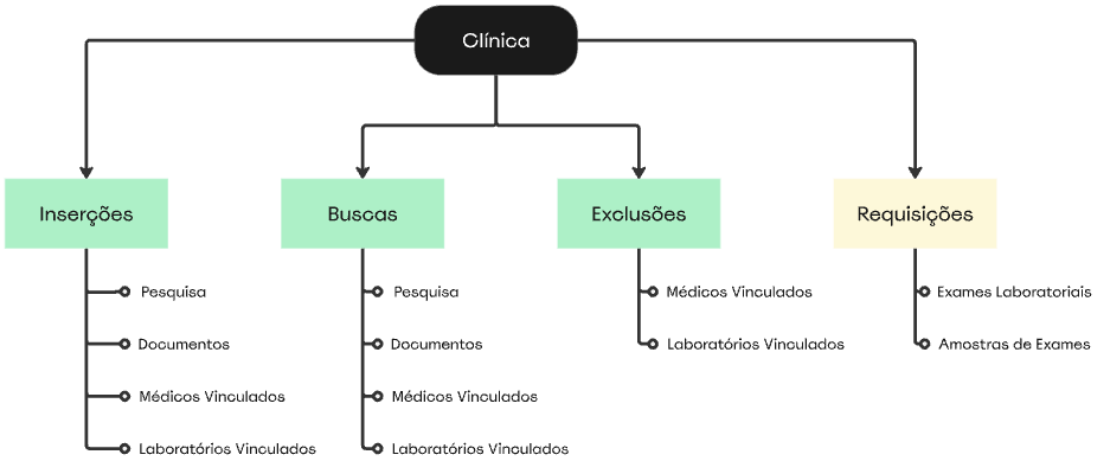

Figura 2: Diagrama Clínicas. Fonte: autores.

Para a clínica existirão as funções básicas de inserir, buscar e remover os dados apresentados no diagrama (2).

- Inserção
  - Pesquisas - novas pesquisas que estarão relacionadas à clínica em questão, de forma a poder manter o controle e poder analisar os resultados posteriormente.

  - Documentos - novos documentos, desde os relacionados as pesquisas em si (relatórios, descrições, projetos), até outros documentos requisitados pelos pesquisadores (exames, resultados de pesquisas anteriores ou estatísticas).
  - Médicos Vinculados - médicos que não constam como pesquisadores não terão acesso ao sistema, porém, eles podem estar ligados as pesquisas fazendo exames ou análises especializadas, por isso devem constar no sistema, ligados à clínica em que trabalham.
  - Laboratórios Vinculados - assim como para os médicos, como pode ser necessário a realização de exames ou coleta de dados específicos, é preciso que os laboratórios que realizem tal estejam conectados à clínica para que suas informações possam ser encontradas.

- Busca - a clínica pode fazer buscas sobre esses tópicos mostrados na figura (2), isto é, pode pesquisar um médico pelo seu ID único e descobrir se a clínica está vinculada a esse médico e em quais pesquisas ele está atuando - caso esteja em alguma -, por sua vez.

- Exclusões - permite que a clínica remova algum médico ou laboratório vinculado, caso não estejam mais trabalhando em conjunto.

Além disso, é importante ressaltar que as "Requisições", como intitulado na figura (2), não são ações necessariamente configuradas no sistema, mas que suas informações são necessárias. Por exemplo, como citado na inserção de documentos, pode ser necessário para uma pesquisa resultados de exames de pacientes ou mesmo amostras. A mediação entre os laboratórios e pesquisadores para que tenham acesso a esses exames e amostras será feita pelas clínicas vinculadas às pesquisas. Dentro dessa relação, é importante explicitar a necessidade de uma requisição e de uma devolução e atributos relacionados a isso.

1.2.2 Lado Agência de Fomento

Para as agências de fomento temos ações parecidas com o que foi apresentado na seção (1.2.1), porém algumas considerações devem ser feitas.

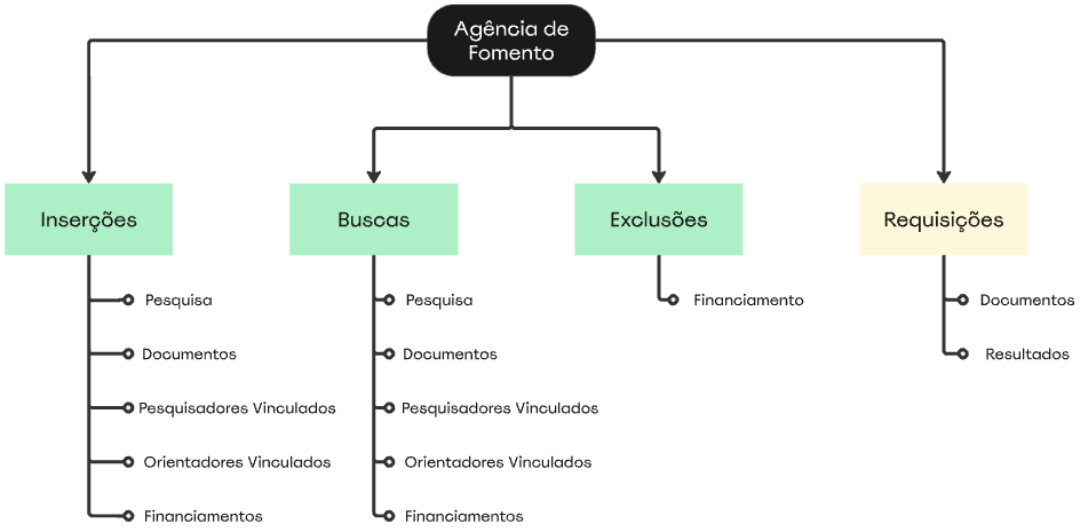

Figura 3: Diagrama Agência de Fomento. Fonte: autores.

- Inserções - a agência de fomento não cadastra novos pesquisadores ou orientadores, mas apenas os vinculados às pesquisas que a agência, por sua vez, financia. Não haverá duplicações de cadastros ou dados, e isso será mais aprofundado durante a implementação do sistema.

- Buscas - a agência pode buscar informações de acordo com as informações apresentadas na figura (3).

- Exclusões - a agência de fomento pode remover um financiamento, isto é, desistir de uma pesquisa por quaisquer motivos, como resultados não apresentados ou requisitos não cumpridos, dentre outros.

Além disso, nota-se que "Requisições"se refere ao fato de que a agência de fomento pode requerer documentos não apresentados, sejam eles dos pesquisadores ou das pesquisas.

1.2.3 Lado Pesquisador

Para o pesquisador, temos situação semelhante às demais, tendo as principais ações de uma base de dados como:

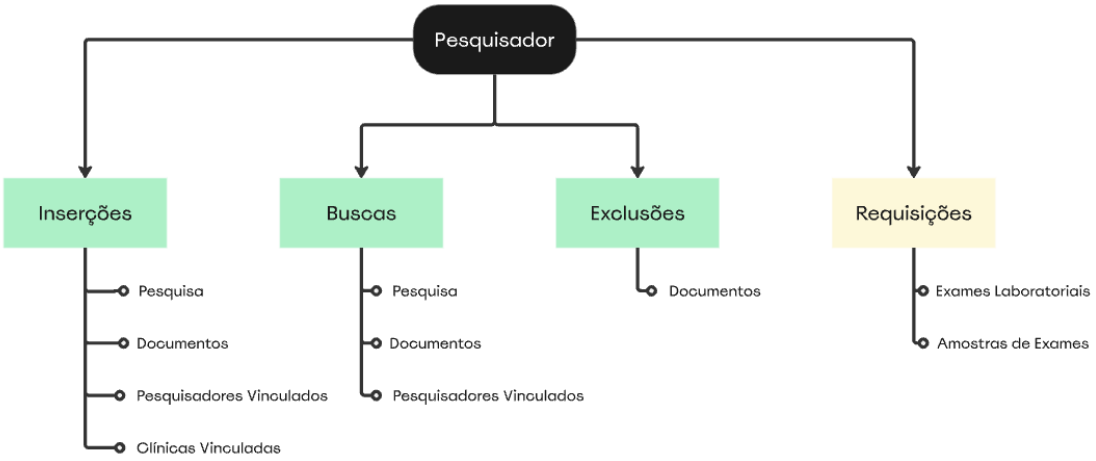

Figura 4: Diagrama Pesquisador. Fonte: autores.

- Inserções - anexar documentos como relatórios, resultados, etc. Deixar vinculados outros pesquisadores, que podem ser colegas ou orientadores, além de saber a quais clínicas estão vinculadas suas pesquisas, sendo a essas para as quais irão exigir exames e amostras dos laboratórios que estão aliados as clínicas, como explicado em (1.2.1).

- Buscas - buscas gerais sobre pesquisas das quais faz parte.

- Exclusões - pode excluir documentos (não após a pesquisa ser finalizada), caso estejam incorretos.

Como explicitado anteriormente, os pesquisadores, em específico os que serão orientadores - essa separação será mais explorada futuramente - podem requerer resultados ou amostras de exames às respectivas clínicas a que estão vinculados.

1.3 Descrição do Problema e do Requisito de Dados

De forma mais detalhada, abaixo serão listados todos os conceitos utilizados para descrever a aplicação, suas entidades, atributos e relações.

Nesse sistema teremos a relação entre uma clínica e uma agência de fomento para a criação e divulgação de pesquisas médicas de remédios e tratamentos. Dentro do sistema, serão registrados dois tipos de estabelecimentos de saúde com diferentes

funções: hospitais / clínicas e laboratórios de análise. Para esses, é importante que o sistema salve informações como CNES, para identificá-las, endereço (UF, cidade, bairro, rua, número), telefones para contato.

Para cada uma das pessoas envolvidas nesse projeto, é importante salvar informações relevantes como CPF, para identificá-las, nome, função dentro do sistema, endereço (UF, cidade, bairro, rua, número), telefones para contato, data de nascimento e idade.

Para atuar na pesquisa, haverá um pesquisador, que pode ser um pesquisador comum ou um pesquisador responsável, sendo que esse último pode solicitar dados ou amostras de um hospital ou clínica. Dessa solicitação é importante guardar a situação da solicitação (aceita, em andamento ou negada) e uma pequena descrição do por quê da solicitação e para que será usada.

Além disso, é importante ter o registros dos médicos que auxiliam na pesquisa. Deles é importante saber seu CRM, hospital ou clínica no qual estão vinculados, se estiverem em alguma.

Com relação a execução da pesquisa é importante saber se estão vinculadas a uma universidade e, no caso de estarem, é preciso saber dessas universidades seu CNPJ para identificá-las, nome, campus e endereço. É preciso saber quem é o pesquisador relacionado a essa pesquisa, sua data de início e de conclusão, caso ainda não tenha sido concluída, salva-se a data esperada. Também precisa-se ter a informação acerca da agência que financia essa pesquisa. Sobre a pesquisa também quer-se saber seu título, área e descrição. Quanto a agência de fomento espera-se os dados de CNPJ, nome, abrangência e seu tipo (pública ou privada).

Durante a execução da pesquisa pode ser necessário que sejam analisados dados de amostras ou exames. Quanto aos dados de amostras, é necessário saber a data de coleta, os dados obtidos, e sua identificação. Em caso de uma amostra ser disponibilizada, é importante saber a data em que foi disponibilizada e sua previsão de devolução. Também é importante saber a que estabelecimento de saúde ela está vinculada. Da amostra em si, são relevantes os dados sobre sua validade, estado, quantidade e outras informações relevantes. Além disso, é importante saber qual o tipo de coleta, sua categoria (sangue, urina, pele, etc...) e a periculosidade desse

tipo de coleta (invasiva, não invasiva, etc...). É importante também que tenha-se os dados do enfermeiro que performou a coleta de amotra, como seu COREN, e que seja certificado que ele não fez a coleta nele próprio.

Com relação aos exames que podem ser requisitados, é importantes saber o médico que performou a coleta, é importante certificar que ele não tenham realizado o exame nele mesmo. Sobre o exame são relevantes as informações sobre a data do exame, seus resultados, tipo de exame (categoria e periculosidade), se foi disponibilizado ou revogado e a data dessa resposta. No caso de o exame ser disponibilizado é importante também a data de prevista para a devolução.

Do paciente queremos saber quais as doenças, porém para a pesquisa efetivamente é importante que seus dados pessoais e / ou sensíveis não sejam revelados.

2 Modelo Entidade Relacionamento - MER

2.1 Análise de Ciclos

Alguns ciclos podem ser analisados dentro do MER, eles decorrem da sobreposição das especializações de pessoa, isto é, como o pesquisador pode ser médico, o médico ser paciente, e assim por diante, podemos ter alguns ciclos.

Ciclo Médico Pesquisador e Execução: ciclo de redundância que deverá ser tratado na aplicação.

Ciclo Pesquisador Enfermeiro e Execução: ciclo de redundância que deverá ser tratado na aplicação.

Ciclo Paciente Enfermeiro e Execução: como foi constatado na requisição de dados, um enfermeiro não pode retirar a própria amostra, dessa forma, esse ciclo não ocorre na prática.

Ciclo Médico Paciente e Exame: esse caso é um ciclo de dependência, pois há uma diferença semânticas nas relações das entidades, além de que, é especificado na requisições de dados que o médico não pode realizar o próprio exame, de forma que esse caso não ocorrerá na prática.

Ciclo Paciente Pesquisador e Execução: apesar de ser um ciclo de dependência, futuramente esse caso será tratado após a análise LGPD, pois os dados do paciente não estão vinculados á pesquisa, de forma que os dados não serão duplicados.

2.2 Duplicações

Como é possível observar pelo MER (5), atributos não foram duplicados, como referenciado em (1.2.2). Haverá um cuidado para que não haja duplicações nos registros e serão utilizadas tabelas e chaves estrangeiras para que a integridade do sistema seja mantida.

2.3 Restrições de Integridade

- Possíveis inconsistências entre vinculações de pesquisadores e clínicas - é necessário verificar na aplicação se existe de fato a clínica ou pesquisa vinculado na base de dados.

- Possíveis inconsistências entre vinculações de clínicas e médicos - novamente é necessário certificar na aplicação de que clínica e médicos estão registrados na base de dados.

- Possíveis inconsistências entre vinculações de clínicas e laboratórios - é necessário que os exames ou amostras que serão requeridos pelas clínicas provenham de um laboratório existente, essa informação não pode ser nula.

- Possíveis inconsistências entre as entidades e as informações das pesquisas - para adicionar informações a alguma pesquisa, é necessário checar na aplicação se ela é uma pesquisa iniciada com todos os dados necessários e que não haja inconsistência entre uma mesma pesquisa e as demais entidades que estão vinculadas a ela.

2.4 Leis de Proteção de Dados

Como resultados de exames e análises médicas são dados muito sensíveis para os pacientes, futuramente será evidenciado como será estruturado o sistema para que os resultados e informações dos pacientes não sejam expostos de nenhuma forma.

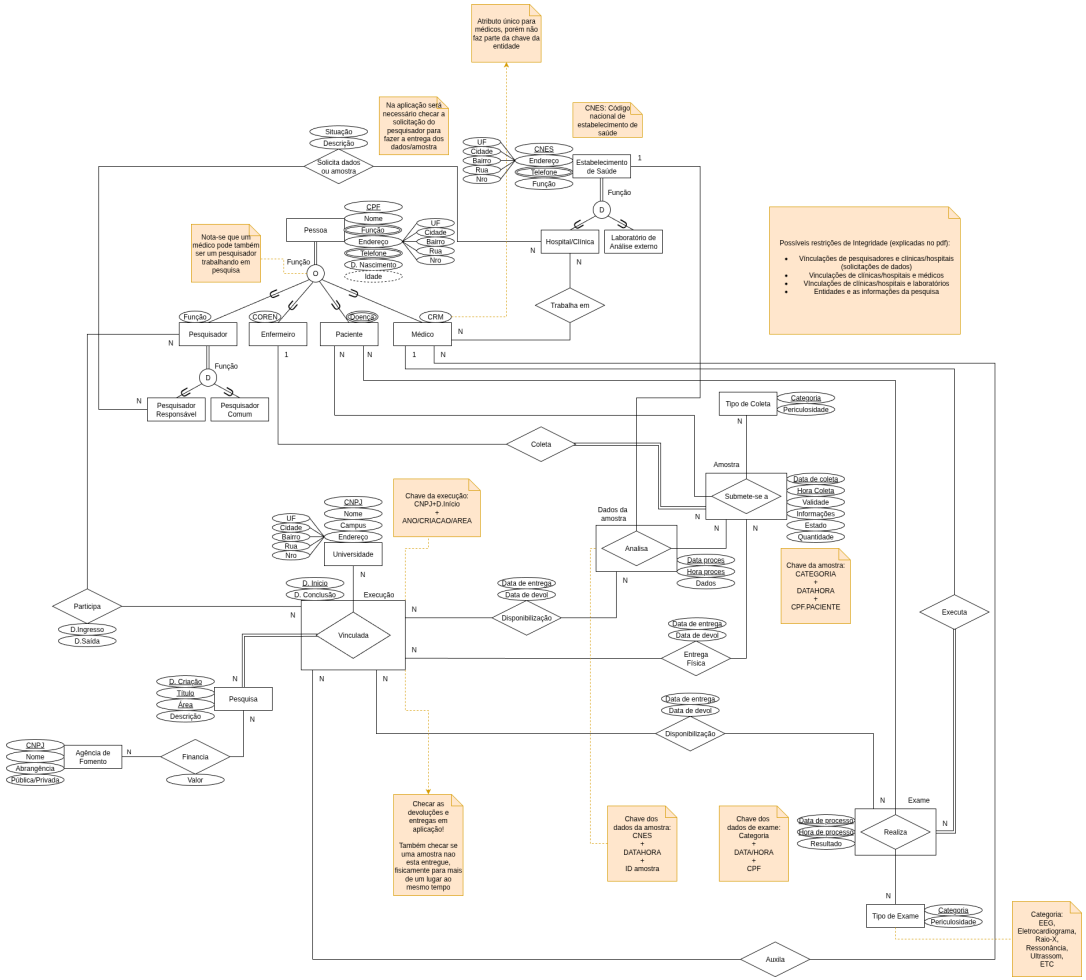

Figura 5: MER. Fonte: autores.

3 Alterações no Projeto

3.1 Alterações no MER

- Adição do atributo de especialização na especialização de Pessoa, na de Pesquisador e na de Estabelecimento de Saúde.
- Alteração de algumas cardinalidades.
- Adição de notas para atributos identificadores que não são chaves primárias.
- Alteração de chaves primárias anteriormente consideradas como números gerados automaticamente e passando isso para atributos presentes nas entidades.
- Alteração dos relacionamentos entre Execução e Amostras, Dados de Amostra e Exame.

3.2 Alterações no Relatório

- Adição da seção 1.3 - Descrição do Problema e Do Requisito de Dados -, na qual é descrita um pouco sobre os dados que serão armazenados e como eles devem ser utilizados no banco de dados.
- Adição da seção 2.1 - Análise de Ciclos -, na qual analisamos a natureza dos ciclos presentes em nosso modelo entidade relacionamento e como tratá-los se necessário.

4 Modelo Relacional

4.1 Modelo

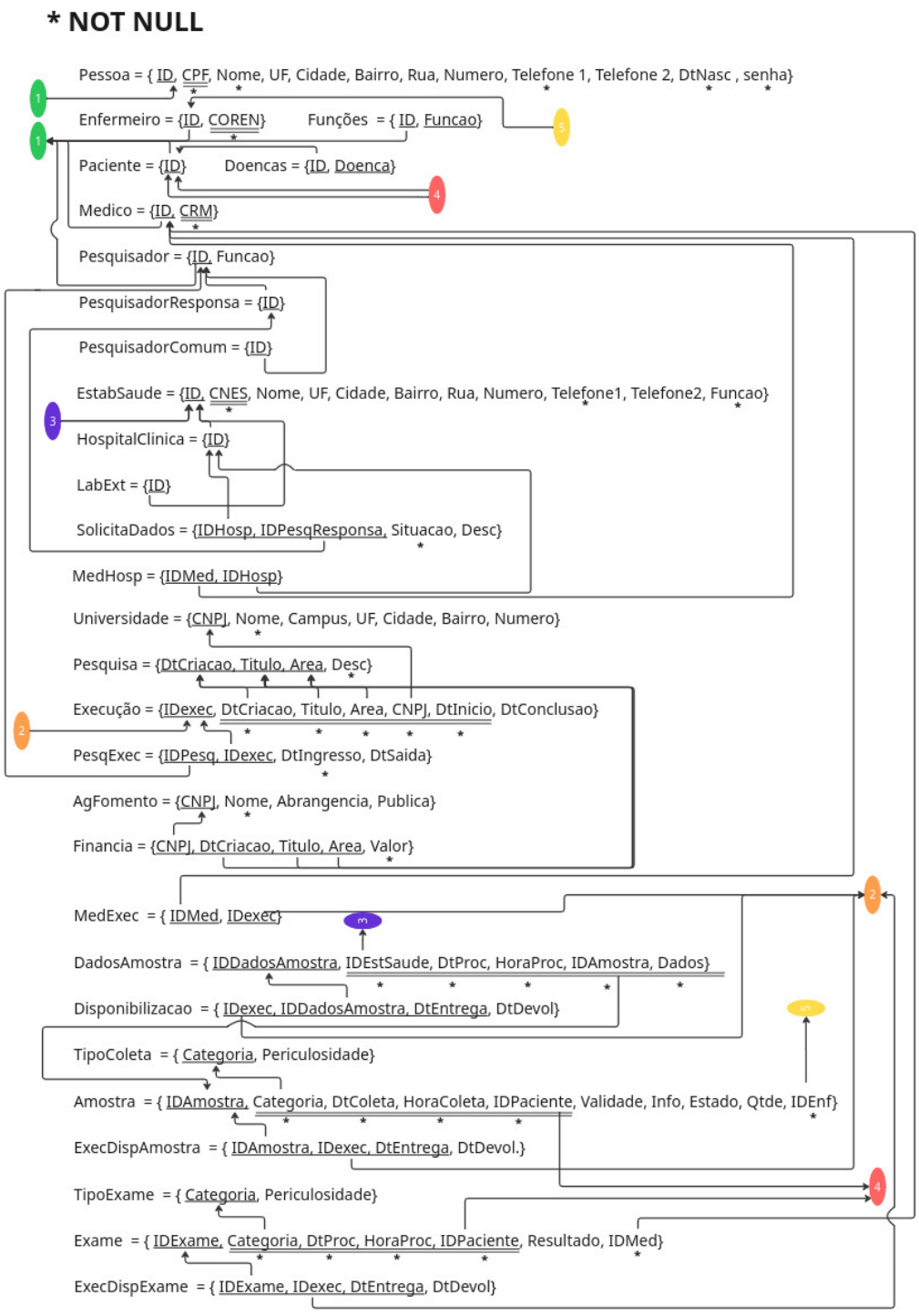

Figura 6: Modelo Relacional. Fonte: autores.

4.2 Justificativas

Segue abaixo algumas justificativas das escolhas de modelagem.

4.2.1 Campo Multi-valorado: Telefone

Em todos os casos onde havia o atributo telefone multi-valorado foram adicionados ao modelo apenas 2 campos para o telefone, dado que, para nossa aplicação, acreditamos que isso seja o suficiente.

Vantagens: menos espaço de armazenamento, sem necessidade de fazer um join nas tabelas.

Desvantagens: o número de telefones fica restrito a apenas dois, caso fossem necessários mais, precisaria alterar a estrutura da tabela.

Mapeamento Alternativo: poderia ser criado uma nova tabela 'Tefelone', semelhante a tabela apresentada na seção (4.2.2). Essa tabela seria criado como Telefone = {ID, telefone}, onde 'ID' seria chave estrangeira liga a 'ID' da tabela 'Paciente'.

4.2.2 Campo Multi-valorado: Paciente - Doenças

Como pode ocorrer de um paciente ter diversas doenças, foi considerada a melhor modelagem a criação de uma nova tabela de nome 'Doencas', na qual pode relacionar um paciente a diversas doenças.

Vantagens: possibilidade de adicionar diversas doenças, sem alterar o formato da tabela ou sem deixar um campo de texto com tamanho indefinido, que poderia gerar confusões.

Desvantagens: Adição de uma nova tabela, maior gasto de espaço de armazenamento e necessidade de fazer um join entre tabelas.

Mapeamento Alternativo: semelhantemente ao demonstrado na seção (4.2.1), poderíamos mapear isso de forma a limitar a quantidade de doenças deixando a tabela 'Paciente' no formato Paciente = {ID, doenca1, doenca2, ..., doencaN}, onde N é o número máximo do campo. Essa maneira não foi utilizada por acreditar-se que isso limitaria muito o sistema.

4.2.3 Campo Derivado Idade - entidade Pessoa

Nesse caso não salvamos o dado idade, já que podemos facilmente obtê-lo utilizando a data de nascimento. Apesar de gastar processamento com esse cálculo, acredita-se que não será uma ação recorrente e é de custo leve.

Vantagens: um campo a menos na tabela, menos possibilidade de inconsistências.

Desvantagens: toda vez que for necessário saber a idade será necessário performar um cálculo, gastando processamento.

Mapeamento Alternativo: poderíamos guardar a informação na tabela 'Pessoa'.

4.2.4 Normalização do campo Endereço

Em algumas entidades foi adicionado o campo endereço de forma normalizada, seguindo da 1° Forna Normal, isto é, abrimos esse campo em campos mais específicos como: UF, Cidade, Bairro, Rua, Numero.

Vantagens: isso facilita para recuperar partes do endereço da entidade e serão campos menores, ao em vez um único grande campo de texto.

Desvantagens: existem mais campos para serem inseridos.

Mapeamento Alternativo: seria possível ter deixado o campo apenas como 'endereco' de forma a manter apenas uma longa string com essa informação.

4.2.5 ID artificial - Chave Primária Composta

Alguns campos - em especial as agregações - possuíam como chaves primárias campos compostos, o que tornava a referenciação a essas tabelas complexa. Portanto, acredita-se que trocar essas chaves por IDs artificiais seja uma boa opção. Os campos que anteriormente compunham a chave primária passaram a compor uma chave secundária, todos colocados como not null, de forma a certificar de que esses campos - que contém informações relevantes - não serão vazios.

Vantagens: diminui o tamanho da tabela, de forma a poder guardar mais registros dentro de uma página de arquivo, o que por sua vez diminui a altura da

árvore binária criada para a tabela.

Desvantagens: informações anteriormente presentes na tela como chaves estrangeiras, são substituídas pelos ids artificiais, de forma que, caso queiramos reaver essas informações são necessários mais joins do que antes.

4.2.6 ID artificial - Lei Geral de Proteção de Dados

Devido à LGPD, é necessário que nenhuma pessoa possa ser ligada ao seu CPF, isto é, seu CPF não deve ser vinculado pelo banco de dados publicamente. Para resolver esse problema, criamos um ID artificial para ser a chave primária da entidade Pessoa, e passamos essa chave adiante na especialização. Dessa forma, as demais entidades como Pesquisa, Execução, Estabelecimento de Saúde, etc. não poderão obter o dado de CPF das pessoas.

Além disso, também foi criado um ID para os estabelecimentos de saúde, para que seu CNES não seja obtido por terceiros, de forma a comprometer a segurança dos espaços de saúde.

Vantagens: mantemos em anonimato os participantes da pesquisa, como previsto por lei.

Mapeamento Alternativo: seria possível utilizar o Particionamento Vertical, na qual utilizaríamos duas tabelas ou mais que compartilhariam a mesma chave primária, como por exemplo:

- PACIENTE_IDENTIFICACAO = {ID_Paciente [PK], ID_Artificial, Nome, CPF,...}
- PACIENTE_INFO = {ID_Paciente [PK, FK], Historico_Clinico, Alergias,...}

Dessa forma, as informações do paciente não ficariam diretamente ligadas ao seu CPF.

4.2.7 Especialização Pessoa

Para essa especialização foi escolhida a modelagem na qual mantivemos uma tabela para a entidade principal, com as chaves e seus atributos, uma tabela separada para separar as funções e os CPFs, mantendo ambos como chaves primárias para permitir

o Overlap das entidades filhas. Além disso, existe uma tabela para cada uma das funções com seus respectivos atributos

Vantagens: garantimos que o Overlap será mantido e podemos fazer buscas rapidamente para saber ligar o CPF à função no projeto em questão. Também escolhemos essa forma devido à estrutura da especialização em que existem diversos atributos na entidade principal, e poucos atributos específicos, além de termos diversos relacionamentos específicos.

Desvantagens: não podemos garantir a especialização total no modelo, apenas em aplicação.

Mapeamento Alternativo: uma outra maneira de mapear seria não utilizando a tabela 'Funcoes', e mantendo as outras iguais, isso ocuparia menos espaço, porém seria mais custoso fazer as buscas, caso não se soubesse a função da pessoa buscada.

4.2.8 Especialização Pesquisador

Aqui foi escolhido outro tipo de especialização, devido a ser de disjunção e a natureza de ter poucos atributos específicos ou gerais, existindo apenas devido ao relacionamento específico de Pesquisador Responsável com Hospital / Clínica, foi utilizado um mapeamento que garante a disjunção, mas não a especialização total, porém não adiciona tabelas ou cópia de campos.

Vantagens: busca rápida, sem repetição de informação.

Desvantagens: não garante especialização total, nem disjunção.

Mapeamento Alternativo: uma outra forma de mapear seria remover a tabela 'Pesquisador' e manter apenas as tabelas 'PesquisadorResponsa' e 'PesquisadorComum', porém, obviamente, isso tornaria mais difícil a consulta caso não se soubesse em qual das tabelas a pessoa procurada poderia estar.

4.2.9 Especialização Estabelecimento de Saúde

Pelos mesmo motivos apresentados em 4.2.8 - Especialização Pesquisador - decidimos fazer um mapeamento semelhante.

4.2.10 Relação N:N - Sem Participação total

Devido à complexidade de relações N:N, resolvemos mapear essa relações, como pode ser observado na relação Pesquisador Responsável e Hospital / Clínica, criando uma terceira tabela para a relação, dessa forma garantindo o N:N, porém perdendo espaço de armazenamento com uma nova tabela.

Vantagens: conseguimos manter a relação N:N requerida pelo MER.

4.2.11 Relação 1:N com participação total entre Enfermeiro e Amostra

Nessa relação mapeamos de forma a certificar a participação total de amostra em relação ao enfermeiro, utilizando um not null no campo de enfermeiro.

4.2.12 Agregações - Chaves Compostas Entre Entidades

Para as agregações presentes no MER modelamos criando uma terceira tabela para relacionar as duas entidades que participam da agregação. Nessa terceira tabela colocamos como chave primária composta, as três chaves indicadas no MER, das entidades que participam e a chave que caracteriza o relacionamento entre elas, dessa forma garantimos o relacionamento N:N e também a possibilidade de múltiplas ocorrências do mesmo par - por isso a agregação.

Mapeamento Alternativo: poderíamos ter mapeado também as relações dentro das agregações, porém isso não se fez necessário, pois não tínhamos atributos dessas relações específicas.

4.2.13 Outras Observações

- Relações N:N com participação total em um lado ou em ambos deve ser garantido em aplicação a participação total.
- Para facilitar compreensão do modelo relacional foram adicionados "conexões"e notas coloridas de forma a indicar a quem ou a o que se referem.

5 Alterações para a 3° parte do Projeto

5.1 Alterações no MER

- Transformação dos atributos de endereço em atributos composto.
- Retirada da participação total de pesquisador em execução. Após análise, o grupo notou que, pensando em sistemas semelhantes de cadastro de pesquisadores, é possível que um pesquisador - comum ou responsável - possa se cadastrar sem nenhuma pesquisa.

5.2 Alterações no Modelo Relacional

- Separação das linhas de chaves estrangeiras no modelo relacional para facilitar a visualização.
- Adição de NOT NULL para o campo 'Funcao', na tabela 'EstabSaude'.
- Adição de NOT NULL no campo 'COREN' na tabela 'Enfermeiro' e no campo 'CRM' na tabelam 'Medico'.
- Adição da chave estrangeira 'IDexec' na tabela 'ExecDispExame'.
- Correção de mau uso de chave estrangeira, trocando pelo ID (chave primária) da tabela amostra.
- Adição do campo 'senha' na tabela Pessoa.

5.3 Alterações no Relatório

- Adição de alternativas de mapeamentos.
- Alteração de algumas vantagens e desvantagens.

6 Descrição da Aplicação

6.1 Tecnologias Utilizadas

- **SGBD:** utilizamos o postgreSQL como nosso SGBD por ser mais simples utilizá-lo de forma local e para adaptá-lo ao escopo do projeto.
- **SO para Gerenciamento de Base de Dados:** utilizamos o Dbeaver para poder escrever códigos SQLs mais facilmente.
- **Linguagem de Programação:** utilizamos python por ser uma linguagem mais simples e com boa documentação.

6.2 Bibliotecas Auxiliares

- psycopg2
- dotenv

6.3 Funcionalidades

6.3.1 Cadastro e Conectar

As duas primeiras funcionalidades iniciais são de cadastro e login. Logo na tela inicial pode-se escolher uma dessas opções. Caso escolha-se a opção de cadastro serão mostrados os campos necessários para preencher a tabela 'Pessoa' mostrado no modelo relacional (4.1). Após o preenchimento desses inputs, ocorre uma operação de 'INSERT' no banco de dados.

```
1
2     connection, cursor = con.GetConnectionAndCursor()
3     sql = """
4         INSERT INTO PESSOA (CPF, NOME, UF, CIDADE, BAIRRO, RUA,
    NUMERO, TELEFONE1, TELEFONE2, DATA_NASC, SENHA)
5         VALUES (
6             %(cpf_bind)s, %(nome_bind)s, %(uf_bind)s, %(cidade_bind
    )s, %(bairro_bind)s, %(rua_bind)s, %(numero_bind)s,
```

```
            %(telefone1_bind)s, %(telefone2_bind)s, TO_DATE(%(
    data_nasc_bind)s, 'DD/MM/YYYY'), %(senha_bind)s)
        RETURNING ID;
    """
    cursor.execute(sql, parametros)
    person_id = cursor.fetchone()[0]
    print("Cadastro de Pessoa realizado com sucesso!")
```

Além disso, são inseridos ainda, de forma semelhante, as informações necessárias nas tabelas: 'Funcoes', 'Enfermeiro', 'Medico', 'Pesquisador', 'Pesquisador Responsavel', 'Pesquisador Comum', 'Paciente'.

Para a tela de login é necessário passar o ID para que se procure a senha relacionada ao cadastro no banco, nesse caso utilizamos uma operação de 'SELECT' para relacionar ID com senha.

```
    sql_query = """
            SELECT SENHA
            FROM PESSOA
            WHERE ID = %(id)s;
        """

    cursor.execute(sql_query, {"id": id})
    resultado = cursor.fetchone()
    cursor.close()
```

6.3.2 Menu Inicial - Selecionar e Mostrar Detalhes

A primeira opção dentro do menu inicial é selecionar todas as pesquisas cadastradas dentro do banco que são ligadas a pessoa logada. Todas essas pesquisas são listadas e pode-se selecionar uma delas para ver mais detalhes.

Para fazer a seleção de todos os dados revelantes da pesquisa e que queremos mostrar para o usuário, temos o seguinte SQL:

```
        connection, cursor = con.GetConnectionAndCursor()
        cursor.execute("""
            SELECT PE.data_ingresso, PE.data_saida, E.TITULO, E.
    AREA, E.DATA_CRIACAO, P.descricao, U.NOME AS Universidade, A.
    NOME AS AgenciaFomento, F.VALOR AS ValorFinanciado
```

```
            FROM Pesq_Exec PE
            JOIN Execucao E
                ON pe.ID_Exec = e.ID
            JOIN Universidade u
                ON e.CNPJ = u.CNPJ
            JOIN pesquisa p
              ON E.titulo = P.titulo
              AND E.area  = P.area
              AND E.data_criacao = P.data_criacao
            LEFT JOIN Financia f
                ON E.titulo     = f.Titulo
                AND E.AREA      = f.Area
                AND E.data_criacao = f.data_criacao
            LEFT JOIN Ag_Fomento a
                ON f.CNPJ = a.CNPJ
            WHERE pe.ID_Pesq = %s;
        """, [idPerson])

        result = cursor.fetchall()
        cursor.close()
```

6.3.3 Menu Inicial - Buscar por Área

Além do SELECT para buscar todas as pesquisas relacionadas ao pesquisador logado, também há a função de buscar pela área de uma pesquisa:

```
    area = input("Area: ").upper()
    try:
        connection, cursor = con.GetConnectionAndCursor()
        cursor.execute("""
            SELECT E.TITULO, E.AREA, E.DATA_CRIACAO, P.descricao, U
.NOME AS Universidade, A.NOME AS AgenciaFomento, F.VALOR AS
ValorFinanciado
            FROM EXECUCAO E
            JOIN Universidade u
                ON e.CNPJ = u.CNPJ
            JOIN pesquisa p
              ON E.titulo = P.titulo
```

```
            AND E.area  = P.area
            AND E.data_criacao = P.data_criacao
        LEFT JOIN Financia f
            ON E.titulo     = f.Titulo
            AND E.AREA      = f.Area
            AND E.data_criacao = f.data_criacao
        LEFT JOIN Ag_Fomento a
            ON f.CNPJ = a.CNPJ
        WHERE E.AREA = %s;
    """, [area])

    result = cursor.fetchall()
    cursor.close()
```

Foi utilizado a função upper() aqui para que não precisássemos utilizar no SELECT ao buscar na tabela, pois isso dificultaria a busca pelos dados.

6.3.4 Menu Inicial - Adicionar Pesquisa

Além das demais funções, pode-se adicionar uma nova pesquisa. Fazemos operaçoes nas tabelas 'Pesquisa', 'Execucao', 'Pesq_Exec' (para cada um dos pesquisadores que irão participar da nova pesquias) e 'Financia' (se houver financiamento). A seguir, um recorte de todos os SQLs para essas tabelas:

```
 cursor.execute("""
    INSERT INTO PESQUISA (DATA_CRIACAO, TITULO, AREA, DESCRICAO)
    VALUES (TO_DATE(%s, 'DD/MM/YYYY'), %s, %s, %s)
 """, [dtCriacao, titulo, area, desc])

 ...

  cursor.execute("""
    INSERT INTO EXECUCAO (DATA_CRIACAO, TITULO, AREA, CNPJ, DATA_INICIO, DATA_CONCLUSAO)
    VALUES (TO_DATE(%s, 'DD/MM/YYYY'), %s, %s, %s, TO_DATE(%s, 'DD/MM/YYYY'), TO_DATE(%s, 'DD/MM/YYYY'))
    RETURNING ID;
```

```
    """, [dtCriacao, titulo, area, universidade, dtInicio,
dtConclusao])
    idExec = cursor.fetchone()[0]

    ...

    qtdePesq = int(input("Quantos pesquisadores comuns irao
participar: "))
    while(qtdePesq != 0):
        idPesq = input("ID pesquisador: ")
        dtIngresso = input ("Data de Inicio do pesquisador (dd/mm/
yyyy): ")
        dtSaida = input("Data de Conclusao do pesquisador (dd/mm/
yyyy): ")
        cursor.execute("""
            INSERT INTO PESQ_EXEC (ID_PESQ, ID_EXEC, DATA_INGRESSO,
 DATA_SAIDA)
            VALUES (%s, %s, TO_DATE(%s, 'DD/MM/YYYY'), TO_DATE(%s,
'DD/MM/YYYY'));
        """, [idPesq, idExec, dtIngresso, dtSaida])

        qtdePesq = qtdePesq - 1

    ...

cursor.execute("""
        INSERT INTO FINANCIA (CNPJ, DATA_CRIACAO, TITULO, AREA,
VALOR)
        VALUES (%s, TO_DATE(%s, 'DD/MM/YYYY'), %s, %s, %s);
    """, [agFomento, dtCriacao, titulo, area, valor])
```

Note que, na aplicação existem ‘if’s‘ e outras condições para que esses inserts sejam executados, apenas foi colocado aqui um recorte dos SQLs utilizados.

6.4 Tratamentos de Erros e Mal Usos

6.4.1 Erros provenientes do SGBD

Para erros provenientes do SGBD utilizamos a estrutura de 'try' e 'except', coletando os erros provenientes da própria biblioteca utilizada, como pode ser visto no exemplo abaixo:

```
    try:
        connection, cursor = con.GetConnectionAndCursor()
        ...

    except psycopg2.Error as error:
        connection.rollback()
        print("Erro:", error)

        opt = input("Deseja tentar novamente? [s/n]: ").lower()
        if opt == "s":
            sc.Cadastrar()
        else:
            sc.Sair()
```

6.4.2 Outros Tipos de Erros

Existiram alguns erros que acreditamos ser mais facilmente tratados na aplicação pelo python e não pelo SQL, como, por exemplo, erros de lógica de datas de diferentes tabelas. Por exemplo, em nossa aplicação ao adicionar uma pesquisa (6.3.4), temos de adicionar quando se inicia a pesquisa, mas também, quando efetivamente se inicia sua execução e não faria sentido que essa última data ocorresse antes da de inicio da pesquisa. Para erros desse gênero, foi criado uma classe de erro para sempre utilizarmos um raise para chamar para esse erro.

```
    class DateError(Exception):
        pass

    ...

```

```
6       if (datetime.strptime(dtCriacao, "%d/%m/%Y") > datetime.
    strptime(dtInicio, "%d/%m/%Y")):
7           raise errors.DateError("A data de inicio da execucao nao
    deve ser antes da data de criacao da pesquisa...")
8
9       ...
10
11      except errors.DateError as e:
12          connection.rollback()
13          print("Erro: ", e)
14          input("Pressione qualquer tecla para voltar... ")
15          sc.Menu(idPerson)
```

6.4.3 SQL Injection

Para prevenir possíveis entradas maliciosas por SQL Injection utilizamos uma parametrização especial no python em todas as execuções feitas no banco de dados. Podemos mostrar como exemplo o seguinte código:

```
1       sql = """
2           INSERT INTO PESSOA (CPF, NOME, UF, CIDADE, BAIRRO, RUA,
    NUMERO, TELEFONE1, TELEFONE2, DATA_NASC, SENHA)
3           VALUES (
4               %(cpf_bind)s, %(nome_bind)s, %(uf_bind)s, %(cidade_bind
    )s, %(bairro_bind)s, %(rua_bind)s, %(numero_bind)s, %(
    telefone1_bind)s, %(telefone2_bind)s, TO_DATE(%(data_nasc_bind)s
    , 'DD/MM/YYYY'), %(senha_bind)s)
5               RETURNING ID;
6           """
```

6.4.4 Proteção das credenciais da base de dados

Para que as credenciais não ficassem expostas dentro do código foi utilizado a biblioteca dotenv e a criação de um arquivo .env que cria variáveis de ambiente que podem ser usadas no código. Utilizando essas variáveis nossa função de conexão se manteve da seguinte maneira:

```
1       load_dotenv()
```

```
2
3     def GetConnectionAndCursor():
4         conn = psycopg2.connect(
5             host=os.getenv("DB_HOST"),
6             port=os.getenv("DB_PORT"),
7             database=os.getenv("DB_NAME"),
8             user=os.getenv("DB_USER"),
9             password=os.getenv("DB_PASSWORD")
10        )
11        return conn, conn.cursor()
```

6.4.5 LGPDs

Para manter sigilosas as informações das pessoas e hospitais, clínicas e laboratórios em nosso projeto. Quando selecionamos na aplicação alguma pesquisa para ver seu detalhamento, não mostramos nenhum dado sensível. Mostramos apenas as informações que podem ser vistas na figura (7).

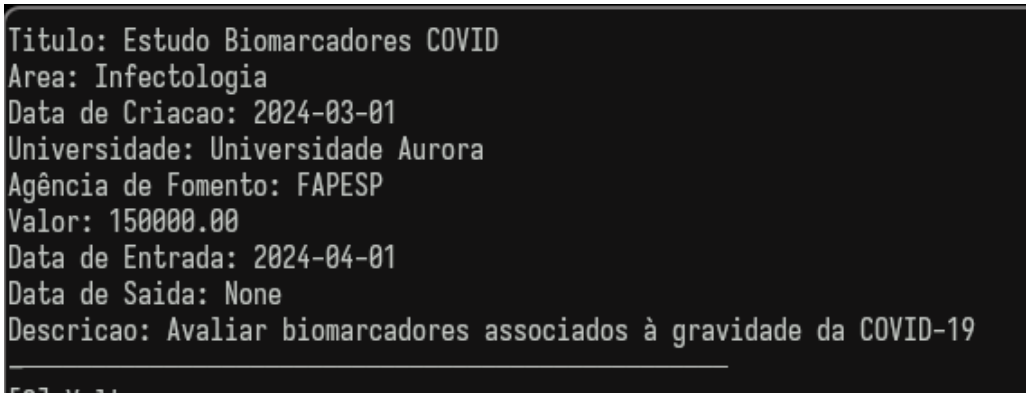

Figura 7: Tela de detalhamento de pesquisa. Fonte: autores.

Além disso, exceto no momento de cadastro, em nenhum outro momento é disponibilizado por exemplo o CPF de alguém. Ao cadastrar algum pesquisador em uma pesquisa, por exemplo, é requerido apenas seu ID.

6.4.6 Garantias em Aplicação

- **Garantir Especialização Total de Pessoa:** como não pudemos garantir pela aplicação que toda pessoa cadastrada teria uma função, isso foi feito

através da aplicação:

```
1     if ((coren == None) and (crm == None) and (pesquisadorR ==
    "n") and (pesquisadorC == "n") and (paciente == "n")):
2         print("Voce precisa escolher um papel (medico,
    enfermeiro, pesquisador responsavel, pesquisador comum ou
    paciente) para se cadastar...")
3         input("Pressione qualquer tecla para continuar...")
4         sc.Iniciar()
5
```

- **Garantir Especialização Total e Disjunção de Pesquisador:** garantimos que, caso a pessoa seja um pesquisador, ela tem que escolher entre o pesquisador responsável ou pesquisador comum, isso ocorre logo nos inputs:

```
1         print("Quais papeis voce tera no sistema?")
2         pesquisadorR = input("Pesquisador Responsavel [s/n]: ")
3         if pesquisadorR == "n":
4             pesquisadorC = input("Pesquisador Comum [s/n]: ")
5
```

- **Relações N:N com participação total:** existem uma relação N:N com participação total em nosso MER (5), entre universidade e pesquisa que é garantido da seguinte forma:

```
1          universidade = input("CNPJ da Universidade: ") or None
2         if(universidade == None):
3             print("ERRO: voce nao pode deixar o campo
    universidade vazio...")
4             input("Pressione qualquer tecla para voltar")
5             sc.Menu(idPerson)
6
```

7 Apresentação das Inserções e Consultas

7.1 Inserções de Exemplo

Nesta seção serão mostradas alguns tipos de inserções em algumas tabelas para exemplificar o funcionamento de nossas tabelas.

7.1.1 Tabela Amostra

```
1  INSERT INTO amostra (categoria, data_coleta, id_pac, validade,
   info, estado, qtde, id_enf) VALUES
2  ('Sangue', '2024-05-10', 15, '2024-05-20', 'Amostra rotulada A'
   , 'CONSERVADO', 1, 1),
3  ('Sangue', '2024-06-01', 16, '2024-06-11', 'Amostra rotulada B'
   , 'CONSERVADO', 2, 1),
4  ('Saliva','2024-07-12', 17, '2024-07-22', 'Amostra rotulada C',
   'ALTERADO', 1, 1);
```

7.1.2 Tabela Tipo_Coleta

```
1  INSERT INTO tipo_coleta (categoria, periculosidade) VALUES
2  ('Sangue', 'ALTA'),
3  ('Saliva', 'BAIXA');
```

7.2 Consultas de Exemplo

Os códigos .sql estão comentados, porém os comentários estão omitidos no relatório e serão substituidos por uma breve explicação de funcionamento das pesquisas:

7.2.1 Consulta 1

Objetivo: Exibir a quantidade de dados de amostras que uma pesquisa ainda não devolveu.

Explicação: A consulta inicia selecionando as informações básicas da pesquisa e realiza junções com as tabelas `EXECUCAO` e `DISPONIBILIZACAO`. Em seguida, filtra apenas os registros cuja data de devolução é nula ou ainda não ocorreu. O uso

de `COUNT(*)` permite contabilizar o total de amostras não devolvidas por pesquisa. Os resultados são agrupados e ordenados em ordem decrescente da quantidade de pendências.

```
SELECT
    P.data_criacao,
    P.titulo,
    P.area,
    COUNT(*) AS qtd_n_devolvida
FROM PESQUISA P
JOIN EXECUCAO E
      ON E.data_criacao = P.data_criacao
     AND E.titulo       = P.titulo
     AND E.area         = P.area
JOIN DISPONIBILIZACAO D
      ON D.id_exec = E.id
WHERE (D.data_devol IS NULL OR CURRENT_DATE < D.data_devol)
GROUP BY
    P.data_criacao,
    P.titulo,
    P.area
ORDER BY qtd_n_devolvida DESC;
```

7.2.2 Consulta 2

Objetivo: Encontrar as 5 pesquisas que mais utilizaram amostras.

Explicação: A consulta percorre toda a cadeia de tabelas relacionadas a amostras, conectando `PESQUISA`, `EXECUCAO`, `DISPONIBILIZACAO`, `DADOS_AMOSTRA` e `AMOSTRA`. O campo `COUNT(A.id)` calcula quantas amostras foram utilizadas em cada pesquisa. Após o agrupamento pelos atributos identificadores da pesquisa, os resultados são ordenados em ordem decrescente e limitados aos cinco maiores valores.

```
SELECT
    P.*,
    COUNT(A.id) AS total_amostras
FROM PESQUISA P
JOIN EXECUCAO E
```

```
      ON E.titulo         = P.titulo
     AND E.area           = P.area
     AND E.data_criacao   = P.data_criacao
JOIN DISPONIBILIZACAO D
      ON D.id_exec = E.id
JOIN DADOS_AMOSTRA DD
      ON DD.id = D.id_dados
JOIN AMOSTRA A
      ON A.id = DD.id_amostra
GROUP BY
    P.data_criacao,
    P.titulo,
    P.area,
    P.descricao
ORDER BY
    total_amostras DESC
LIMIT 5;
```

7.2.3 Consulta 3

Objetivo: Para cada combinação de mês/ano e estado (UF), gerar estatísticas sobre amostras e pacientes.

Explicação: A consulta utiliza funções de formatação de data para extrair mês e ano da coleta das amostras. Com a junção da tabela `PESSOA`, obtém-se o estado dos pacientes. São calculadas estatísticas como número de amostras, número de pacientes distintos, média, mínimo e máximo da quantidade coletada. O agrupamento é realizado por mês, ano e UF, e o resultado é ordenado seguindo essa mesma sequência.

```
SELECT
  TO_CHAR(A.data_coleta, 'mm') AS mes,
    TO_CHAR(A.data_coleta, 'YYYY') AS ano,
    P.uf                              AS uf_pacientes,
    COUNT(*)                          AS tipos_amostras_distintas,
    COUNT(DISTINCT A.id_pac)          AS pacientes_distintos,
    AVG(A.qtde)                        AS media_qtde,
    MIN(A.qtde)                        AS min_qtde,
```

```
    MAX(A.qtde)                          AS max_qtde
FROM AMOSTRA A
JOIN PESSOA P
    ON A.id_pac = P.id
GROUP BY
    TO_CHAR(A.data_coleta, 'YYYY'),
    TO_CHAR(A.data_coleta, 'mm'),
    P.uf
ORDER BY
    mes,ano,uf_pacientes;
```

7.2.4 Consulta 4

Objetivo: Encontrar pesquisas que receberam financiamento acima da média geral das agências de fomento.

Explicação: Por meio de uma CTE, a consulta calcula a média de todos os valores de financiamento. Essa média é combinada com cada financiamento registrado por `CROSS JOIN`. Em seguida, são retornadas apenas as pesquisas cujo valor de financiamento supera a média obtida. O resultado inclui informações da pesquisa, o valor recebido e a agência financiadora, ordenados pelos maiores valores depois das pesquisas.

```
WITH media_valores AS (
    SELECT AVG(valor) AS valor_medio
    FROM FINANCIA
)
SELECT
    P.data_criacao,
    P.titulo,
    P.area,
    F.valor AS valor_financiamento_pesquisa,
    F.cnpj as ag_fomento
FROM PESQUISA P
JOIN FINANCIA F
    ON F.titulo = P.titulo
   AND F.area = P.area
   AND F.data_criacao = P.data_criacao
```

```
CROSS JOIN media_valores M
WHERE F.valor > M.valor_medio
ORDER BY P.data_criacao,P.titulo,F.valor DESC;
```

7.2.5 Consulta 5

Objetivo: Retornar todas as pesquisas relacionadas a um estabelecimento que forneceu dados de pacientes de todos os estados brasileiros.

Explicação: A consulta percorre as tabelas `PESQUISA`, `EXECUCAO`, `DISPONIBILIZACAO`, `DADOS_AMOSTRA`, `AMOSTRA`, `PACIENTE` e `PESSOA` para identificar a UF de cada paciente envolvido. Após agrupar os resultados por pesquisa, utiliza-se um `HAVING` para verificar se o número de estados distintos encontrados é igual a 27. Apenas pesquisas vinculadas a um estabelecimento que possui abrangência nacional completa são retornadas.

```
SELECT
    P.*
FROM PESQUISA P
JOIN EXECUCAO E
      ON E.titulo        = P.titulo
     AND E.area          = P.area
     AND E.data_criacao  = P.data_criacao
JOIN DISPONIBILIZACAO D
      ON D.id_exec = E.id
JOIN DADOS_AMOSTRA DA
      ON DA.id = D.id_dados
JOIN AMOSTRA A
      ON A.id = DA.id_amostra
JOIN PACIENTE Pa
      ON Pa.id = A.id_pac
JOIN PESSOA Pe
      ON Pe.id = Pa.id
JOIN LAB_EXT LE
      ON LE.id = DA.id_estab
    P.data_criacao,
    P.titulo,
    P.area,
```

```
    P.descricao
HAVING
    COUNT(DISTINCT Pe.uf) = 27;
```

7.2.6 Consulta 6

Objetivo: Exibir as universidades que possuem vínculo, via pesquisas executadas, com todas as agências de fomento registradas.

Explicação: A consulta relaciona `UNIVERSIDADE` com `EXECUCAO`, `FINANCIA` e `AG_FOMENTO`, permitindo identificar quais agências financiaram pesquisas executadas por cada universidade. Após o agrupamento, o `HAVING` compara a quantidade de agências distintas associadas à universidade com o total de agências cadastradas no sistema. Apenas universidades com vínculo completo são retornadas.

```
SELECT
    U.*
FROM UNIVERSIDADE U
JOIN EXECUCAO E
      ON U.cnpj = E.cnpj
JOIN FINANCIA F
      ON F.titulo       = E.titulo
     AND F.area         = E.area
     AND F.data_criacao = E.data_criacao
JOIN AG_FOMENTO A
      ON A.cnpj = F.cnpj
GROUP BY
    U.cnpj,
    U.nome
HAVING
    COUNT(DISTINCT F.cnpj) = (
        SELECT COUNT(DISTINCT cnpj)
        FROM AG_FOMENTO);
```

8 Conclusão

Em conclusão, nesse trabalho tivemos a oportunidade de atravessar todos os conteúdos relevantes para a construção e manutenção de base de dados. Iniciamos por entender como conseguir o máximo possível de informações sobre o sistema que vamos criar ao fazer o levantamento de requisitos de dados (1.3), o que nos mostrou um caminho direto para onde seguir e uma ideia inicial do que seria implementado a seguir.

Com esses dados em mãos passamos para a criação do MER (2), nele estruturamos melhor as relações entre as diversas entidades do projeto e como atingir todos os objetivos esperados do nosso programa. Aqui também foi um desafio definir todas as conexões - com o menor número de ciclos possíveis - e os atributos pertinentes à estrutura.

Após isso, passamos para pequenas correções e a estruturação do modelo relacional (4), nosso projeto já se aproximando mais da implementação em SQL. Aqui construímos uma ideia mais clara de como seriam as tabelas e as conexões e identificação entre elas a partir de chaves estrangeiras, primárias, secundárias, etc. Foi aqui também que começamos nossas preocupações com as LGPDs e o que teríamos que fazer para nos certificar que informações sensíveis dos cadastrados no banco não teriam fácil acesso daqueles que não deveriam. Nessa parte do projeto foi necessário decidir entre diversas maneiras de modelar as diferentes situações e tentamos encontrar a melhor solução para cada problema.

Enfim, o projeto foi passado para a parte de implementação (6), com o nosso modelo relacional já corrigido, conseguimos com facilidade criar os a estrutura da nossa base de dados em PostgreSQL. Para a aplicação utilizamos python como nossa linguagem de desenvolvimento pela sua simplicidade, dado que a aplicação não era o ponto principal do trabalho. Utilizamos bibliotecas como psycopg2 e dotenv para manter a conexão com o banco de dados e também para medidas de segurança (6.4). Dessa mesma forma, pensando na segurança e saúde da nossa base de dados, prevenimos SQLInjection através da parametrização das informações que o usuário passa pelos inputs.

Para terminar, à parte da aplicação foram criados três arquivos SQL: criação da tabela, inserções e consultas. Em especial algumas inserções e algumas consultas específicas podem ser observadas ao final do relatório (7), essas consultas tiveram o intuito de mostrar possíveis consultas complexas que poderiam ser implementadas em nossa aplicação, mostrando a importância de toda a teoria por trás da criação de uma base de dados.

Gostaria-se também de salientar que a parte mais desafiadora do projeto acredita-se ter sido a estruturação do problema, isto é, a estruturação do MER, dado que, à partir do MER corrigido e estando adequado, o resto se segue de forma mais mecânica.

Em conclusão, esse trabalho nos levou a entender profundamente a matéria de base de dados e colocar em prática toda a teoria passada em aula. Após uma discussão do grupo, pensamos que se pudéssemos melhorar algo na estruturação do trabalho seria nos primeiros passos ter pensado mais cuidadosamente sobre como funcionaria a aplicação e suas funcionalidades principais para facilitar e tornar mais claro o que era esperado da aplicação ao final do projeto.