Plataforma de Gestão de Cuidados para Idosos e Familiares

Informações do Aluno

Nome: Victor André Lopes Brasileiro

Matrícula: 202407269

Curso: Ciência da Computação

Descrição do Sistema

O sistema será desenvolvido no contexto da saúde e da assistência domiciliar, com foco na gestão dos cuidados prestados a pessoas idosas e no apoio direto às suas famílias. A ideia é oferecer uma solução que ajude a organizar melhor os agendamentos, registrar cuidadores, acompanhar os atendimentos e, principalmente, facilitar a comunicação com os familiares responsáveis.

Os principais envolvidos nesse processo são os próprios idosos, que merecem atenção e cuidado individualizado, e os cuidadores, profissionais ou auxiliares que realizam tarefas essenciais como higiene, alimentação e companhia no dia a dia. Também são parte fundamental os familiares ou responsáveis, que acompanham de perto a saúde e o bem-estar do idoso e precisam ter acesso claro e direto às informações sobre os atendimentos realizados.

Quando pensei neste projeto, foi justamente a partir de algumas situações que observei na vida real: a dificuldade em organizar os horários das visitas, a falta de um histórico claro sobre quem cuidou de quem, e a insegurança de muitos familiares que não conseguem acompanhar de perto o que está acontecendo. Por isso, meu principal objetivo é melhorar a comunicação e o relacionamento entre cuidadores, idosos e suas famílias, promovendo mais confiança, organização e clareza em todo o processo.

Com esse sistema, quero centralizar todas as informações em um só lugar, dados dos idosos, registros de atendimento e escalas dos cuidadores, de forma prática e acessível. A proposta é permitir agendamentos bem definidos, com histórico de quem realizou o cuidado, em que data e horário, trazendo mais transparência. Além disso, o sistema contará com um canal de feedback, para que cuidadores possam ser avaliados

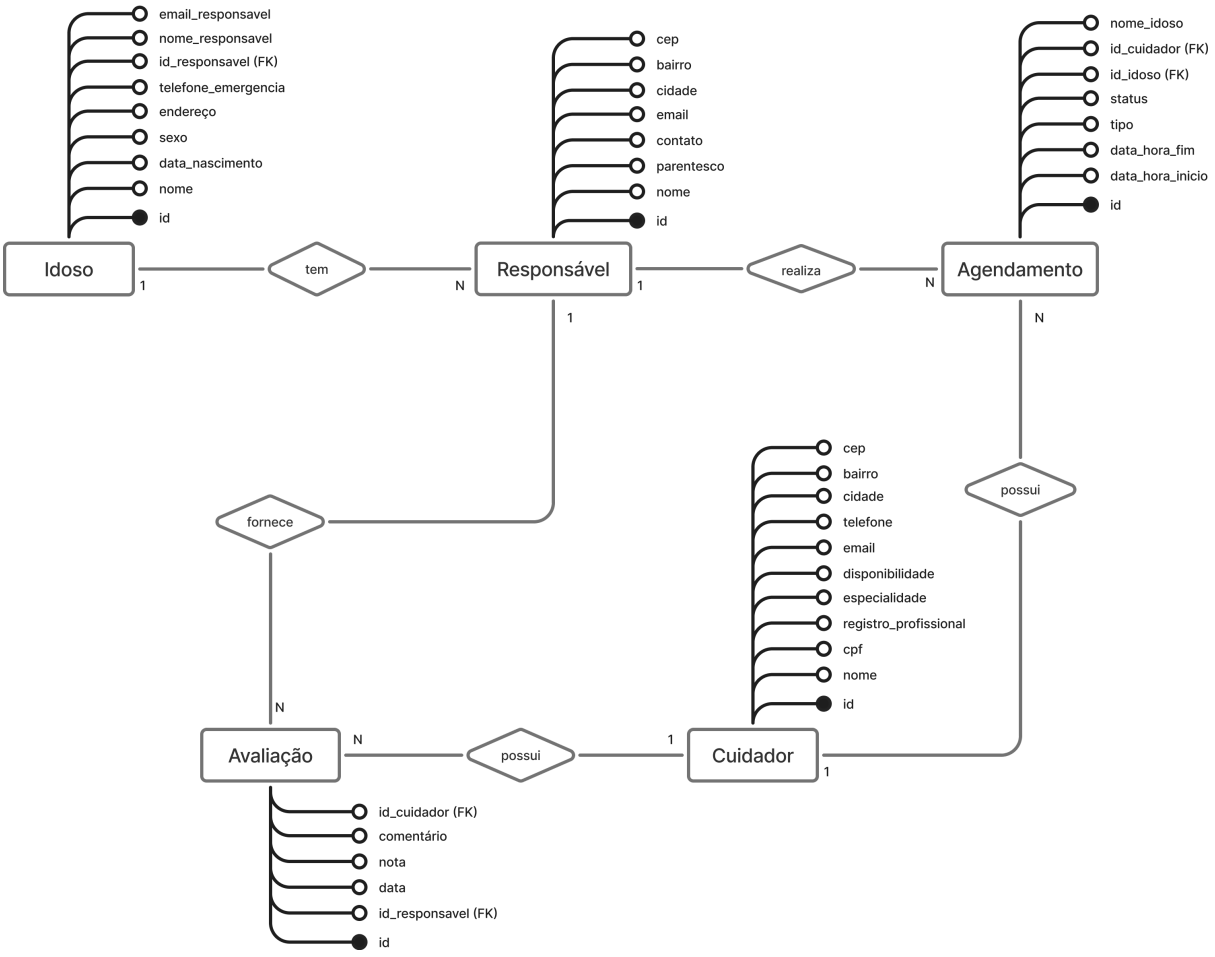
e os serviços aprimorados continuamente. Tudo isso visa oferecer mais segurança aos familiares e garantir que o cuidado com o idoso seja sempre acompanhado de perto, com carinho, responsabilidade e eficiência.

Requisitos do Sistema

- O sistema deve permitir cadastrar usuários com diferentes perfis: cuidador ou familiar.
- O sistema deve permitir cadastrar idosos com dados pessoais e contato de emergência.
- O sistema deve permitir cadastrar cuidadores, incluindo especialidade, disponibilidade e informações de contato.
- O sistema deve permitir registrar agendamentos de visitas, vinculando cuidador, idoso, tipo de atividade, data e horário.
- O sistema deve permitir editar e cancelar agendamentos existentes.
- O sistema deve permitir consultar o histórico de atendimentos realizados para cada idoso.
- O sistema deve permitir que familiares visualizem os agendamentos e visitas realizadas para o idoso sob sua responsabilidade.
- O sistema deve permitir o envio e visualização de feedbacks sobre os cuidadores, contendo nota e comentário.
- O sistema deve permitir filtrar agendamentos por data, cuidador ou idoso.
- O sistema deve gerar relatórios de visitas agendadas e realizadas, com filtros por período, status e profissional.

Diagrama Entidade-Relacionamento (MER)

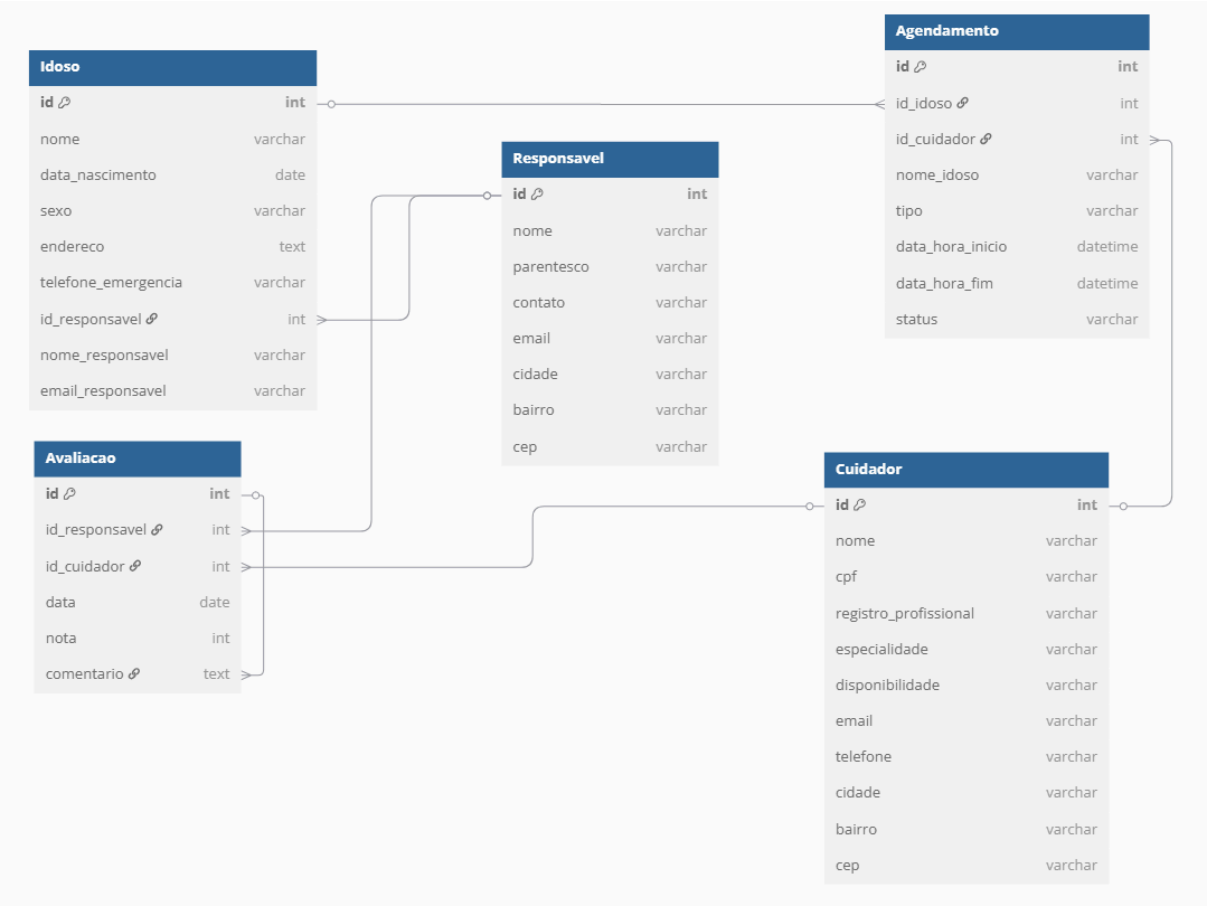
O MER desse projeto representa a estrutura de um sistema de gestão de cuidados domiciliares para idosos, com seis entidades principais: Idoso, responsável, cuidador, agendamento, avaliação e seus respectivos relacionamentos. Cada idoso é vinculado a um responsável, que realiza agendamentos de atendimentos executados por cuidadores. Após os atendimentos, o responsável pode registrar uma avaliação do cuidador.



Dependências Funcionais

Esquematização do MER em tabelas

Abaixo está o diagrama MER transformado em tabelas, para facilitar a visualização da estrutura do banco. Também mostro as chaves primárias, os atributos dependentes e como essas informações se conectam na álgebra relacional, ajudando a entender melhor a lógica por trás do modelo.



IDOSO

id	nome	data_nascimento	sexo	endereço	telefone_emergencia	id_responsavel	nome_responsavel	email_responsavel
1	Aldo Alves	19/01/1969	Masculino	Rua nascimento silva, 107, Ipanema, 22421-025 ,Rio de Janeiro, Brasil.	+55 21 99999-0000	1	Milton Jobim	jobim@jobim.com
2	Marcio Lopes	29/12/1952	Masculino	Rua São Franscisco, 1500, Serraria, 57044-878, Alagoas, Maceió, Brasil	+55 82 11223-0220	2	José Silva	silva@gmail.com

Chaves primárias: id

Chaves estrangeiras: id_responsavel

Dependências funcionais:

id → nome, data_nascimento, sexo, endereço, telefone_emergencia, id_responsavel

id_responsavel → nome_responsavel, email_responsavel

Exemplo de álgebra relacional:

1. σ <sexo = 'Masculino'> (IDOSO)
2. π <endereço, telefone_emergencia> (IDOSO)

Na tabela Idoso, id é a chave primária e determina todos os outros atributos diretamente. No entanto, existe uma dependência funcional transitiva: o campo id_responsavel determina nome_responsavel e email_responsavel.

RESPONSÁVEL

id	nome	parentesco	contato	email	cidade	bairro	cep
1	Milton Jobim	Filho	+55 21 12229-0000 +55 21 99999-0000	jobim@jobim.com	Rio de Janeiro	Ipanema	22421-025
2	José Silva	Irmão	+55 82 11223-0220	silva@gmail.com	Maceió	Jaraguá	12310-121

Chaves primárias: id

Dependências funcionais:

id → nome, parentesco, contato, email, cep

cep → bairro, cidade

Exemplo de álgebra relacional:

1. σ <parentesco = 'Filho'> (RESPONSAVEL)
2. π <nome> (RESPONSAVEL)

Na tabela Responsável, id é a chave primária e determina todos os outros atributos diretamente. No entanto, existe uma dependência funcional transitiva: o campo cep determina bairro e cidade.

CUIDADOR

id	nome	cpf	registro_profissional	especialidade	disponibilidade	email	telefone	cidade	bairro	cep
1	Maria José	999.999.999-99	12328829-12	Enfermagem Nutrição	Seg, Qua, Sex - 08h às 12h	maria@maria.com	+55 82 999987-2212	Maceió	Tabuleiro	231232-767

Chaves primárias: id

Dependências funcionais:

id → nome, cpf, registro_profissional, especialidade, disponibilidade, email, telefone, cep

cep → bairro, cidade

Exemplo de álgebra relacional:

1. σ <especialidade = 'Enfermagem'> (CUIDADOR)
2. σ <cidade = 'Maceió'> (CUIDADOR)

Na tabela Cuidador, o campo id é a chave primária e determina diretamente todos os demais atributos. No entanto, existe uma dependência funcional transitiva: o campo cep determina bairro e cidade.

AGENDAMENTO

id	data_hora_inicio	data_hora_fim	tipo	status	id_idoso	nome_idoso	id_cuidador
1	12/04/2024-12:00	12/05/2024-12:00	Acompanhamento diário	Pendente	1	Aldo Alves	1

Chaves primárias: id

Dependências funcionais:

id → data_hora_inicio, data_hora_fim, tipo, status, id_idoso, id_cuidador

id_idoso → nome_idoso

Exemplo de álgebra relacional:

1. σ <status = 'Concluido'> (AGENDAMENTO)

Na tabela Agendamento, o atributo id é a chave primária e determina todos os demais atributos. No entanto, existe uma dependência funcional transitiva: o campo nome_idoso depende de id_idoso.

AVALIAÇÃO

id	data	nota	comentario	id_responsavel	id_cuidador
1	31/12/2004	5	Ótimo profissional!	1	1

Chaves primárias: id

Dependências funcionais:

id → data, nota, comentario, id_responsavel, id_cuidador

Exemplo de álgebra relacional:

1. σ <nota ≥ 4> (AVALIACAO)

Na tabela Agendamento, o atributo id é a chave primária e determina todos os demais atributos.

Normalização

Primeira forma normal (1FN)

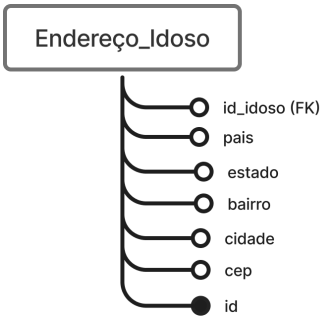
“A 1FN garante que cada tabela em um banco de dados tenha apenas valores atômicos em suas colunas”

Tabela IDOSO

id	nome	data_nascimento	sexo	endereço	telefone_emergencia	id_responsavel	nome_responsavel	email_responsavel
1	Aldo Alves	19/01/1969	Masculino	Rua nascimento silva, 107, Ipanema, 22421-025 ,Rio de Janeiro, Brasil.	+55 21 99999-0000	1	Milton Jobim	jobim@jobim.com
2	Marcio Lopes	29/12/1952	Masculino	Rua São Franscisco, 1500, Serraria, 57044-878, Alagoas, Maceió, Brasil	+55 82 11223-0220	2	José Silva	silva@gmail.com

A tabela Idoso tem uma chave primária (id), garantindo o pilar de chave primária da 1FN. Mas o campo endereço guarda várias infos juntas (rua, número, cidade e cep), e isso quebra a regra de valor atômico por campo. Então, mesmo com chave primária, ela ainda não está 100% na 1FN.

Para resolver isso, vamos criar uma nova tabela exclusiva para armazenar as informações de endereço dos idosos.



Endereço_Idoso

id	cep	cidade	bairro	estado	pais	id_idoso
1	22421-025	Rio de Janeiro	Ipanema	Rio de Janeiro	Brasil	1

Com essa correção, a tabela Idoso passa a ficar assim, garantindo a 1FN:

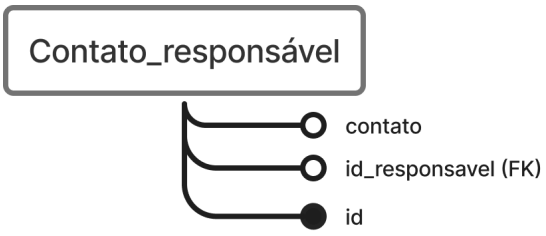
id	nome	data_nascimento	sexo	telefone_emergencia	id_responsavel	nome_responsavel	email_responsavel
1	Aldo Alves	19/01/1969	Masculino	+55 21 99999-0000	1	Milton Jobim	jobim@jobim.com
2	Marcio Lopes	29/12/1952	Masculino	+55 82 11223-0220	2	José Silva	silva@gmail.com

Tabela RESPONSÁVEL

id	nome	parentesco	contato	email	cidade	bairro	cep
1	Milton Jobim	Filho	+55 21 12229-0000 +55 21 99999-0000	jobim@jobim.com	Rio de Janeiro	Ipanema	22421-025
2	José Silva	Irmão	+55 82 11223-0220	silva@gmail.com	Maceió	Jaraguá	12310-121

A tabela Responsável tem uma chave primária (id), garantindo o pilar de chave primária da 1FN. Porém, quebra o princípio de valores atômicos, uma vez que o atributo contato é multivalorado.

Para resolver isso, vamos criar uma nova tabela exclusiva para armazenar as informações dos contatos dos responsáveis.



Contato_Responsável

id	id_responsavel	contato
1	1	+55 21 12229-0000
2	1	+55 21 99999-0000

Com essa correção, a tabela Responsável passa a ficar assim, garantindo a 1FN:

id	nome	parentesco	email	cidade	bairro	cep
1	Milton Jobim	Filho	jobim@jobim.com	Rio de Janeiro	Ipanema	22421-025
2	José Silva	Irmão	silva@gmail.com	Maceió	Jaraguá	12310-121

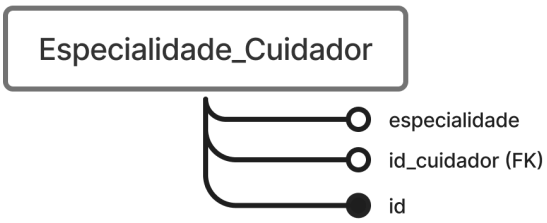
Observações: O atributo “email” não é considerado multivalorado, pois, em uma possível aplicação desse sistema, não será possível adicionar mais de um e-mail, já que, no cadastro do usuário responsável, esse e-mail será usado para realizar o login no sistema.

Tabela CUIDADOR

id	nome	cpf	registro_profissional	especialidade	disponibilidade	email	telefone	cidade	bairro	cep
1	Maria José	999.999.999-99	12328829-12	Enfermagem Nutrição	Seg, Qua, Sex - 08h às 12h	maria@maria.com	+55 82 999987-2212	Maceió	Tabuleiro	231232-767

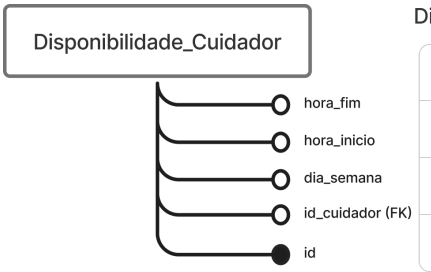
A tabela Responsável tem uma chave primária (id), garantindo o pilar de chave primária da 1FN. No entanto, os campos especialidade, disponibilidade e telefone podem violar o princípio de valores atômicos, já que podem conter múltiplas informações em um único campo.

Para resolver isso, vamos criar uma nova tabela exclusiva para armazenar as informações dos seguintes atributos:



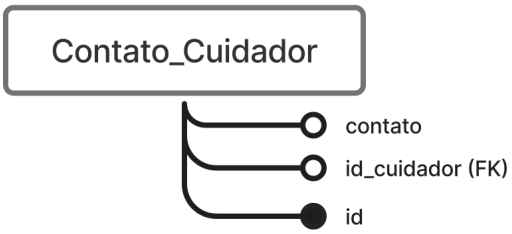
Especialidade_Cuidador

id	id_cuidador	especialidade
1	1	Enfermagem
2	1	Nutrição



Disponibilidade_Cuidador

id	id_cuidador	dia_semana	hora_inicio	hora_fim
1	1	Segunda	08:00	20:00
2	1	Quarta	08:00	20:00
3	1	Sexta	10:00	17:00



Contato_Cuidador

id	id_cuidador	contato
1	1	+55 82 999987-2212

Com essa correção, a tabela Responsável passa a ficar assim, garantindo a 1FN:

id	nome	cpf	registro_profissional	email	cidade	bairro	cep
1	Maria José	999.999.999-99	12328829-12	maria@maria.com	Maceió	Tabuleiro	231232-767

Tabela AGENDAMENTO e Tabela AVALIAÇÃO

As duas tabelas estão na primeira forma normal, uma vez que possuem chaves primárias e valores atômicos.

AGENDAMENTO

id	data_hora_inicio	data_hora_fim	tipo	status	id_idoso	nome_idoso	id_cuidador
1	12/04/2024-12:00	12/05/2024-12:00	Acompanhamento diário	Pendente	1	Aldo Alves	1

AVALIAÇÃO

id	data	nota	comentario	id_responsavel	id_cuidador
1	31/12/2004	5	Ótimo profissional!	1	1

Segunda forma normal (2FN)

"A 2FN garante que os atributos não-chave de uma tabela dependam totalmente da chave primária, eliminando dependências parciais"

Tabela IDOSO

id	nome	data_nascimento	sexo	telefone_emergencia	id_responsavel	nome_responsavel	email_responsavel
1	Aldo Alves	19/01/1969	Masculino	+55 21 99999-0000	1	Milton Jobim	jobim@jobim.com
2	Marcio Lopes	29/12/1952	Masculino	+55 82 11223-0220	2	José Silva	silva@gmail.com

Como a nossa tabela utiliza apenas o campo “id” como chave primária, não existem dependências parciais, apenas dependências totais e transitivas. Isso significa que todos os atributos não-chave dependem totalmente/transitivamente do “id”, e não de parte de uma chave composta. Por isso, a tabela já atende aos requisitos da Segunda Forma Normal (2FN).

Dependências Funcionais:

id → nome, data_nascimento, sexo, telefone_emergencia, id_responsavel

id_responsavel → nome_responsavel, email_responsavel

Tabela RESPONSÁVEL

id	nome	parentesco	email	cidade	bairro	cep
1	Milton Jobim	Filho	jobim@jobim.com	Rio de Janeiro	Ipanema	22421-025
2	José Silva	Irmão	silva@gmail.com	Maceió	Jaraguá	12310-121

Não existe dependência parcial, pois a chave primária é simples. Portanto, atende aos requisitos da 2FN.

Dependências funcionais:

id → nome, parentesco, email, cep

cep → bairro, cidade

Tabela CUIDADOR

id	nome	cpf	registro_profissional	email	cidade	bairro	cep
1	Maria José	999.999.999-99	12328829-12	maria@maria.com	Maceió	Tabuleiro	231232-767

Não existe dependência parcial, pois a chave primária é simples. Portanto, atende aos requisitos da 2FN.

Dependências funcionais:

id → nome, cpf, registro_profissional, especialidade, disponibilidade, email, telefone, cep

cep → bairro, cidade

Tabela AGENDAMENTO E AVALIAÇÃO

AGENDAMENTO

id	data_hora_inicio	data_hora_fim	tipo	status	id_idoso	nome_idoso	id_cuidador
1	12/04/2024-12:00	12/05/2024-12:00	Acompanhamento diário	Pendente	1	Aldo Alves	1

AVALIAÇÃO

id	data	nota	comentario	id_responsavel	id_cuidador
1	31/12/2004	5	Ótimo profissional!	1	1

Tanto na tabela de agendamento quanto na de avaliação, não existem dependências parciais, pois a chave primária utilizada em cada uma delas é simples. Dessa forma, ambas atendem aos requisitos da Segunda Forma Normal (2FN).

Terceira forma normal (3FN)

"A 3FN visa eliminar dependências transitivas entre os atributos não chave de uma tabela, garantindo que cada atributo dependa exclusivamente da chave primária

Tabela IDOSO

id	nome	data_nascimento	sexo	telefone_emergencia	id_responsavel	nome_responsavel	email_responsavel
1	Aldo Alves	19/01/1969	Masculino	+55 21 99999-0000	1	Milton Jobim	jobim@jobim.com
2	Marcio Lopes	29/12/1952	Masculino	+55 82 11223-0220	2	José Silva	silva@gmail.com

Nossa tabela infringe um dos requisitos da Terceira Forma Normal (3FN), pois dois atributos apresentam dependência transitiva, ou seja, não dependem diretamente da chave primária (id), mas sim do id_responsavel. Para atender à 3FN, seria necessário separar essas informações em outra tabela. No entanto, como já existe uma tabela específica para o responsável, basta remover esses atributos da tabela atual e manter as informações relacionadas apenas na tabela do responsável.

Correção:

id	nome	data_nascimento	sexo	telefone_emergencia	id_responsavel
1	Aldo Alves	19/01/1969	Masculino	+55 21 99999-0000	1
2	Marcio Lopes	29/12/1952	Masculino	+55 82 11223-0220	2

Dependências Funcionais:

id → nome, data_nascimento, sexo, telefone_emergencia, id_responsavel

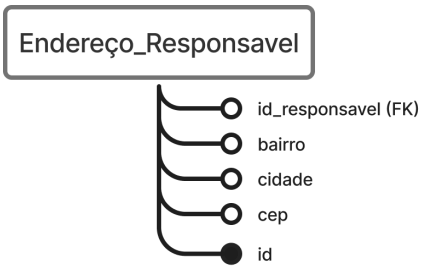
---

Tabela RESPONSÁVEL

id	nome	parentesco	email	cidade	bairro	cep
1	Milton Jobim	Filho	jobim@jobim.com	Rio de Janeiro	Ipanema	22421-025
2	José Silva	Irmão	silva@gmail.com	Maceió	Jaraguá	12310-121

A tabela de responsável também não atende a 3FN, pois os atributos cidade e bairro dependem de cep e não diretamente do id - uma dependência transitiva. Nesse caso, é

necessário criar uma tabela para armazenar as informações de endereço dos responsáveis.



Endereço_Responsavel

id	cep	cidade	bairro	id_responsavel
1	22421-025	Rio de Janeiro	Ipanema	1
2	12310-121	Maceió	Jaraguá	2

Correção:

id	nome	parentesco	email
1	Milton Jobim	Filho	jobim@jobim.com
2	José Silva	Irmão	silva@gmail.com

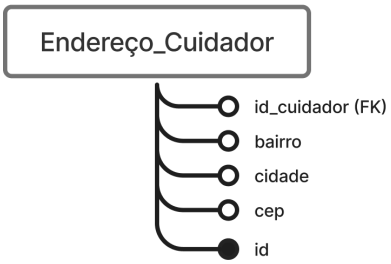
Dependências funcionais:

id → nome, parentesco, email

Tabela CUIDADOR

id	nome	cpf	registro_profissional	email	cidade	bairro	cep
1	Maria José	999.999.999-99	12328829-12	maria@maria.com	Maceió	Tabuleiro	231232-767

A tabela de cuidador também apresenta o mesmo problema da tabela de responsável, incluindo atributos com dependência transitiva, como bairro e cidade. Assim como foi feito para a tabela de responsável, será necessário criar uma tabela separada para armazenar o endereço do cuidador.



Endereço_Cuidador

id	cep	cidade	bairro	id_cuidador
1	231232-767	Maceió	Tabuleiro	1

Correção:

id	nome	cpf	registro_profissional	email
1	Maria José	999.999.999-99	12328829-12	maria@maria.com

Dependências funcionais:

id → nome, cpf, registro_profissional, email, telefone

Tabela AGENDAMENTO

AGENDAMENTO

id	data_hora_inicio	data_hora_fim	tipo	status	id_idoso	nome_idoso	id_cuidador
1	12/04/2024-12:00	12/05/2024-12:00	Acompanhamento diário	Pendente	1	Aldo Alves	1

A tabela de agendamento possui apenas um atributo que fere as restrições da 3FN, que é o nome_idoso. Para evitar a criação de uma nova tabela, basta remover esse atributo e manter essa informação apenas na tabela de idoso. Caso seja necessário consultar o nome do idoso em um agendamento, basta realizar um join utilizando o id_idoso para obter o nome desejado.

Correção:

id	data_hora_inicio	data_hora_fim	tipo	status	id_idoso	id_cuidador
1	12/04/2024-12:00	12/05/2024-12:00	Acompanhamento diário	Pendente	1	1

Dependências funcionais:

id → data_hora_inicio, data_hora_fim, tipo, status, id_idoso, id_cuidador

---

Tabela AVALIAÇÃO

id	data	nota	comentario	id_responsavel	id_cuidador
1	31/12/2004	5	Ótimo profissional!	1	1

A tabela de avaliação não possui dependências transitivas, atendendo assim aos requisitos da 3FN.

Dependências funcionais:

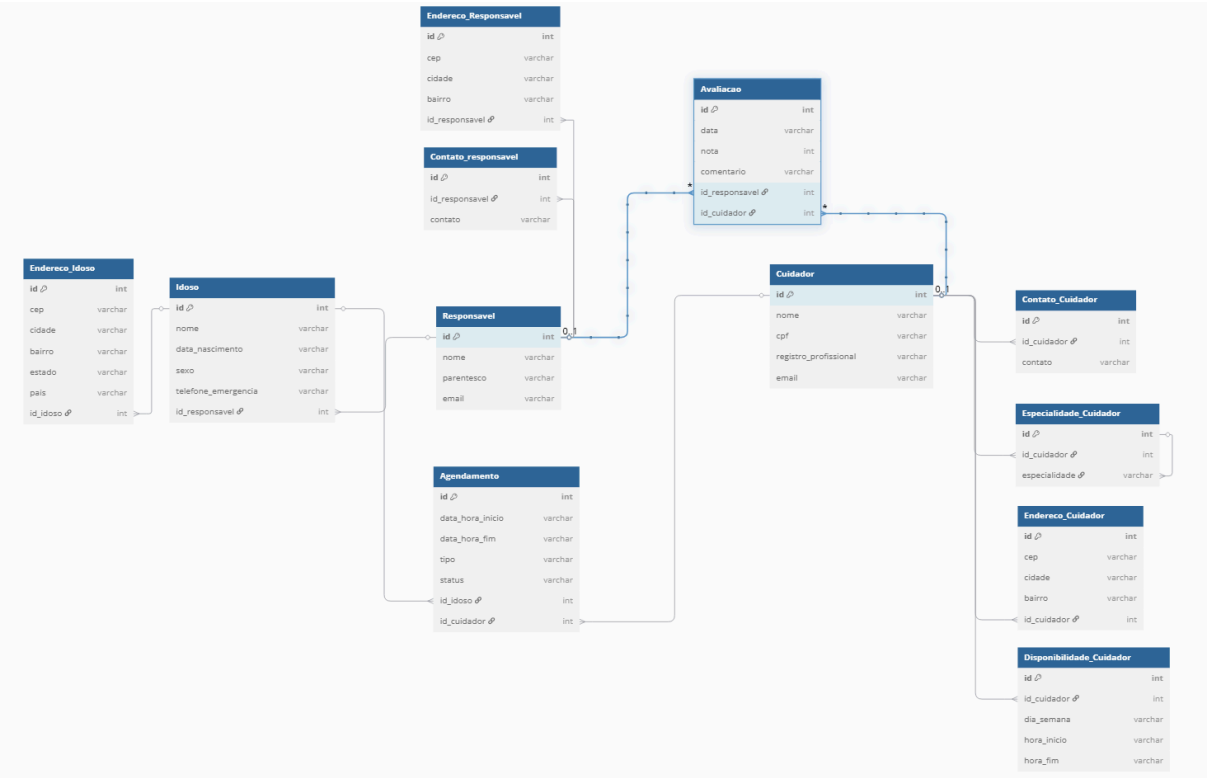
id → data, nota, comentario, id_responsavel, id_cuidador

Quarta Forma Normal (4FN) e Forma Normal de Boyce-Codd (BCNF)

“A 4FN visa eliminar dependências multivaloradas que não sejam determinadas por uma chave candidata e a BCNF visa eliminar redundâncias e inconsistências nos dados”

Analisando todas as alterações feitas até agora, dá para perceber que todas as tabelas e relacionamentos atendem tanto à 4FN quanto à BCNF. Ou seja, o modelo está totalmente normalizado e a parte de normalização já está concluída.

Resultado da Normalização



https://dbdiagram.io/d/682741d81227bdcb4ea5f554

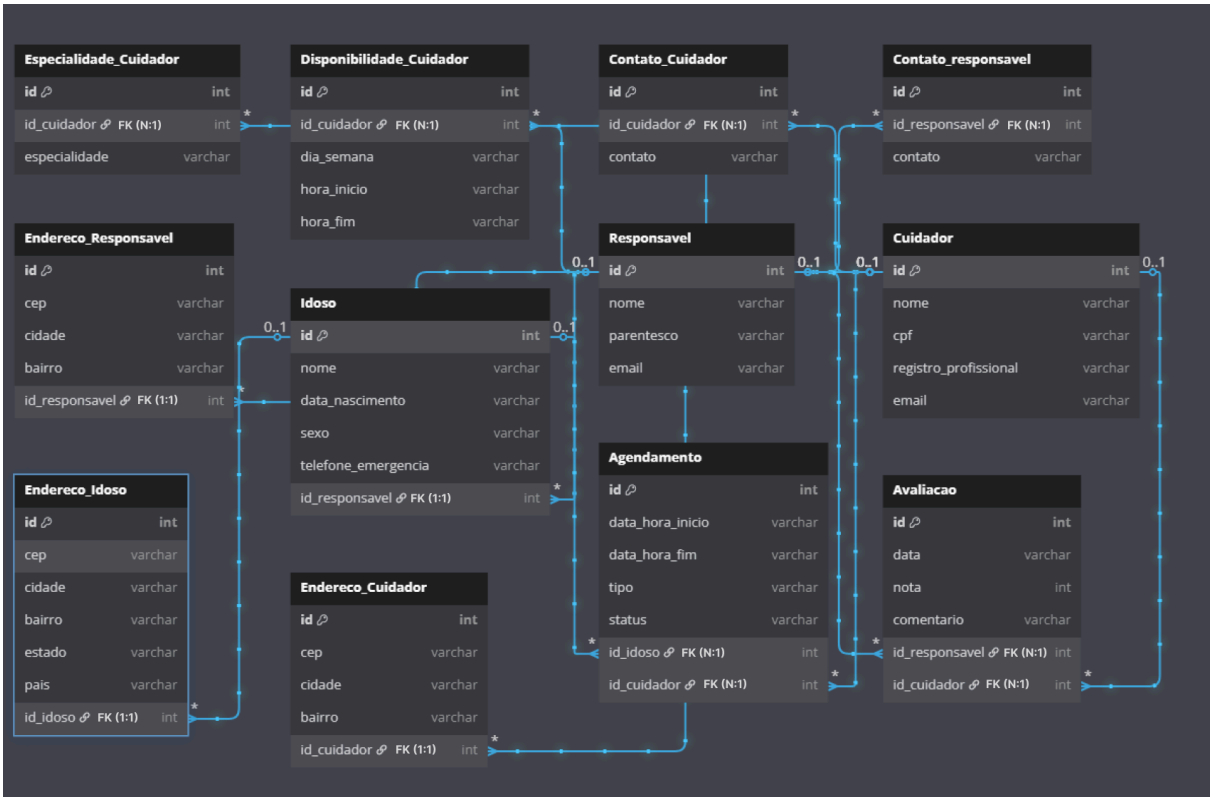
Considerações finais

Existem algumas tabelas repetidas, como as de endereço e contato. Isso aconteceu, pois cada uma delas está ligada a um tipo específico (idoso, cuidador ou responsável), usando um ID diferente para manter as relações corretas.

No futuro, dá pra simplificar e transformar em uma tabela única de endereços e outra de contatos, usando um campo para indicar a quem pertence aquele registro (se é do idoso, cuidador ou responsável). Assim, o modelo fica mais prático e fácil de manter caso precise expandir o sistema.

Modelo Lógico Relacional

Abaixo está o modelo lógico do projeto, apresentado em formato de tabelas relacionais, feito no site dbDiagram. O modelo mostra os nomes das entidades, atributos, tipos de dados, chaves primárias, chaves estrangeiras e suas cardinalidades. Também segue o link para visualizar o modelo diretamente no site.



https://dbdiagram.io/d/682741d81227bdcb4ea5f554

Consultas SQL

Após transformar nosso modelo lógico em SQL e criar todas as tabelas, vou realizar algumas operações para demonstrar as principais funcionalidades do sistema.

> 💡 ***Todos os códigos estarão no dump e no repositório do projeto.***

1. INSERT, UPDATE, DELETE

Para começar, vamos adicionar, atualizar e remover alguns dados para verificar se as operações básicas do sistema estão funcionando corretamente. Vou mostrar apenas algumas dessas operações aqui, para não deixar o PDF muito extenso. O código completo está disponível no repositório e no arquivo de dump.

```sql
-- 1.1 INSERÇÃO
INSERT INTO Responsavel (nome,parentesco,email)
  VALUES ('João Marques', 'Filho', 'joao@joao.com');

INSERT INTO Contato_Responsavel (telefone,id_responsavel)
  VALUES ('82999993231', 1);

INSERT INTO Cuidador (nome,cpf,registro_profissional,email)
  VALUES ('Matheus Cabral', '000.000.000-00', '1231422-12', 'matheus@gmail.com');

INSERT INTO Endereco_Cuidador (cep,cidade,bairro,id_cuidador)
  VALUES ('19864-821', 'Murici', 'Ouro Branco', 1);

INSERT INTO Especialidade_Cuidador (especialidade,id_cuidador)
  VALUES ('Enfermagem Dermatológica', 1);

INSERT INTO Disponibilidade_Cuidador (dia_semana,hora_inicio,hora_fim,id_cuidador)
  VALUES ('Segunda-feira','07:30:00','20:30:00', 1);

INSERT INTO Disponibilidade_Cuidador (dia_semana,hora_inicio,hora_fim,id_cuidador)
  VALUES ('Quarta-feira','07:30:00','20:30:00', 1);

INSERT INTO Idoso (nome,data_nASCimento,sexo,telefone_emergencia,id_responsavel)
  VALUES ('Aldo Alvez', '1969-12-02', 'MASCulino' ,'82998213221', 1);

INSERT INTO Agendamento (data_hora_inicio,data_hora_fim,tipo,status_consulta,id_cuidador,id_idoso)
  VALUES ('2025-04-28 09:00:00','2025-04-29 09:00:00','Diária Completa','Concluido',1,5);

INSERT INTO Avaliacao (data,nota,comentario,id_cuidador,id_responsavel)
  VALUES ('2025-04-18 09:00:00', 5, 'Ótima profissional' , 1, 3);

-- 1.2 REMOÇÃO
DELETE FROM Agendamento WHERE id = 4;
DELETE FROM Endereco_Idoso WHERE id = 4;
DELETE FROM Idoso WHERE id = 6;

-- 1.3 ALTERAÇÃO
UPDATE Cuidador SET email = 'elina@eliana.com' WHERE id = 3;
UPDATE Responsavel SET parentesco = 'Sobrinha' WHERE id = 7;
```

2. SELECT E JOIN

```sql
-- 1. Listar todas as informações dos idosos
SELECT * FROM idoso;

-- 2. Listar todos os idosos e seus responsaveis
SELECT
  idoso.nome as nome_idoso
  , responsavel.nome as nome_responsavel,
  responsavel.parentesco as parentesco_responsavel
  FROM idoso
  INNER JOIN responsavel ON idoso.id_responsavel = responsavel.id;

-- 3. Listar todos os cuidadores e seus contatos
SELECT
  cuidador.nome as nome_cuidador,
  contato_cuidador.telefone as contato_cuidador
  FROM cuidador
    LEFT JOIN contato_cuidador ON cuidador.id = contato_cuidador.id_cuidador;

-- 4. Listar apenas os agendamentos que já foram concluídos
  SELECT
    agendamento.data_hora_fim as horario_finalizacao_consulta,
    idoso.nome as nome_idoso, cuidador.nome as nome_cuidador
    FROM agendamento
        INNER JOIN idoso ON agendamento.id_idoso = idoso.id
        INNER JOIN cuidador ON agendamento.id_cuidador = cuidador.id
      WHERE agendamento.status_consulta = 'Concluido';

-- 5. Listar cuidadores e suas especializações
  SELECT
    cuidador.nome as nome_cuidador,
    especialidade_cuidador.especialidade as especializacao
    FROM cuidador
        INNER JOIN especialidade_cuidador ON cuidador.id = especialidade_cuidador.id_cuidador;
```

```sql
-- 6. Listar todos os idosos e agendamentos
  SELECT
    idoso.nome as nome_idoso,
    agendamento.data_hora_inicio, agendamento.status_consulta
    FROM idoso
        LEFT JOIN agendamento ON idoso.id = agendamento.id_idoso;

-- 7 Listar responsaveis e endereços
  SELECT
    responsavel.nome as nome_responsavel,
    endereco_responsavel.cidade, endereco_responsavel.bairro
    FROM endereco_responsavel
        RIGHT JOIN responsavel On endereco_responsavel.id_responsavel = responsavel.id;
```

```sql
-- 8 Listar cuidadores e endereços
  SELECT
    responsavel.nome as nome_responsavel,
    endereco_responsavel.cidade, endereco_responsavel.bairro
    FROM endereco_responsavel
        RIGHT JOIN responsavel On endereco_responsavel.id_responsavel = responsavel.id;

-- 9 Listar avaliacoes que possuem nota acima de 3.0
  SELECT *
    FROM avaliacao
      WHERE nota >= 3;

-- 10. Listar todas as avaliações e as informações dos idoso e responsaveis
  SELECT
    avaliacao.id, avaliacao.nota,
    cuidador.nome AS nome_cuidador,
    cuidador.registro_profissional as registro_cuidador,
    responsavel.nome AS nome_responsavel
    FROM avaliacao
    INNER JOIN cuidador ON avaliacao.id_cuidador = cuidador.id
    INNER JOIN responsavel ON avaliacao.id_responsavel = responsavel.id;
```

3. AGREGAÇÕES

```sql
-- 1. Total de cuidadores cadastrados
  SELECT COUNT(*) as total_cuidadores FROM cuidador;
-- 2. Total de idosos cadastrados
  SELECT COUNT(*) as total_idosos FROM idoso;
-- 3. Média das notas de avaliações
  SELECT AVG(nota) as media_nota FROM avaliacao;
-- 4. Média das idades dos idosos
  SELECT AVG(YEAR(CURDATE()) - YEAR(data_nascimento)) as media_idade FROM idoso;
-- 5. Total de agendamentos pendentes
  SELECT COUNT(*) as total_pendentes FROM agendamento WHERE status_consulta = 'Pendente';
-- 6. Maior e menor nota das avaliacoes
  SELECT MAX(nota) as nota_maxima, MIN(nota) as nota_minima FROM avaliacao;
-- 7. Total de avaliações
  SELECT COUNT(*) as total_avaliacoes FROM avaliacao;
-- 8. Total de contatos de responsaveis cadastrados
  SELECT COUNT(*) FROM contato_responsavel;
-- 9. Total de responsaveis
  SELECT COUNT(*) FROM responsavel;
-- 10. total de endereços de idosos cadastrados
  SELECT COUNT(*) FROM endereco_idoso;
```

4. ORDENAÇÕES

```sql
-- 1. Quantidade de avaliações por cuidador
SELECT id_cuidador, count(*) as total_avaliacoes
  FROM avaliacao
  GROUP BY id_cuidador;
-- 2. Média de notas por cuidador
SELECT id_cuidador, avg(nota) as media_nota
  FROM avaliacao
  GROUP BY id_cuidador;
-- 3. Quantidade de agendamentos por status --> ordenando do maior para o menor
SELECT status_consulta, count(*) as total_agendamentos
  FROM agendamento
  GROUP BY status_consulta
  ORDER BY total_agendamentos DESC;
-- 4. Cuidadores ordenados por nome
SELECT * FROM cuidador
  ORDER BY nome ASC;
-- 5. Idosos ordenados por idade
SELECT * FROM idoso
  ORDER BY data_nASCimento ASC;
-- 6. Cuidadores que possuem mais de uma especialização
SELECT id_cuidador, count(*) as total_especialidades
  FROM especialidade_cuidador
  GROUP BY id_cuidador
  HAVING count(*) > 1;
-- 7. 5 idosos mias novos
SELECT * FROM idoso
  ORDER BY data_nASCimento DESC
  LIMIT 5;
-- 8. Média das notas ordenadas
SELECT id_cuidador, avg(nota) as media_nota
  FROM avaliacao
  GROUP BY id_cuidador
  ORDER BY media_nota DESC;
-- 9. Total de contatos cadastrados dos cuidadores ordenados
SELECT id_cuidador, count(*) as total_contatos
  FROM contato_cuidador
  GROUP BY id_cuidador
  ORDER BY total_contatos DESC;
-- 10. Agendamentos ordenados por data de inicio
SELECT * FROM agendamento
  ORDER BY data_hora_inicio ASC;
```

5. SUBCONSULTAS

```sql
-- 1. Listar cuidadores que possuem avaliações
SELECT * FROM cuidador
  WHERE id in (SELECT id_cuidador FROM avaliacao);
```

```sql
-- 2. Listar responsáveis que possuem mais de um idosoo
SELECT * FROM responsavel
  WHERE id in (
      SELECT id_responsavel FROM idoso
      GROUP BY id_responsavel
      HAVING count(*) > 1
  );

-- 3. Listar idosos que não têm agendamento
SELECT * FROM idoso i
  WHERE NOT EXISTS (
      SELECT 1 FROM agendamento a
      WHERE a.id_idoso = i.id
  );

-- 4. Listar cuidadores que nunac receberam avaliação
SELECT * FROM cuidador
  WHERE id NOT IN (SELECT id_cuidador FROM avaliacao);

-- 5. Listar agendamentos que receberam nota 5
SELECT * FROM agendamento
  WHERE id_idoso in (
      SELECT id_idoso FROM avaliacao
      WHERE nota = (SELECT max(nota) FROM avaliacao)
  );

-- 6. Listar responsáveis que já avaliaram algum cuidador
SELECT distinct r.*
  FROM responsavel r
  WHERE EXISTS (
      SELECT 1 FROM avaliacao a
      WHERE a.id_responsavel = r.id and a.id_cuidador = 1
  );

-- 7. Listar cuidadores que tem mais de uma especialidade
SELECT * FROM cuidador
  WHERE id in (
      SELECT id_cuidador FROM especialidade_cuidador
      GROUP BY id_cuidador
      HAVING count(*) > 1
  );

-- 8. Listar todos os idosos que moram nos bairros dos cuidadores
SELECT * FROM idoso
  WHERE id in (
      SELECT e.id_idoso FROM endereco_idoso e
      WHERE e.bairro in (
          SELECT ec.bairro FROM endereco_cuidador ec
      )
  );

-- 9. Listar cuidadores que tem mais de umm agendamento concluído
SELECT * FROM cuidador c
  WHERE EXISTS (
      SELECT 1 FROM agendamento a
      WHERE a.id_cuidador = c.id and a.status_consulta = 'concluido'
  );

-- 10. Listar responsáveis que nunca cadastraram contato
SELECT * FROM responsavel r
  WHERE NOT EXISTS (
      SELECT 1 FROM contato_responsavel cr
      WHERE cr.id_responsavel = r.id
  );
```

Conclusão

Segue junto ao PDF o arquivo SQL com todas as operações demonstradas, além do dump completo do banco de dados. Chegando ao final desse projeto, deixo aqui algumas conclusões e observações.

O sistema, apesar de básico e ainda não preparado para grande escalabilidade, está bem estruturado, documentado e totalmente normalizado. Meu principal objetivo foi criar uma base sólida, que possa servir de ponto de partida para projetos futuros, onde pretendo aperfeiçoar os conceitos, implementar uma interface e construir um backend robusto. Quem sabe, no futuro, transformar essa ideia em uma aplicação completa!

Quero agradecer à professora pelos ensinamentos que me deram base e confiança para chegar até aqui, ao monitor pelo apoio e pelas correções ao longo do desenvolvimento, e também a todos os colegas que estiveram juntos comigo durante esse período na disciplina de banco de dados.

Repositório

https://github.com/VictorBrasileiroo/Projeto-BD-AB2