



Faculdade de Informática e Administração Paulista

**Data Application & Data Science**

**Challenge 2023**

## INTEGRANTES

<b>RM (SOMENTE NÚMEROS)</b>	<b>NOME COMPLEMENTO (SEM ABREVIAR)</b>
97068	Gustavo Sorrilha Sanches
97324	Natan Cruz
97092	Vitor Rubim Passos
97503	Mariana Santos Fernandes de Sousa
96466	Kaue Caponero Figueiredo



**SUMÁRIO**

- 1 – DESCRIÇÃO DO PROJETO E REGRAS DE NEGÓCIO
- 2 – Projeto Lógico
- 3 – Projeto Físico
- 4 – Prints das evidências da criação dos objetos no Banco
- 5 – Prints das evidências da carga de dados
- 6 – Prints das evidências da execução das duas pesquisas

## **1 – Descrição do Projeto e Regras de Negócio**

HealthHear é uma solução móvel inovadora que visa transformar a interação entre pacientes e profissionais da saúde. Permitindo feedbacks e reclamações anônimas ou identificadas, o aplicativo proporciona transparência e confiança na escolha de serviços médicos. Com foco na melhoria contínua, HealthHear utiliza tecnologia avançada para oferecer uma experiência segura e informada aos usuários, incentivando práticas médicas de alta qualidade.

Ao facilitar o compartilhamento de experiências, HealthHear busca construir um ambiente onde pacientes se sintam capacitados a avaliar médicos e serviços de saúde, promovendo transparência e confiança na comunidade médica. A plataforma utiliza recursos tecnológicos de ponta para analisar feedbacks, identificar tendências e áreas de aprimoramento, visando sempre à qualidade dos serviços oferecidos e à segurança dos pacientes.

Com a missão de promover uma relação mais transparente e segura entre pacientes e profissionais da saúde, HealthHear aspira a criar um ecossistema de cuidados médicos onde a confiança e a qualidade dos serviços sejam constantemente aprimoradas, proporcionando uma jornada de cuidados mais consciente e confiável para todos os envolvidos.

## Regras de Negócio

### Política de Cadastro (RN01):

Cada paciente e profissional de saúde deve possuir apenas uma conta na plataforma, identificada por um e-mail único e CPF para todos os usuários e tipo de registro (CRM, CRO, CRN etc.), número do registro e o UF para os profissionais.

Todas as informações de cadastro devem passar por um processo de validação para garantir sua autenticidade e integridade no banco de dados.

### Política de Comunicação (RN02):

O sistema deve permitir que pacientes e profissionais de saúde se comuniquem de forma eficaz por meio de feedbacks e respostas registrados.

### Política de Registro (RN03):

Deve existir um sistema de registro detalhado (logs visuais) para acompanhar as atividades, como feedbacks, reclamações, respostas e interações entre usuários.

### Política de Avaliação de Usuários (RN04):

Os pacientes podem avaliar os profissionais de saúde após a consulta ou tratamento, atribuindo uma nota e compartilhando experiências.

O sistema deve garantir que essas avaliações sejam transparentes e acessíveis a outros usuários, contribuindo para a confiabilidade das interações na plataforma.

#### Política de Privacidade e Confidencialidade (RN05):

Todas as informações dos usuários, incluindo feedbacks, mensagens e registros médicos, devem ser protegidas por medidas de segurança robustas.

A plataforma deve garantir a privacidade e confidencialidade dos dados dos usuários, seguindo normas e regulamentações de proteção de dados vigentes.

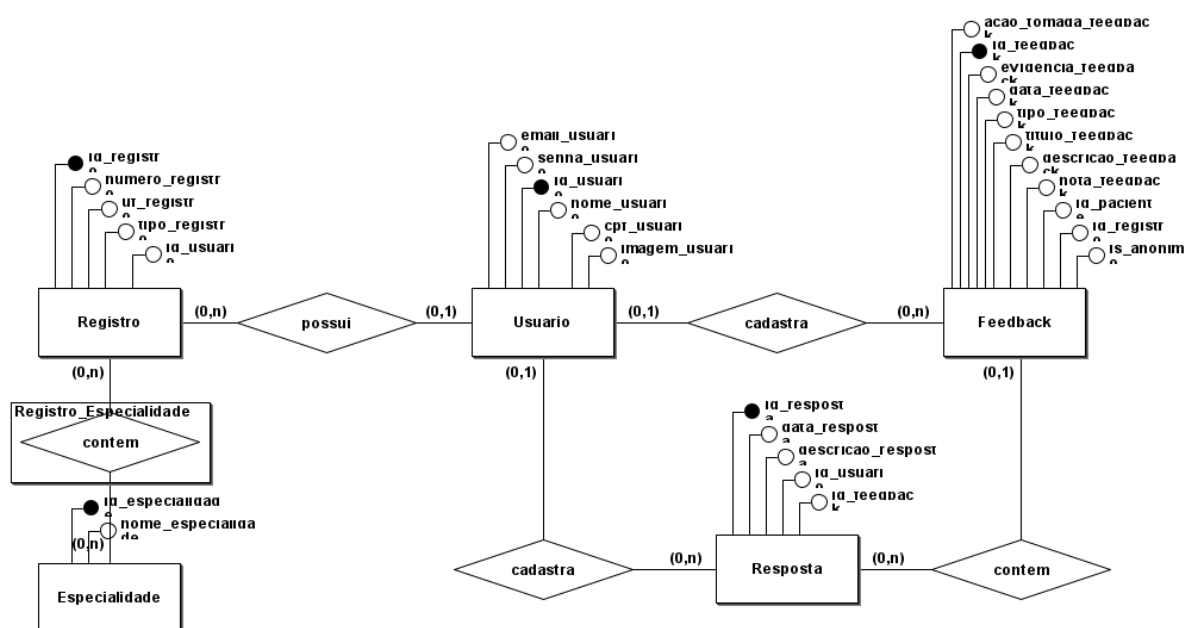
Os feedbacks podem ser anônimos para garantir a integridade dos dados do paciente, caso este deseje.

#### Política de Cadastramento de Registro (RN06):

O sistema deve permitir que os usuários cadastrem registros, vinculando-os a si ou não. Isso possibilita que os pacientes cadastrem feedbacks apenas com informações do registro, enquanto os profissionais da saúde podem cadastrar-se ou vincular-se a um registro já existente.

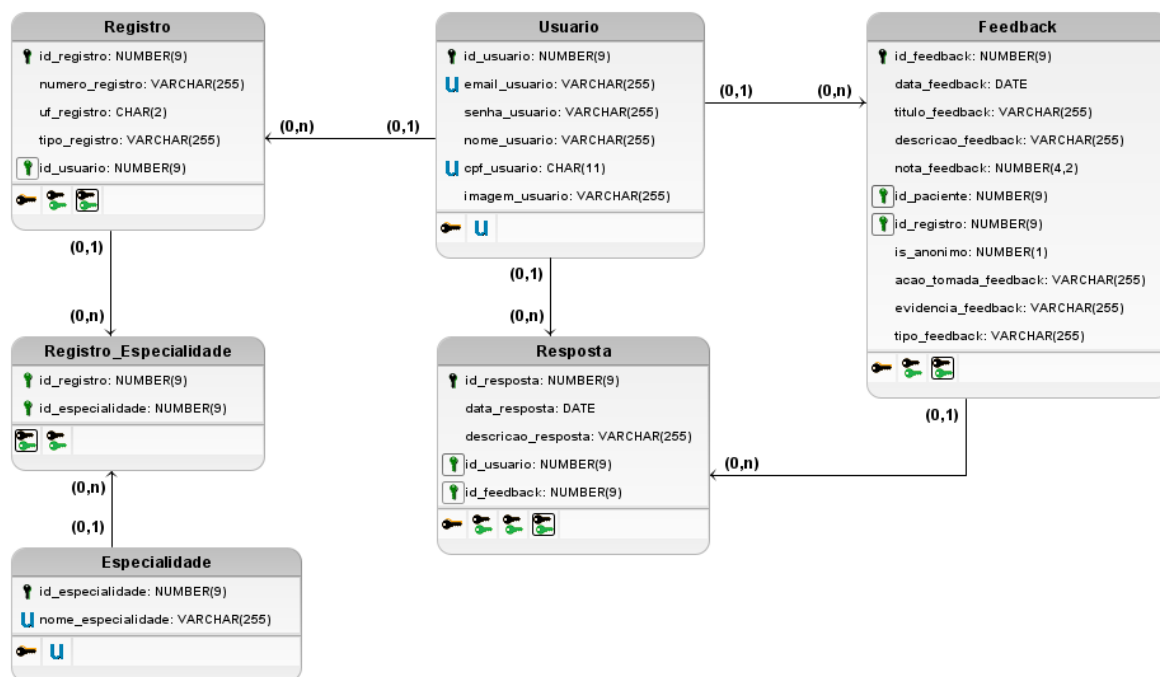
Observações: A diferença entre `id_registro` e `numero_registro` é essencial, pois o `numero_registro` pode ser o mesmo para diferentes médicos, variando de acordo com o tipo de registro (CRM, CRF, etc.) e a UF. Por exemplo, CRM-12345-SP pode ser do João, CRF-12345-SP pode ser do Natan, e CRM-12345-GO pode ser do Kaue.

## 2 – Projeto Lógico do Banco de Dados





## 3 – Projeto Físico do Banco de Dados



## 4 - Prints das evidências da criação dos objetos no Banco

### 4.1 Usuário

```
CREATE TABLE Usuario (
  id_usuario NUMBER(9) CONSTRAINT pk_id_usuario PRIMARY KEY,
  email_usuario VARCHAR(255) CONSTRAINT uk_email_usuario UNIQUE CONSTRAINT nn_email_usuario NOT NULL,
  senha_usuario VARCHAR(255) CONSTRAINT nn_senha_usuario NOT NULL,
  nome_usuario VARCHAR(255) CONSTRAINT nn_nome_usuario NOT NULL,
  cpf_usuario CHAR(11) CONSTRAINT uk_cpf_usuario UNIQUE CONSTRAINT nn_cpf_usuario NOT NULL,
  imagem_usuario VARCHAR(255)
);
```

Saída do Script x

Tarefa concluída em 0,039 segundos

Table USUARIO criado.

### 4.2 Registro

```
CREATE TABLE Registro (
  id_registro NUMBER(9) CONSTRAINT pk_id_registro PRIMARY KEY,
  numero_registro VARCHAR(255) CONSTRAINT nn_numero_registro NOT NULL,
  uf_registro CHAR(2) CONSTRAINT nn_uf_registro NOT NULL,
  tipo_registro VARCHAR(255) CONSTRAINT nn_tipo_registro NOT NULL,
  id_usuario NUMBER(9) CONSTRAINT fk_usuario_registro REFERENCES Usuario(id_usuario),
  CONSTRAINT uk_registro_unico UNIQUE (numero_registro, uf_registro, tipo_registro)
);
```

Saída do Script x

Tarefa concluída em 0,037 segundos

Table REGISTRO criado.

### 4.3 Feedback

```
CREATE TABLE Feedback (
  id_feedback NUMBER(9) CONSTRAINT pk_id_feedback PRIMARY KEY,
  data_feedback DATE CONSTRAINT nn_data_feedback NOT NULL,
  titulo_feedback VARCHAR(255) CONSTRAINT nn_titulo_feedback NOT NULL,
  descricao_feedback VARCHAR(255) CONSTRAINT nn_descricao_feedback NOT NULL,
  nota_feedback NUMBER(4,2) CONSTRAINT nn_nota_feedback NOT NULL,
  id_paciente NUMBER(9) CONSTRAINT fk_paciente_feedback REFERENCES Usuario(id_usuario),
  id_registro NUMBER(9) CONSTRAINT fk_registro_feedback REFERENCES Registro(id_registro),
  is_anonimo NUMBER(1) CONSTRAINT nn_is_anonimo_feedback NOT NULL,
  acao_tomada_feedback VARCHAR(255),
  evidencia_feedback VARCHAR(255),
  tipo_feedback VARCHAR(255) CONSTRAINT nn_tipo_feedback NOT NULL
);
```

Salida do Script x

Tarefa concluída em 0,04 segundos

Table FEEDBACK criado.

### 4.4 Especialidade

```
CREATE TABLE Especialidade (
  id_especialidade NUMBER(9) CONSTRAINT pk_id_especialidade PRIMARY KEY,
  nome_especialidade VARCHAR(255) CONSTRAINT uk_nome_especialidade UNIQUE CONSTRAINT nn_nome_especialidade NOT NULL
);
```

Salida do Script x

Tarefa concluída em 0,032 segundos

Table ESPECIALIDADE criado.

### 4.5 Registro\_Especialidade

```
CREATE TABLE Registro_Especialidade (
  id_registro NUMBER(9) CONSTRAINT fk_registro_especialidade REFERENCES Registro(id_registro),
  id_especialidade NUMBER(9) CONSTRAINT fk_especialidade_registro REFERENCES Especialidade(id_especialidade)
);
```

Salida do Script x

Tarefa concluída em 0,029 segundos

Table REGISTRO\_ESPECIALIDADE criado.

## 4.6 Resposta

```
CREATE TABLE Resposta (  
    id_resposta NUMBER(9) CONSTRAINT pk_id_resposta PRIMARY KEY,  
    data_resposta DATE CONSTRAINT nn_data_resposta NOT NULL,  
    descricao_resposta VARCHAR(255) CONSTRAINT nn_descricao_resposta NOT NULL,  
    id_usuario NUMBER(9) CONSTRAINT fk_resposta_usuario REFERENCES Usuario(id_usuario),  
    id_feedback NUMBER(9) CONSTRAINT fk_resposta_feedback REFERENCES Feedback(id_feedback)  
);
```

Saída do Script x

Tarefa concluída em 0,037 segundos

Table RESPOSTA criado.

## 5 - Prints das evidências da carga de dados

### 5.1 Usuario

```
INSERT INTO Usuario (id_usuario, email_usuario, senha_usuario, nome_usuario, cpf_usuario, imagem_usuario)
VALUES (v_id_usuario, v_email_usuario, v_senha_usuario, v_nome_usuario, v_cpf_usuario, v_imagem_usuario);

COMMIT;
DBMS_OUTPUT.PUT_LINE('Usu?rio inserido com sucesso. ID do Usu?rio: ' || v_id_usuario);

EXCEPTION
  WHEN OTHERS THEN
    DBMS_OUTPUT.PUT_LINE('Ocorreu um erro: ' || SQLERRM);
    ROLLBACK;
END;
Usu?rio inserido com sucesso. ID do Usu?rio: 21

Procedimento PL/SQL concluído com sucesso.
```

### 5.2 Registro

```
INSERT INTO Registro (id_registro, numero_registro, uf_registro, tipo_registro, id_usuario)
VALUES (v_id_registro, v_numero_registro, v_uf_registro, v_tipo_registro, v_id_usuario);

COMMIT;
DBMS_OUTPUT.PUT_LINE('Dados inseridos com sucesso. ID do Registro: ' || v_id_registro);

EXCEPTION
  WHEN OTHERS THEN
    DBMS_OUTPUT.PUT_LINE('Ocorreu um erro: ' || SQLERRM);
    ROLLBACK;
END;
Dados inseridos com sucesso. ID do Registro: 21

Procedimento PL/SQL concluído com sucesso.
```

### 5.3 Feedback

```
INSERT INTO Feedback (id_feedback, data_feedback, titulo_feedback, descricao_feedback, nota_feedback, id_paciente, id_registro, is_anonimo, acao_
VALUES (v_id_feedback, v_data_feedback, v_titulo_feedback, v_descricao_feedback, v_nota_feedback, v_id_paciente, v_id_registro, v_is_anonimo, v_a

COMMIT;
DBMS_OUTPUT.PUT_LINE('Feedback inserido com sucesso. ID do Feedback: ' || v_id_feedback);

EXCEPTION
  WHEN OTHERS THEN
    DBMS_OUTPUT.PUT_LINE('Ocorreu um erro: ' || SQLERRM);
    ROLLBACK;
END;
Feedback inserido com sucesso. ID do Feedback: 21

Procedimento PL/SQL concluído com sucesso.
```

#### 5.4 Especialidade

```
        COMMIT;
        DBMS_OUTPUT.PUT_LINE('Dados inseridos com sucesso. ID da Especialidade: ' || v_id_especialidade);

    EXCEPTION
        WHEN OTHERS THEN
            DBMS_OUTPUT.PUT_LINE('Ocorreu um erro: ' || SQLERRM);
            ROLLBACK;
    END;
Dados inseridos com sucesso. ID da Especialidade: 21

Procedimento PL/SQL concluído com sucesso.
```

#### 5.5 Registro\_Especialidade

```
        RAISE_APPLICATION_ERROR(-20007, 'Esta combina??o de Registro e Especialidade j? existe.');
```

```
    END IF;

    INSERT INTO Registro_Especialidade (id_registro, id_especialidade)
    VALUES (v_id_registro, v_id_especialidade);

    COMMIT;
    DBMS_OUTPUT.PUT_LINE('Associa??o inserida com sucesso.');
```

```
    EXCEPTION
        WHEN OTHERS THEN
            DBMS_OUTPUT.PUT_LINE('Ocorreu um erro: ' || SQLERRM);
            ROLLBACK;
    END;
Associa??o inserida com sucesso.

Procedimento PL/SQL concluído com sucesso.
```

#### 5.6 Resposta

```
    INSERT INTO Resposta (id_resposta, data_resposta, descricao_resposta, id_usuario, id_feedback)
    VALUES (v_id_resposta, v_data_resposta, v_descricao_resposta, v_id_usuario, v_id_feedback);

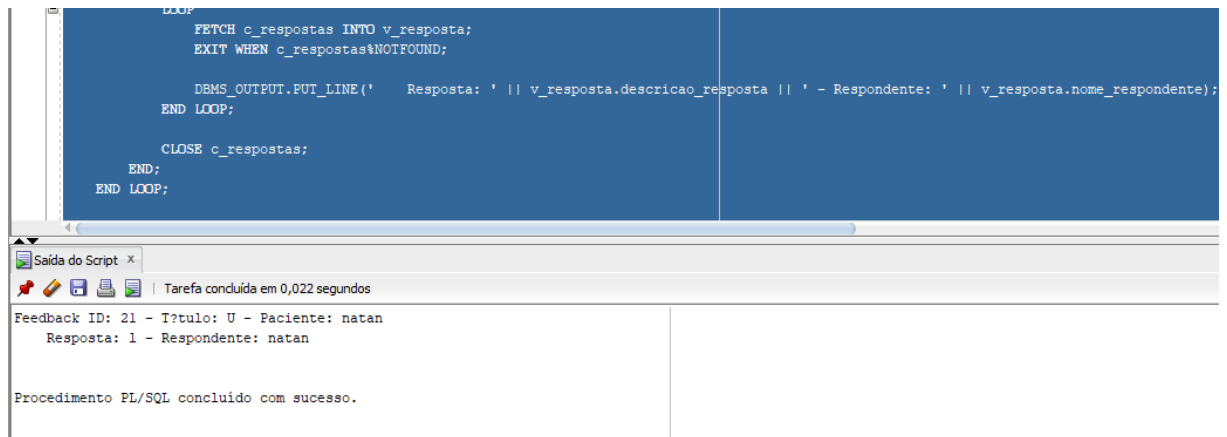
    COMMIT;
    DBMS_OUTPUT.PUT_LINE('Resposta inserida com sucesso. ID da Resposta: ' || v_id_resposta);

    EXCEPTION
        WHEN OTHERS THEN
            DBMS_OUTPUT.PUT_LINE('Ocorreu um erro: ' || SQLERRM);
            ROLLBACK;
    END;
Resposta inserida com sucesso. ID da Resposta: 21

Procedimento PL/SQL concluído com sucesso.
```

## 6 - Prints das evidências da execução das duas pesquisas

### 6.1 Relatório de Feedbacks e suas Respostas Correspondentes



The screenshot displays a SQL Developer window with a PL/SQL script in the editor. The script iterates through a cursor named `c_respostas`, fetching data into `v_resposta` and printing it using `DBMS_OUTPUT.PUT_LINE`. The output format is: `Resposta: ' || v_resposta.descricao_resposta || ' - Respondente: ' || v_resposta.nome_respondente);`. Below the editor, the 'Saída do Script' (Script Output) window shows the execution results.

```
LOOP
    FETCH c_respostas INTO v_resposta;
    EXIT WHEN c_respostas%NOTFOUND;

    DBMS_OUTPUT.PUT_LINE('    Resposta: ' || v_resposta.descricao_resposta || ' - Respondente: ' || v_resposta.nome_respondente);
END LOOP;

CLOSE c_respostas;
END;
END LOOP;
```

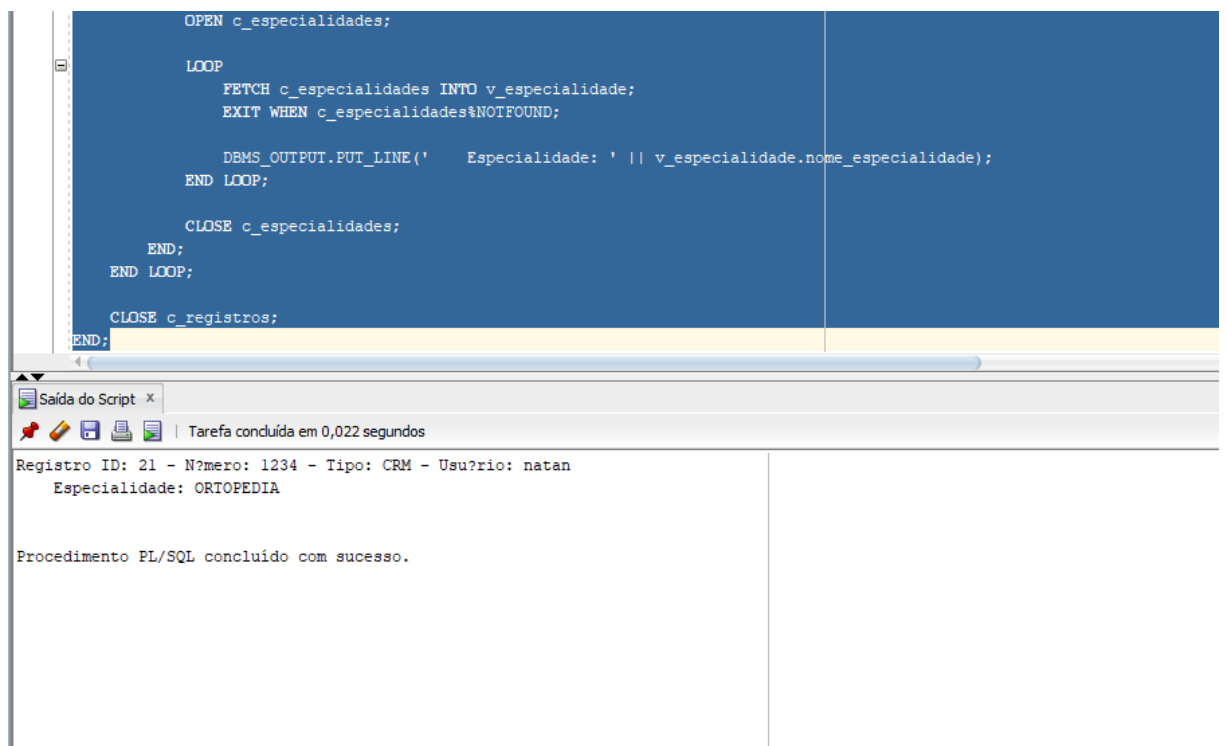
Saída do Script x

Tarefa concluída em 0,022 segundos

Feedback ID: 21 - Título: U - Paciente: natan  
Resposta: 1 - Respondente: natan

Procedimento PL/SQL concluído com sucesso.

### 6.2 Relatório de Registros e Especialidades Associadas



The screenshot displays a SQL Developer window with a PL/SQL script in the editor. The script opens a cursor `c_especialidades`, iterates through it, fetching data into `v_especialidade` and printing it using `DBMS_OUTPUT.PUT_LINE`. The output format is: `Especialidade: ' || v_especialidade.nome_especialidade);`. Below the editor, the 'Saída do Script' (Script Output) window shows the execution results.

```
OPEN c_especialidades;

LOOP
    FETCH c_especialidades INTO v_especialidade;
    EXIT WHEN c_especialidades%NOTFOUND;

    DBMS_OUTPUT.PUT_LINE('    Especialidade: ' || v_especialidade.nome_especialidade);
END LOOP;

CLOSE c_especialidades;
END;
END LOOP;

CLOSE c_registros;
END;
```

Saída do Script x

Tarefa concluída em 0,022 segundos

Registro ID: 21 - Número: 1234 - Tipo: CRM - Usuário: natan  
Especialidade: ORTOPEDIA

Procedimento PL/SQL concluído com sucesso.