**Centro Universitário UNISATC**

Engenharia de Software 3a fase – Banco de Dados II – Prof.Jorge Luiz da Silva

**TRABALHO FINAL COM BASE EM METODOLOGIAS ATIVAS DE APRENDIZAGEM**

**Projeto de banco de dados para um sistema de Atendimento Fisioterapêutico**

Gabriel Tarciso Macieiski + GTM16

Jefferson Barzan Alexandrino + JeffAlexandrino

João Eduardo Milak Farias + conta github

João Vittor Gomes Fernandes + conta github

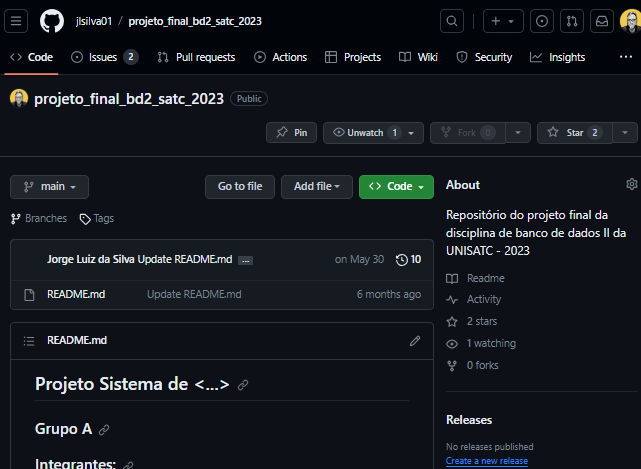
Maria Eduarda Monteiro Marcos + Guna-ME

Maria Luíza Arend da Silva + conta github

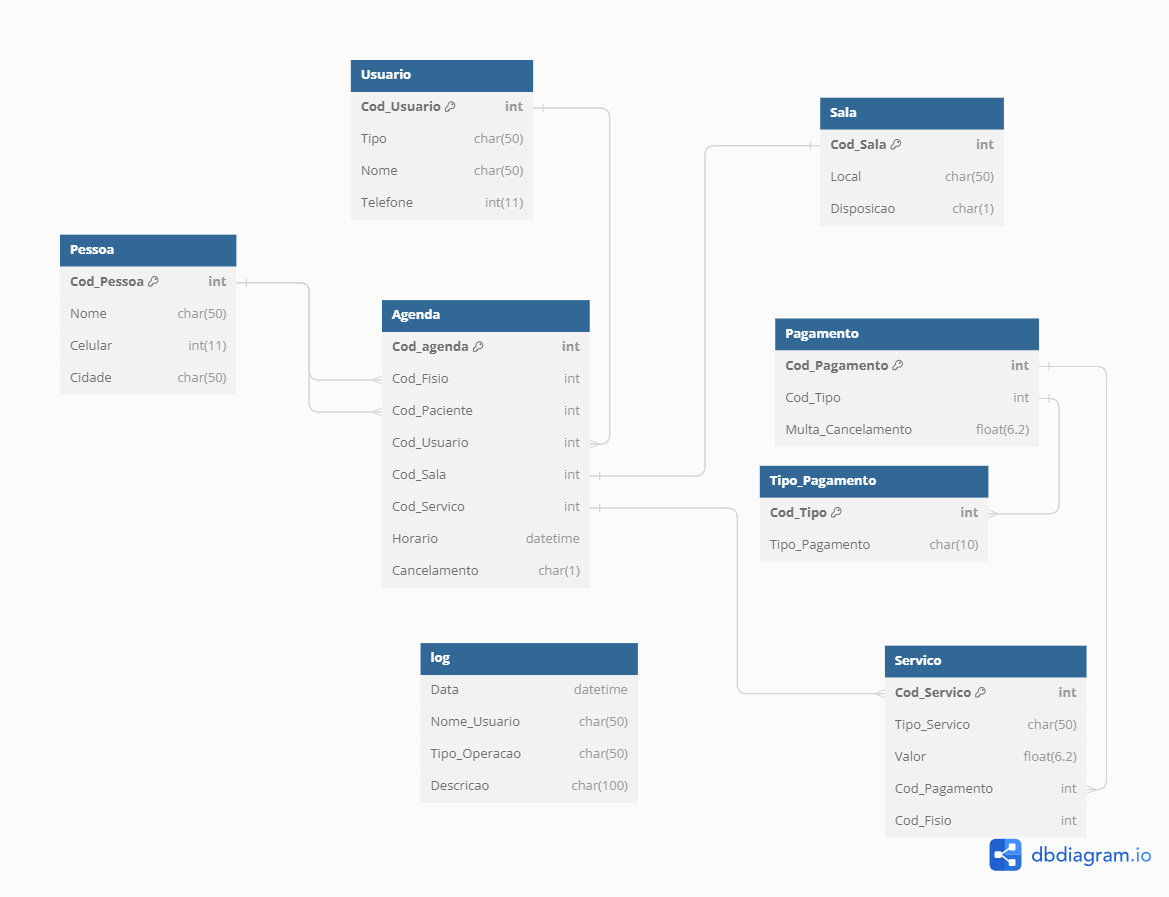
Criciúma, 13/11/2023

**URL do projeto no GitHub**

<https://github.com/Guna-ME/FisioRangers.git>

****

**Modelo ER Físico**

****

**Dicionário de Dados**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Tabela** | Avaliação | | | | | | |
| **Descrição** | Tabela responsável por armazenar os dados das avaliações criadas pelos professores | | | | | | |
| **Atributos** | | | | | | | |
| **Nome da Coluna** | | **Tipo do Dado** | **Valor min e max** | **Nulidade** | **PK** | **FK** | **Descrição** |
| cd\_avaliacao | | int | 1 – sem limite | NOT NULL | X |  | Código de identificador da avaliação |
| nm\_avaliacao | | varchar(200) | 10 - 200 | NOT NULL |  |  | Nome da avaliação |
|  | |  |  |  |  |  |  |
|  | |  |  |  |  |  |  |
|  | |  |  |  |  |  |  |
|  | |  |  |  |  |  |  |
| **Índice** | | | | | | | |
| **Nome do índice** | | **Clustered** | **NonClustered** | **Unique** | **Colunas** | | |
| Idx\_dt\_avaliacao | | X | X | X | cd\_avaliacao, dt\_avaliacao | | |
| Idx\_tipo | |  | X |  | cd\_avaliacao, dt\_avaliacao | | |

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Tabela** | Avaliação | | | | | | |
| **Descrição** | Tabela responsável por armazenar os dados das avaliações criadas pelos professores | | | | | | |
| **Atributos** | | | | | | | |
| **Nome da Coluna** | | **Tipo do Dado** | **Valor min e max** | **Nulidade** | **PK** | **FK** | **Descrição** |
| cd\_avaliacao | | int | 1 – sem limite | NOT NULL | X |  | Código de identificador da avaliação |
| nm\_avaliacao | | varchar(200) | 10 - 200 | NOT NULL |  |  | Nome da avaliação |
|  | |  |  |  |  |  |  |
|  | |  |  |  |  |  |  |
|  | |  |  |  |  |  |  |
|  | |  |  |  |  |  |  |
| **Índice** | | | | | | | |
| **Nome do índice** | | **Clustered** | **NonClustered** | **Unique** | **Colunas** | | |
| Idx\_dt\_avaliacao | | X | X | X | cd\_avaliacao, dt\_avaliacao | | |
| Idx\_tipo | |  | X |  | cd\_avaliacao, dt\_avaliacao | | |

**Script dos comandos DDL para criação do Banco de dados**

-- Create Table: Pessoa

CREATE TABLE [Pessoa] (

[Cod\_Pessoa] int PRIMARY KEY,

[Nome] char(50),

[Celular] int(11),

[Cidade] char(50)

)

GO

-- Create Table: Usuario

CREATE TABLE [Usuario] (

[Cod\_Usuario] int PRIMARY KEY,

[Tipo] char(50),

[Nome] char(50),

[Telefone] int(11)

)

GO

-- Create Table: Sala

CREATE TABLE [Sala] (

[Cod\_Sala] int PRIMARY KEY,

[Local] char(50),

[Disposicao] char(1)

)

GO

-- Create Table: Agenda

CREATE TABLE [Agenda] (

[Cod\_agenda] int PRIMARY KEY,

[Cod\_Fisio] int,

[Cod\_Paciente] int,

[Cod\_Usuario] int,

[Cod\_Sala] int,

[Cod\_Servico] int,

[Horario] datetime,

[Cancelamento] char(1)

)

GO

-- Create Table: Servico

CREATE TABLE [Servico] (

[Cod\_Servico] int PRIMARY KEY,

[Tipo\_Servico] char(50),

[Valor] float(6.2),

[Cod\_Pagamento] int,

[Cod\_Fisio] int

)

GO

-- Create Table: Pagamento

CREATE TABLE [Pagamento] (

[Cod\_Pagamento] int PRIMARY KEY,

[Cod\_Tipo] int,

[Multa\_Cancelamento] float(6.2)

)

GO

-- Create Table: Tipo\_Pagamento

CREATE TABLE [Tipo\_Pagamento] (

[Cod\_Tipo] int PRIMARY KEY,

[Tipo\_Pagamento] char(10)

)

GO

-- Create Table: log

CREATE TABLE [log] (

[Data] datetime,

[Nome\_Usuario] char(50),

[Tipo\_Operacao] char(50),

[Descricao] char(100)

)

GO

ALTER TABLE [Agenda] ADD FOREIGN KEY ([Cod\_Paciente]) REFERENCES [Pessoa] ([Cod\_Pessoa])

GO

ALTER TABLE [Servico] ADD FOREIGN KEY ([Cod\_Pagamento]) REFERENCES [Pagamento] ([Cod\_Pagamento])

GO

ALTER TABLE [Agenda] ADD FOREIGN KEY ([Cod\_Usuario]) REFERENCES [Usuario] ([Cod\_Usuario])

GO

ALTER TABLE [Agenda] ADD FOREIGN KEY ([Cod\_Sala]) REFERENCES [Sala] ([Cod\_Sala])

GO

ALTER TABLE [Agenda] ADD FOREIGN KEY ([Cod\_Fisio]) REFERENCES [Pessoa] ([Cod\_Pessoa])

GO

ALTER TABLE [Servico] ADD FOREIGN KEY ([Cod\_Servico]) REFERENCES [Agenda] ([Cod\_Servico])

GO

ALTER TABLE [Tipo\_Pagamento] ADD FOREIGN KEY ([Cod\_Tipo]) REFERENCES [Pagamento] ([Cod\_Tipo])

GO

**Script que popula as tabelas do Banco de dados**

/\* ADICIONA ALUNOS \*/

--INSERT INTO aluno (nm\_aluno, email)

-- VALUES ('FULANO DE TAL', 'fulano@satc.edu.br');

--GO

insert into aluno (nm\_aluno, email)

select 'MARISA MELO OLIVEIRA', 'email@satc.edu.br' ;

insert into aluno (nm\_aluno, email)

select 'MURILO CARVALHO CARDOSO', 'email@satc.edu.br' ;

insert into aluno (nm\_aluno, email)

select 'VINICIUS ROCHA RODRIGUES', 'email@satc.edu.br' ;

/\* ADICIONA AS ALTERNATIVAS DE CADA QUESTAO\*/

INSERT INTO questao\_item (cd\_questao, ds\_questao\_item, is\_correta) VALUES (1, 'Dados / Comunicação / Conhecimento' ,0);

INSERT INTO questao\_item (cd\_questao, ds\_questao\_item, is\_correta) VALUES (1, 'Análise / Dados / Conhecimento' ,0);

INSERT INTO questao\_item (cd\_questao, ds\_questao\_item, is\_correta) VALUES (1, 'Informação / Compreensão / Comunicação' ,0);

INSERT INTO questao\_item (cd\_questao, ds\_questao\_item, is\_correta) VALUES (1, 'Dados / Informação / Conhecimento' ,1);

GO

INSERT INTO questao\_item (cd\_questao, ds\_questao\_item, is\_correta) VALUES (2, 'F, V, F, F, V', 1);

INSERT INTO questao\_item (cd\_questao, ds\_questao\_item, is\_correta) VALUES (2, 'V, F, F, V, F', 0 );

INSERT INTO questao\_item (cd\_questao, ds\_questao\_item, is\_correta) VALUES (2, 'V, V, V, F, V', 0);

INSERT INTO questao\_item (cd\_questao, ds\_questao\_item, is\_correta) VALUES (2, 'F, F, V, V, F', 0);

GO

**Principais consultas mapeadas baseadas em regras de negócio (mínimo 4)**

--13. (Aberta) Utilizando a linguagem SQL, execute consultas que possam responder a seguinte questão: Qual(is) Carro(s) possui(em) mais sinistros cadastrados?

select s.placa, COUNT(\*)

from sinistro s inner join carro c

on s.placa = c.placa

group by s.placa

order by COUNT(\*) desc

--14. (Aberta) Quantos clientes cadastrados possuem letra “a” no meio e não no final?

select \* --count(\*)

from cliente

where nome like '%a%'

and nome not like '%a'

--15. (Aberta) Quantos clientes não possuem apólice?

select nome, cod\_apolice --\* --count(\*)

from cliente c left join apolice a

on c.cod\_cliente = a.cod\_cliente

where cod\_apolice is null

select nome, cod\_apolice

from cliente c inner join apolice a on c.cod\_cliente = a.cod\_cliente

--16. (Aberta) Quais carros não possuem apólice?

select \* --count(\*)

from carro c left join apolice a on c.placa = a.placa

where cod\_apolice is null

select \*

from apolice

where placa not in (select placa from carro)