**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 Gerenciamento de Estacionamento**

Realdo Justino Junior - https://github.com/Realdo-Justino

Paulo Ronchi Francisconi- https://github.com/pauloFrancisconi

Aluno 3 - https://github.com/rhyanorben

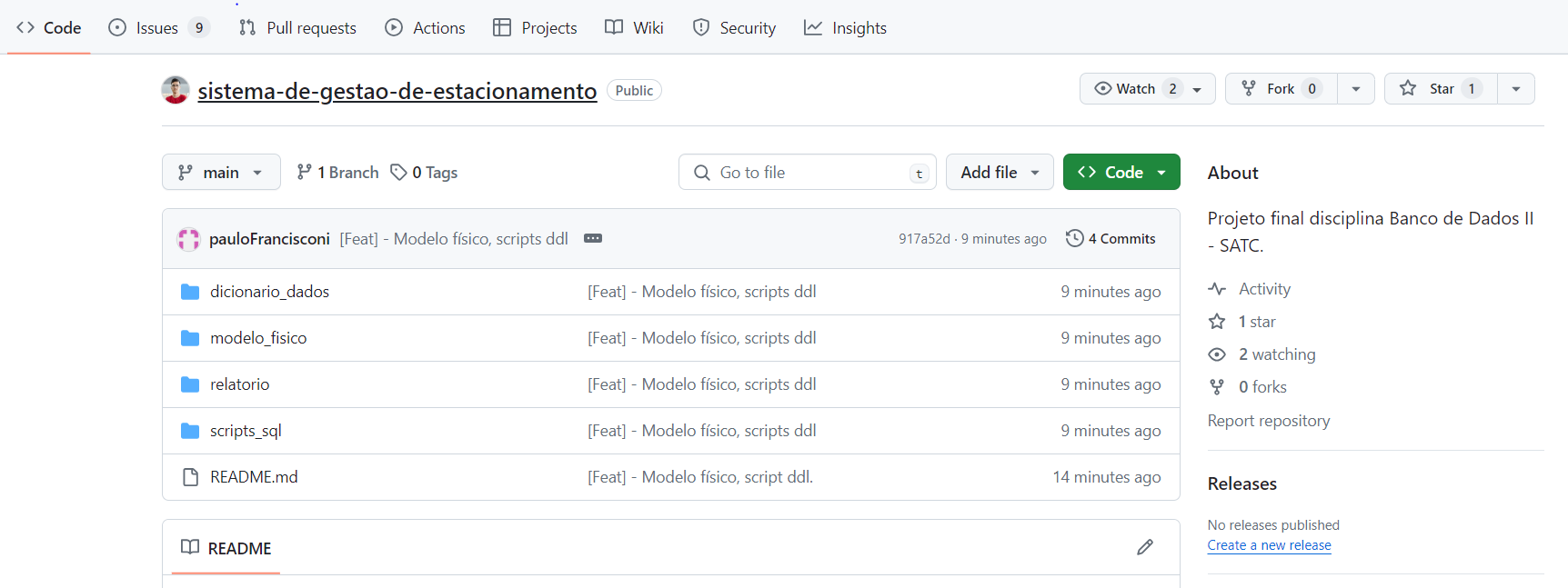
Aluno 4 - https://github.com/igrocha

Aluno 5 - https://github.com/Joao-AugustoPF

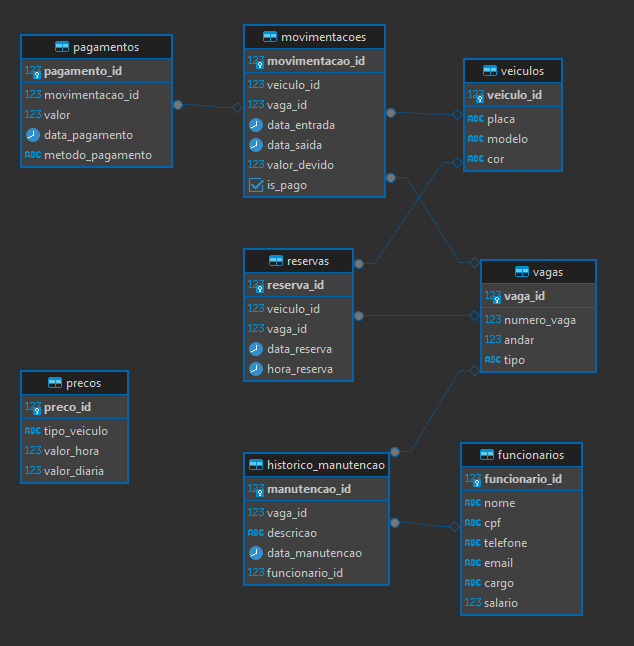
Criciúma, 09/06/2024

**URL do projeto no GitHub**

<https://github.com/Joao-AugustoPF/sistema-de-gestao-de-estacionamento>



**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**

-- Tabela Veículos

CREATE TABLE Veiculos (

veiculo\_id SERIAL PRIMARY KEY,

placa VARCHAR(10) NOT NULL,

modelo VARCHAR(50),

cor VARCHAR(20)

);

-- Tabela Vagas

CREATE TABLE VAGAS (

vaga\_id SERIAL PRIMARY KEY,

numero\_vaga INTEGER NOT NULL,

andar INTEGER,

tipo VARCHAR(20)

);

-- Tabela Movimentações

CREATE TABLE Movimentacoes (

movimentacao\_id SERIAL PRIMARY KEY,

veiculo\_id INTEGER REFERENCES Veiculos(veiculo\_id),

vaga\_id INTEGER REFERENCES Vagas(vaga\_id),

data\_entrada TIMESTAMP NOT NULL,

data\_saida TIMESTAMP,

valor\_devido DECIMAL(10, 2),

is\_pago BOOLEAN

);

Tabela Funcionários

CREATE TABLE Funcionarios (

funcionario\_id SERIAL PRIMARY KEY,

nome VARCHAR(100) NOT NULL,

cpf VARCHAR(14),

telefone VARCHAR(15),

email VARCHAR(100),

cargo VARCHAR(50),

salario DECIMAL(10, 2)

);

-- Tabela Pagamentos

CREATE TABLE Pagamentos (

pagamento\_id SERIAL PRIMARY KEY,

movimentacao\_id INTEGER REFERENCES Movimentacoes(movimentacao\_id),

valor DECIMAL(10, 2) NOT NULL,

data\_pagamento TIMESTAMP NOT NULL,

metodo\_pagamento VARCHAR(50)

);

-- Tabela Preços

CREATE TABLE Precos (

preco\_id SERIAL PRIMARY KEY,

tipo\_veiculo VARCHAR(20) NOT NULL,

valor\_hora DECIMAL(10, 2) NOT NULL,

valor\_diaria DECIMAL(10, 2) NOT NULL

);

-- Tabela Reservas

CREATE TABLE Reservas (

reserva\_id SERIAL PRIMARY KEY,

veiculo\_id INTEGER REFERENCES Veiculos(veiculo\_id),

vaga\_id INTEGER REFERENCES Vagas(vaga\_id),

data\_reserva DATE NOT NULL,

hora\_reserva TIME NOT NULL

);

-- Tabela Histórico de Manutenção

CREATE TABLE Historico\_Manutencao (

manutencao\_id SERIAL PRIMARY KEY,

vaga\_id INTEGER REFERENCES Vagas(vaga\_id),

descricao TEXT,

data\_manutencao DATE NOT NULL,

funcionario\_id INTEGER REFERENCES Funcionarios(funcionario\_id)

);

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)