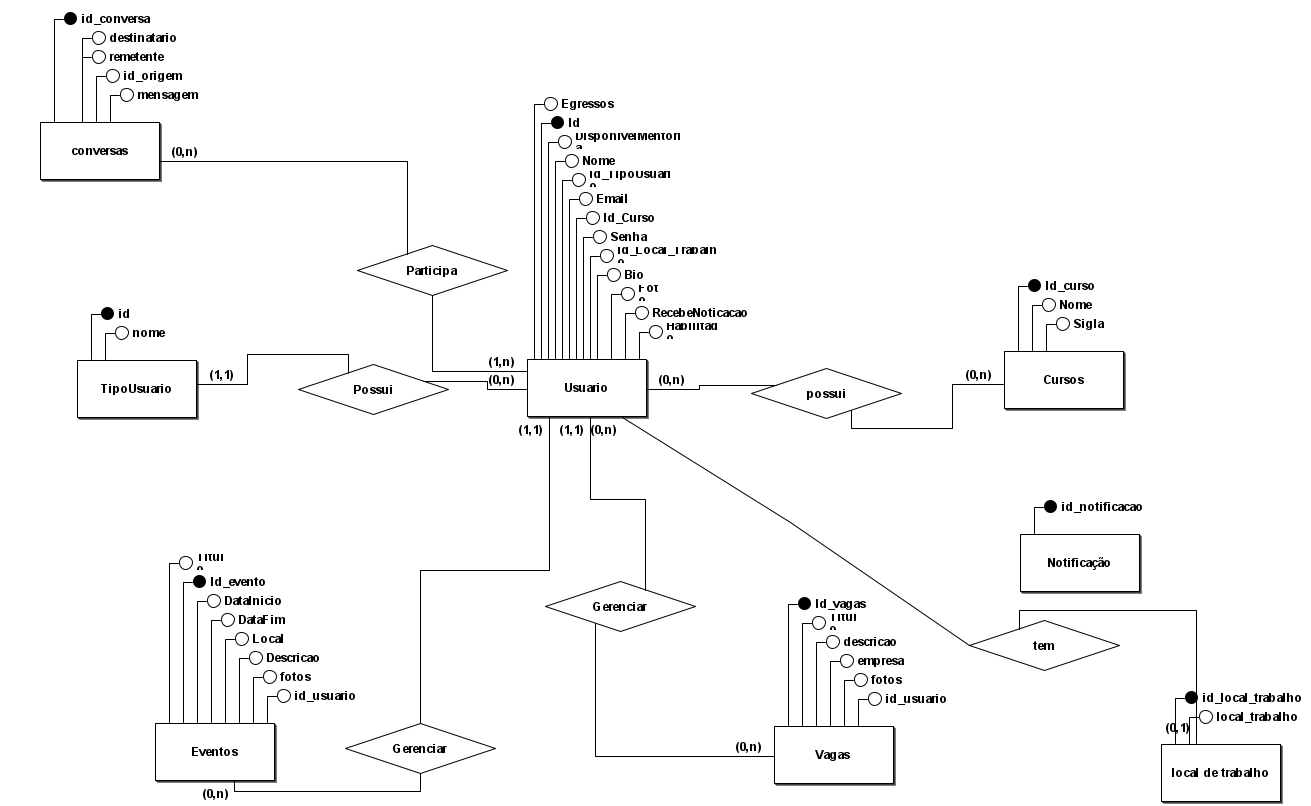
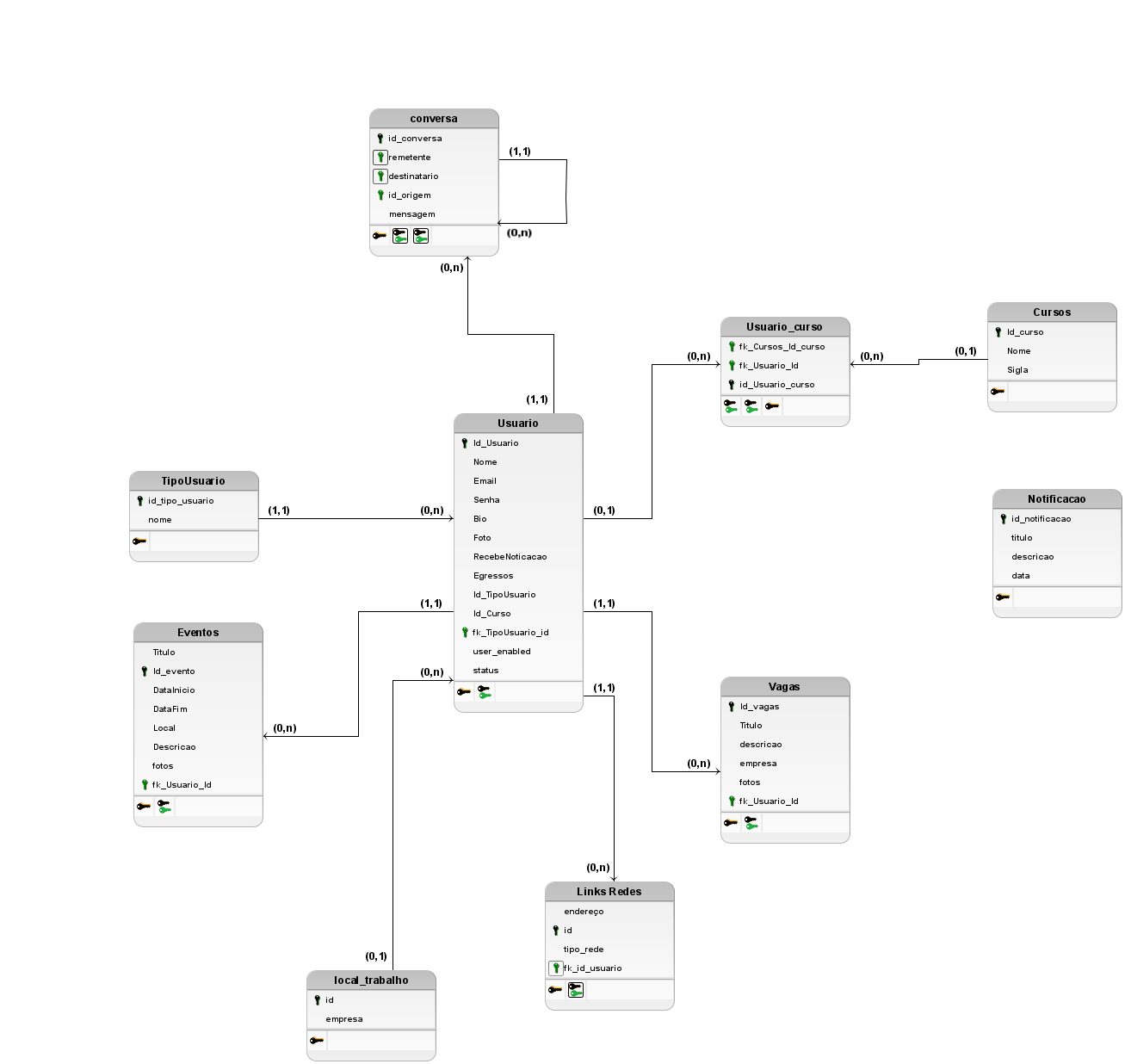
Integrantes:  
Gabriel Bellato   
Leonardo Barbosa da Silva   
Nicolas Alexandrino Ferro  
  
Descrição

A proposta deste projeto é o desenvolvimento de uma plataforma web em que alunos egressos, recém-formados e a Fatec Sorocaba possam criar uma rede de networking e conexões, aumentando a integração e o sentimento de pertencimento entre os egressos, alunos e a faculdade. A conexão e o contato com egressos seriam permanentes no ambiente da Fatec, proporcionando o acompanhamento dos alunos após sua formação. A Fatec teria uma forma de saber como estão seus alunos no mercado e, com isso, até os outros alunos podem se inspirar em suas trajetórias.

O sistema foi modelado para atender diferentes tipos de usuários, como visitantes, empresas parceiras, alunos, professores, administradores e o próprio sistema. Visitantes podem realizar cadastro e login, com possibilidade de recuperação de senha. Após autenticados, os usuários têm acesso a funcionalidades como consultar eventos, editar perfil, enviar mensagens, gerenciar vagas e fazer logout. Além disso, é possível buscar perfis, ação que também está disponível para visitantes e empresas parceiras.

Administradores possuem privilégios avançados, podendo aprovar ou recusar cadastros — ação que inclui o envio da aprovação correspondente —, além de gerenciar relatórios e eventos. Professores também têm acesso à funcionalidade de gerenciamento de eventos, promovendo o envolvimento institucional. Por fim, o sistema contempla o envio automático de notificações, garantindo uma comunicação ativa entre os participantes da plataforma.

--esta tabela é apenas para alimentar um comboBox dinamicamente solicitado pela professora denilce;

-- tem como objetivo gerar relatorios para a função admin da plataforma para realizar o acompanhamento

-- dos alunos;

CREATE TABLE local\_trabalho(

id\_local\_trabalho NUMBER GENERATED ALWAYS AS IDENTITY,

empresa varchar(50) NOT NULL

);

ALTER TABLE local\_trabalho ADD CONSTRAINT PK\_local\_trabalho

PRIMARY KEY(id\_local\_trabalho);

-----------------------------------------------------

CREATE TABLE TipoUsuario (

id\_tipo\_usuario NUMBER GENERATED ALWAYS AS IDENTITY,

nome varchar2(50) NOT NULL

);

ALTER TABLE TipoUsuario ADD CONSTRAINT PK\_TipoUsuario

PRIMARY KEY(id\_tipo\_usuario);

------------------------------------------------------

CREATE TABLE Usuario (

Id\_Usuario NUMBER GENERATED ALWAYS AS IDENTITY,

Nome VARCHAR2(80) NOT NULL,

Email VARCHAR2(50) NOT NULL,

Senha VARCHAR2(20) NOT NULL,

Bio VARCHAR2(200),

Foto VARCHAR2(255),

RecebeNotificacao SMALLINT DEFAULT 0 NOT NULL, -- sempre começar o checkBox no front como false

Egressos SMALLINT DEFAULT 0 NOT NULL, -- sempre começar o checkBox no front como false

Id\_TipoUsuario NUMBER NOT NULL,

User\_Enabled SMALLINT DEFAULT 0 NOT NULL, -- se usuario está aprovado na plataforma

status\_usuario VARCHAR2(20) NOT NULL, -- definido como enum no diagrama de classes

id\_LocalTrabalho NUMBER

);

ALTER TABLE Usuario ADD CONSTRAINT PK\_Id\_Usuario

PRIMARY KEY(Id\_Usuario);

ALTER TABLE Usuario ADD CONSTRAINT FK\_Usuario\_TipoUsuario

FOREIGN KEY (Id\_TipoUsuario)

REFERENCES TipoUsuario (id\_tipo\_usuario);

ALTER TABLE USUARIO ADD CONSTRAINT FK\_Usuario\_LocalTrabalho

FOREIGN KEY (id\_LocalTrabalho)

REFERENCES LOCAL\_TRABALHO (ID\_LOCAL\_TRABALHO);

---------------------------------

CREATE TABLE Eventos (

Id\_evento NUMBER GENERATED ALWAYS AS IDENTITY,

Titulo VARCHAR2(100) NOT NULL,

DataInicio DATE NOT NULL,

DataFim DATE NOT NULL,

Local\_evento VARCHAR2(150),

Descricao VARCHAR2(1000),

Fotos VARCHAR2(255),

fk\_Usuario\_Id NUMBER NOT NULL

);

ALTER TABLE Eventos ADD CONSTRAINT FK\_Eventos\_Usuario

FOREIGN KEY (fk\_Usuario\_Id)

REFERENCES Usuario (Id\_Usuario);

ALTER TABLE Eventos ADD CONSTRAINT PK\_Id\_evento

PRIMARY KEY(Id\_evento);

----------------------------------------------------------

CREATE TABLE Vagas (

Id\_vagas NUMBER GENERATED ALWAYS AS IDENTITY,

Titulo VARCHAR2(100) NOT NULL,

Descricao VARCHAR2(1000),

Empresa VARCHAR2(100),

Fotos VARCHAR2(255),

fk\_Usuario\_Id NUMBER NOT NULL

);

ALTER TABLE Vagas ADD CONSTRAINT FK\_Vagas\_Usuario

FOREIGN KEY (fk\_Usuario\_Id)

REFERENCES Usuario (Id\_Usuario);

----------------------------------------------------------

CREATE TABLE LinksRedes (

id\_link NUMBER GENERATED ALWAYS AS IDENTITY,

endereco VARCHAR2(255) NOT NULL,

tipo\_rede VARCHAR2(50) NOT NULL,--definido como enum no diagrama de classes

fk\_id\_usuario NUMBER NOT NULL

);

ALTER TABLE LinksRedes ADD CONSTRAINT PK\_LinksRedes

PRIMARY KEY(id\_link);

ALTER TABLE LinksRedes ADD CONSTRAINT FK\_LinksRedes\_Usuario

FOREIGN KEY (fk\_id\_usuario)

REFERENCES Usuario (Id\_Usuario);

-----------------------------------------------------------------------

CREATE TABLE Notificacao (

id\_notificacao NUMBER GENERATED ALWAYS AS IDENTITY,

titulo VARCHAR2(50) NOT NULL,

descricao VARCHAR2(100),

data\_envio DATE

);

ALTER TABLE Notificacao ADD CONSTRAINT PK\_notificacao

PRIMARY KEY(id\_notificacao);

--------------------------------------------------------------------------

CREATE TABLE Cursos (

Id\_curso NUMBER GENERATED ALWAYS AS IDENTITY,

Nome VARCHAR2 (255),

Sigla VARCHAR2(10)

);

ALTER TABLE Cursos ADD CONSTRAINT PK\_Cursos

PRIMARY KEY(Id\_curso);

----------------------------------------------------------------------------

CREATE TABLE Usuario\_curso (

fk\_Cursos\_ID NUMBER,

fk\_Usuario\_Id NUMBER,

id\_Usuario\_curso NUMBER GENERATED ALWAYS AS IDENTITY

);

ALTER TABLE Usuario\_curso ADD CONSTRAINT PK\_Usuario\_curso

PRIMARY KEY(id\_Usuario\_curso);

ALTER TABLE Usuario\_curso ADD CONSTRAINT FK\_Usuario\_curso\_CURSO

FOREIGN KEY (fk\_Cursos\_ID)

REFERENCES Cursos (Id\_curso);

ALTER TABLE Usuario\_curso ADD CONSTRAINT FK\_Usuario\_curso\_USUARIO

FOREIGN KEY (fk\_Usuario\_Id)

REFERENCES Usuario (Id\_Usuario);

-------------------------------------------------------------------------------

CREATE TABLE conversa (

id\_mensagem NUMBER GENERATED ALWAYS AS IDENTITY,

remetente NUMBER NOT NULL,

destinatario NUMBER NOT NULL,

id\_origem NUMBER,

mensagem VARCHAR2(500)

);

ALTER TABLE conversa ADD CONSTRAINT PK\_mensagem

PRIMARY KEY(id\_mensagem);

ALTER TABLE conversa ADD CONSTRAINT FK\_remetente

FOREIGN KEY (remetente)

REFERENCES Usuario (Id\_Usuario);

ALTER TABLE conversa ADD CONSTRAINT FK\_destinatario

FOREIGN KEY (destinatario)

REFERENCES Usuario (Id\_Usuario);

alter table conversa ADD CONSTRAINT FK\_id\_origem --auto relacionamento de mensagens

FOREIGN KEY (id\_origem)

REFERENCES conversa (id\_mensagem);

-- questÃ£o 2 letra C

INSERT INTO TipoUsuario (nome) VALUES ('Aluno');

INSERT INTO TipoUsuario (nome) VALUES ('Administrador');

INSERT INTO TipoUsuario (nome) VALUES ('Professor');

---------

INSERT INTO local\_trabalho (empresa)

VALUES ('InfoTech');

INSERT INTO local\_trabalho (empresa)

VALUES ('SoftSolutions');

INSERT INTO local\_trabalho (empresa)

VALUES ('DataCorp');

INSERT INTO local\_trabalho (empresa)

VALUES ('WebDesigners');

INSERT INTO local\_trabalho (empresa)

VALUES ('BackSolutions');

INSERT INTO local\_trabalho (empresa)

VALUES ('Google Inc.');

-------

INSERT INTO Usuario (Nome, Email, Senha, Bio, Foto, RecebeNotificacao, Egressos, Id\_TipoUsuario, User\_Enabled, status\_usuario, id\_LocalTrabalho)

VALUES ('Carlos Silva', 'carlos@email.com', 'senha123', 'Aluno do 3º semestre', NULL, 1, 1, 1, 1, 'ativo',1);

INSERT INTO Usuario (Nome, Email, Senha, Bio, Foto, RecebeNotificacao, Egressos, Id\_TipoUsuario, User\_Enabled, status\_usuario, id\_LocalTrabalho)

VALUES ('Ana Lima', 'ana@email.com', 'senha456', 'Professora de banco de dados', NULL, 1, 1, 3, 1, 'ativo',2);

INSERT INTO Usuario (Nome, Email, Senha, Bio, Foto, RecebeNotificacao, Egressos, Id\_TipoUsuario, User\_Enabled, status\_usuario, id\_LocalTrabalho)

VALUES ('João Souza', 'joao@email.com', 'senha789', 'Administrador do sistema', NULL, 0, 0, 2, 1, 'ativo',3);

INSERT INTO Usuario (Nome, Email, Senha, Bio, Foto, RecebeNotificacao, Egressos, Id\_TipoUsuario, User\_Enabled, status\_usuario, id\_LocalTrabalho)

VALUES ('Maria Fernanda', 'maria@email.com', 'senhaabc', 'Aluna de ADS', NULL, 1, 0, 1, 0, 'pendente',4);

INSERT INTO Usuario (Nome, Email, Senha, Bio, Foto, RecebeNotificacao, Egressos, Id\_TipoUsuario, User\_Enabled, status\_usuario, id\_LocalTrabalho)

VALUES ('Ricardo Costa', 'ricardo@email.com', 'senhaxyz', 'Professor de Java', NULL, 1, 1, 3, 1, 'ativo',4);

------

INSERT INTO Eventos (Titulo, DataInicio, DataFim, Local\_evento, Descricao, Fotos, fk\_Usuario\_Id)

VALUES ('Semana de Tecnologia', DATE '2025-06-10', DATE '2025-06-14', 'Auditório 1', 'Evento anual com palestras e workshops', NULL, 2);

INSERT INTO Eventos (Titulo, DataInicio, DataFim, Local\_evento, Descricao, Fotos, fk\_Usuario\_Id)

VALUES ('Oficina de Spring Boot', DATE '2025-07-01', DATE '2025-07-01', 'Lab 3', 'Introdução ao Spring Boot', NULL, 5);

INSERT INTO Eventos (Titulo, DataInicio, DataFim, Local\_evento, Descricao, Fotos, fk\_Usuario\_Id)

VALUES ('Encontro de Egressos', DATE '2025-08-05', DATE '2025-08-05', 'Salão Nobre', 'Conexão entre ex-alunos', NULL, 3);

INSERT INTO Eventos (Titulo, DataInicio, DataFim, Local\_evento, Descricao, Fotos, fk\_Usuario\_Id)

VALUES ('Palestra de Carreira', DATE '2025-09-01', DATE '2025-09-01', 'Auditório 2', 'Dicas para o mercado de trabalho', NULL, 1);

INSERT INTO Eventos (Titulo, DataInicio, DataFim, Local\_evento, Descricao, Fotos, fk\_Usuario\_Id)

VALUES ('Feira de Estágios', DATE '2025-10-20', DATE '2025-10-21', 'Campus Central', 'Conexão com empresas', NULL, 4);

---------------

INSERT INTO Vagas (Titulo, Descricao, Empresa, Fotos, fk\_Usuario\_Id)

VALUES ('Estágio em TI', 'Suporte técnico e manutenção', 'InfoTech', NULL, 2);

INSERT INTO Vagas (Titulo, Descricao, Empresa, Fotos, fk\_Usuario\_Id)

VALUES ('Desenvolvedor Java Junior', 'Manutenção e criação de APIs', 'SoftSolutions', NULL, 5);

INSERT INTO Vagas (Titulo, Descricao, Empresa, Fotos, fk\_Usuario\_Id)

VALUES ('Analista de Dados', 'SQL, Python e Power BI', 'DataCorp', NULL, 5);

INSERT INTO Vagas (Titulo, Descricao, Empresa, Fotos, fk\_Usuario\_Id)

VALUES ('Front-end Developer', 'React e UX/UI', 'WebDesigners', NULL, 1);

INSERT INTO Vagas (Titulo, Descricao, Empresa, Fotos, fk\_Usuario\_Id)

VALUES ('Backend com Spring', 'Projetos em Java Spring', 'BackSolutions', NULL, 2);

--------

INSERT INTO LinksRedes (endereco, tipo\_rede, fk\_id\_usuario)

VALUES ('https://linkedin.com/in/carlos', 'LinkedIn', 1);

INSERT INTO LinksRedes (endereco, tipo\_rede, fk\_id\_usuario)

VALUES ('https://github.com/ana', 'GitHub', 2);

INSERT INTO LinksRedes (endereco, tipo\_rede, fk\_id\_usuario)

VALUES ('https://linkedin.com/in/joao', 'LinkedIn', 3);

INSERT INTO LinksRedes (endereco, tipo\_rede, fk\_id\_usuario)

VALUES ('https://instagram.com/maria', 'Instagram', 4);

INSERT INTO LinksRedes (endereco, tipo\_rede, fk\_id\_usuario)

VALUES ('https://github.com/ricardo', 'GitHub', 5);

-------

INSERT INTO Notificacao (titulo, descricao, data\_envio)

VALUES ('Nova Vaga Disponível', 'Confira a nova vaga publicada.', SYSDATE);

INSERT INTO Notificacao (titulo, descricao, data\_envio)

VALUES ('Evento Confirmado', 'Semana de Tecnologia confirmada!', SYSDATE);

INSERT INTO Notificacao (titulo, descricao, data\_envio)

VALUES ('Atualização de Perfil', 'Não se esqueça de atualizar seu perfil.', SYSDATE);

INSERT INTO Notificacao (titulo, descricao, data\_envio)

VALUES ('Cadastro Pendente', 'Confirme seu e-mail.', SYSDATE);

INSERT INTO Notificacao (titulo, descricao, data\_envio)

VALUES ('Novo Curso', 'Curso de Cloud Computing disponível.', SYSDATE);

-------

INSERT INTO Cursos (Nome, Sigla) VALUES ('Análise e Desenvolvimento de Sistemas', 'ADS');

INSERT INTO Cursos (Nome, Sigla) VALUES ('Fabricação Mecânica', 'FMEC');

INSERT INTO Cursos (Nome, Sigla) VALUES ('Logística', 'LOG');

INSERT INTO Cursos (Nome, Sigla) VALUES ('Projetos Mecânicos', 'PMEC');

INSERT INTO Cursos (Nome, Sigla) VALUES ('Polímeros', 'POL');

-------

INSERT INTO Usuario\_curso (fk\_Cursos\_ID, fk\_Usuario\_Id) VALUES (1, 1);

INSERT INTO Usuario\_curso (fk\_Cursos\_ID, fk\_Usuario\_Id) VALUES (3, 2);

INSERT INTO Usuario\_curso (fk\_Cursos\_ID, fk\_Usuario\_Id) VALUES (2, 3);

INSERT INTO Usuario\_curso (fk\_Cursos\_ID, fk\_Usuario\_Id) VALUES (1, 4);

INSERT INTO Usuario\_curso (fk\_Cursos\_ID, fk\_Usuario\_Id) VALUES (4, 5);

------

INSERT INTO conversa (remetente, destinatario, id\_origem, mensagem) VALUES (1, 2, NULL, 'Oi, tudo bem?');

INSERT INTO conversa (remetente, destinatario, id\_origem, mensagem) VALUES (2, 1, 1, 'Tudo sim, e você?');

INSERT INTO conversa (remetente, destinatario, id\_origem, mensagem) VALUES (3, 4, NULL, 'Olá, já está aprovada sua conta.');

INSERT INTO conversa (remetente, destinatario, id\_origem, mensagem) VALUES (4, 3, 3, 'Obrigado!');

INSERT INTO conversa (remetente, destinatario, id\_origem, mensagem) VALUES (5, 1, NULL, 'Boa sorte na entrevista!');

-------

SELECT \* FROM USUARIO;

SELECT \* FROM CONVERSA;

SELECT \* FROM EVENTOS;

SELECT \* FROM VAGAS;

SELECT \* FROM LINKSREDES;

SELECT \* FROM local\_trabalho;

SELECT \* FROM NOTIFICACAO;

select \* from cursos;

select \* from Usuario\_curso;

select \* from TipoUsuario;

----------------------------------------------------------------------

-- questão 2 letra D

--Usuario

create index indice\_id\_usuario on usuario (Id\_Usuario);

--mensagem: criação para fazer o pai da arvore

create index indice\_origem\_mensagem on conversa(id\_origem)

------------------------------------------------

-- questão 3 letra A

SELECT u.Nome AS Nome\_Usuario, c.Nome AS Nome\_Curso

FROM Usuario\_curso us

INNER JOIN Usuario u

ON us.fk\_Usuario\_Id = u.Id\_Usuario

INNER JOIN Cursos c

ON us.fk\_Cursos\_ID = c.Id\_curso;

-- questão 3 letra B

SELECT count(\*) as total\_eventos from eventos where TO\_CHAR(DataInicio,'YYYY')='2025'

-- questão 3 letra C

-- só irá retornar o usuario 5, usuario que ninguem quer conversar

SELECT u.Nome, u.Id\_Usuario

FROM Usuario u

LEFT JOIN conversa c

ON u.Id\_Usuario = c.destinatario

WHERE c.destinatario IS NULL;

-- questão 3 letra D

-- usuario participativos na plataforma, que fizeram pelo menos uma publicação de vaga OU evento, não retornará o ID 4

SELECT u.Nome, u.Id\_Usuario

FROM Usuario u

JOIN Eventos e ON u.Id\_Usuario = e.fk\_Usuario\_Id

UNION

SELECT u.Nome, u.Id\_Usuario

FROM Usuario u

JOIN Vagas v ON u.Id\_Usuario = v.fk\_Usuario\_Id;

--questão 3 letra E

--voltará apenas o usuário ID 3 pois ele não tem vagas publicadas, apenas eventos

SELECT u.Nome, u.Id\_Usuario

FROM Usuario u

JOIN Eventos e ON u.Id\_Usuario = e.fk\_Usuario\_Id

MINUS

SELECT u.Nome, u.Id\_Usuario

FROM Usuario u

JOIN Vagas v ON u.Id\_Usuario = v.fk\_Usuario\_Id;

--questão 3 letra F

--retorna apenas os usuários que possuem vagas E eventos

SELECT u.Nome, u.Id\_Usuario

FROM Usuario u

JOIN Eventos e ON u.Id\_Usuario = e.fk\_Usuario\_Id

INTERSECT

SELECT u.Nome, u.Id\_Usuario

FROM Usuario u

JOIN Vagas v ON u.Id\_Usuario = v.fk\_Usuario\_Id;

--questão 4 letra A

--retorna apenas os locais de trabalho que possuem usuários

SELECT \* FROM LOCAL\_TRABALHO

WHERE ID\_LOCAL\_TRABALHO IN (SELECT ID\_LocalTrabalho FROM USUARIO);

--questão 4 letra B

--retorna apenas os locais de trabalho que NÃO possuem usuários

SELECT \* FROM LOCAL\_TRABALHO LT

WHERE NOT EXISTS (SELECT 1 from usuario U where U.ID\_LOCALTRABALHO = LT.ID\_LOCAL\_TRABALHO);

--questão 4 letra C

--altera todos os criadores de eventos para o usuário do tipo ADMINISTRADOR (ID 2)

UPDATE eventos set fk\_usuario\_id = (select id\_usuario from usuario where ID\_TIPOUSUARIO = 2 AND ROWNUM = 1);

--questão 4 letra D

--deleta todas as vagas que foram criadas por usuários do tipo ALUNO (ID 1)

DELETE FROM VAGAS WHERE FK\_USUARIO\_ID IN (SELECT ID\_USUARIO FROM USUARIO WHERE ID\_TIPOUSUARIO = 1);

--questão 5 letra A

--lista todos os alunos do curso que será informado na procedure: ADS, FMEC, LOG, PMEC, POL

CREATE OR REPLACE PROCEDURE SP\_ConsultaAlunosCurso(PSiglaCurso VARCHAR2) AS

VID\_CURSO CURSOS.ID\_CURSO%TYPE;

BEGIN

SELECT ID\_CURSO INTO VID\_CURSO FROM CURSOS WHERE SIGLA LIKE UPPER(PSiglaCurso);

FOR X IN(

SELECT U.NOME NOME, UC.FK\_CURSOS\_ID ID\_CURSO, C.NOME CURSO, U.EGRESSOS EGRESSO FROM USUARIO U

INNER JOIN USUARIO\_CURSO UC

ON UC.FK\_USUARIO\_ID = U.ID\_USUARIO

INNER JOIN CURSOS C

ON C.ID\_CURSO = UC.FK\_CURSOS\_ID

WHERE UC.FK\_CURSOS\_ID = VID\_CURSO

)

LOOP

IF X.EGRESSO = 1 THEN

DBMS\_OUTPUT.PUT\_LINE('Aluno: '||X.NOME||' | Curso: '|| X.CURSO||' | Egresso? SIM');

ELSE

DBMS\_OUTPUT.PUT\_LINE('Aluno: '||X.NOME||' | Curso: '|| X.CURSO||' | Egresso? NÃO');

END IF;

END LOOP;

EXCEPTION

WHEN NO\_DATA\_FOUND THEN

DBMS\_OUTPUT.PUT\_LINE('Curso não encontrado');

WHEN OTHERS THEN

DBMS\_OUTPUT.PUT\_LINE('Erro: ' || SQLERRM);

END;

--deverá ser informado no exec qual a sigla do curso que deseja ver os usuários: ADS, FMEC, LOG, PMEC, POL

SET SERVEROUTPUT ON;

EXEC SP\_ConsultaAlunosCurso('ads');

--questão 5 letra B

--essa função retorna a quantidade de mensagens trocadas entre dois usuários

CREATE OR REPLACE FUNCTION Conta\_Mensagens(P\_ID1 IN CONVERSA.REMETENTE%TYPE, P\_ID2 IN CONVERSA.REMETENTE%TYPE)

RETURN VARCHAR2 IS

V\_CONTADOR NUMBER;

V\_VERIFICA\_REMETENTE USUARIO.ID\_USUARIO%TYPE;

V\_VERIFICA\_DESTINATARIO USUARIO.ID\_USUARIO%TYPE;

BEGIN

SELECT ID\_USUARIO INTO V\_VERIFICA\_REMETENTE FROM USUARIO WHERE ID\_USUARIO = P\_ID1;

SELECT ID\_USUARIO INTO V\_VERIFICA\_DESTINATARIO FROM USUARIO WHERE ID\_USUARIO = P\_ID2;

SELECT COUNT(\*) INTO V\_CONTADOR FROM CONVERSA WHERE ((REMETENTE = P\_ID1 AND DESTINATARIO = P\_ID2) OR (REMETENTE = P\_ID2 AND DESTINATARIO = P\_ID1));

RETURN (''||V\_CONTADOR);

EXCEPTION

WHEN NO\_DATA\_FOUND THEN

RETURN 'Usuário inexistente!';

END;

SELECT Conta\_Mensagens(1,5) from dual;

--questão 5 letra C

--sempre que um usuário for aprovado no sistema após o seu cadastro, será gerado um log de quem aprovou ele junto de seu nome

CREATE TABLE TABLOG(

ID\_LOG NUMBER GENERATED ALWAYS AS IDENTITY,

USUARIO VARCHAR2(50),

DATALOG DATE,

DESCRICAO VARCHAR2(100)

)

CREATE OR REPLACE TRIGGER Usuario\_Aprovado

AFTER UPDATE OF USER\_ENABLED ON USUARIO

FOR EACH ROW

WHEN (NEW.USER\_ENABLED = 1)

BEGIN

INSERT INTO TABLOG (USUARIO, DATALOG, DESCRICAO) VALUES (USER, SYSDATE, 'Usuário '||:NEW.NOME||' teve seu cadastro aprovado!');

END;

UPDATE USUARIO SET USER\_ENABLED = 1 WHERE ID\_USUARIO = 4;

--questão 5 letra D

--essa trigger cancela todo o insert do evento se a data de início for maior que a data de fim

CREATE OR REPLACE TRIGGER Verifica\_Data\_Evento

BEFORE INSERT ON EVENTOS

FOR EACH ROW

BEGIN

IF :NEW.DATAINICIO > :NEW.DATAFIM THEN

RAISE\_APPLICATION\_ERROR(-20001, 'Data de início do evento é maior do que a de fim, favor alterar!');

END IF;

END;

INSERT INTO EVENTOS (TITULO, DATAINICIO, DATAFIM, LOCAL\_EVENTO, DESCRICAO, FOTOS, FK\_USUARIO\_ID)

VALUES ('teste de trigger', DATE '2025-06-07', DATE '2025-06-06', 'MINHA CASA', 'É SÓ UM TESTE, SE FUNCIONAR DEU BOM', NULL, 1);

select \* from eventos;