2ª Fase do projeto

Gestão de loja de vendas online

Grupo 35

Francisco Braço-Forte - 57450
José Morgado - 59457
Bernardo Viegas - 60311

Turno p.3

Descrição da base de dados

Pretendemos implementar uma base de dados para gerir uma loja de vendas online. Esta base de dados será usada para gerir os produtos da loja, as encomendas, os empregados e vários outros dados sobre os seus clientes.

A loja vende produtos, cada um deles com um identificador, um nome, um preço e uma descrição. A loja também deverá saber qual a quantidade de stock existente de cada produto e o peso de cada produto. Cada produto irá também ter uma categoria que irá ter dados associados a si, tais como o id da categoria e o seu nome.

Os dados dos clientes serão guardados na base de dados, sendo estes o email, NIF, nome, saldo, password, morada e pontos (com cada compra o cliente irá receber pontos e com estes pode adquirir cupões de desconto para compras futuras).

Os dados de uma encomenda vão ser guardados, sendo estes o id da encomenda, a data, o tipo de entidade que vai realizar a encomenda, os produtos encomendados, as suas respectivas quantidades, o email do distribuidor associado, a matrícula do veículo que será utilizado e um estado de encomenda (o estado de encomenda pode ter três valores: “processamento”, “expedido”, “entregue”).

Os veículos utilizados para a entrega de encomendas têm uma matrícula associada a si e uma capacidade máxima. Além disso, pretende-se saber se este está a ser utilizado ou não.

Os dados dos empregados também serão guardados, sendo estes o email, NIF, nome, morada e id de horário. Existem dois tipos de empregados: técnicos (tratam do apoio ao cliente por forma de tickets) e distribuidores (tratam da entrega das encomendas).

Os tickets consistem em perguntas ou queixas feitas pelos clientes no site, e estes estão associados ao email do cliente que criou o ticket e a um id de categoria de produtos. Só técnicos associados à categoria do ticket criado pelo cliente é que podem ver e responder ao ticket. Além disso, cada ticket contém uma descrição onde será guardada a mensagem que o cliente escreveu e o estado em que se encontra (aberto - por resolver; fechado - já tratado). Apesar de o ticket ser atendido por um Técnico, o seu estado poderá ainda estar em “aberto”, pois o tratamento do pedido pode encontrar-se em processamento (não finalizado).

Os cupões de desconto terão um identificador, um preço em pontos, percentagem de desconto, datas de início e fim de validade, descrição e categoria de produtos a que o cupão é aplicável. No momento da criação de uma encomenda, se o cliente tiver um cupão da categoria de produtos na qual o produto da encomenda pertence, então será feito o desconto automático no preço do produto. Deste modo, o cliente deixará de poder usufruir deste cupão, pois já foi utilizado.
Caso o cupão se encontre ainda dentro do período de disponibilidade (entre as datas de início e fim), o cliente poderá eventualmente voltar a comprá-lo com a sua carteira de pontos.

Os clientes podem realizar várias encomendas à loja. A loja terá empregados prontos para prestar apoio aos clientes caso seja necessário (por exemplo um reembolso). Os clientes podem comprar cupões com os seus pontos, e após os utilizarem, estes serão removidos das suas contas. Caso o cliente não utilize o seu cupão dentro do prazo de validade este também será removido da sua conta.

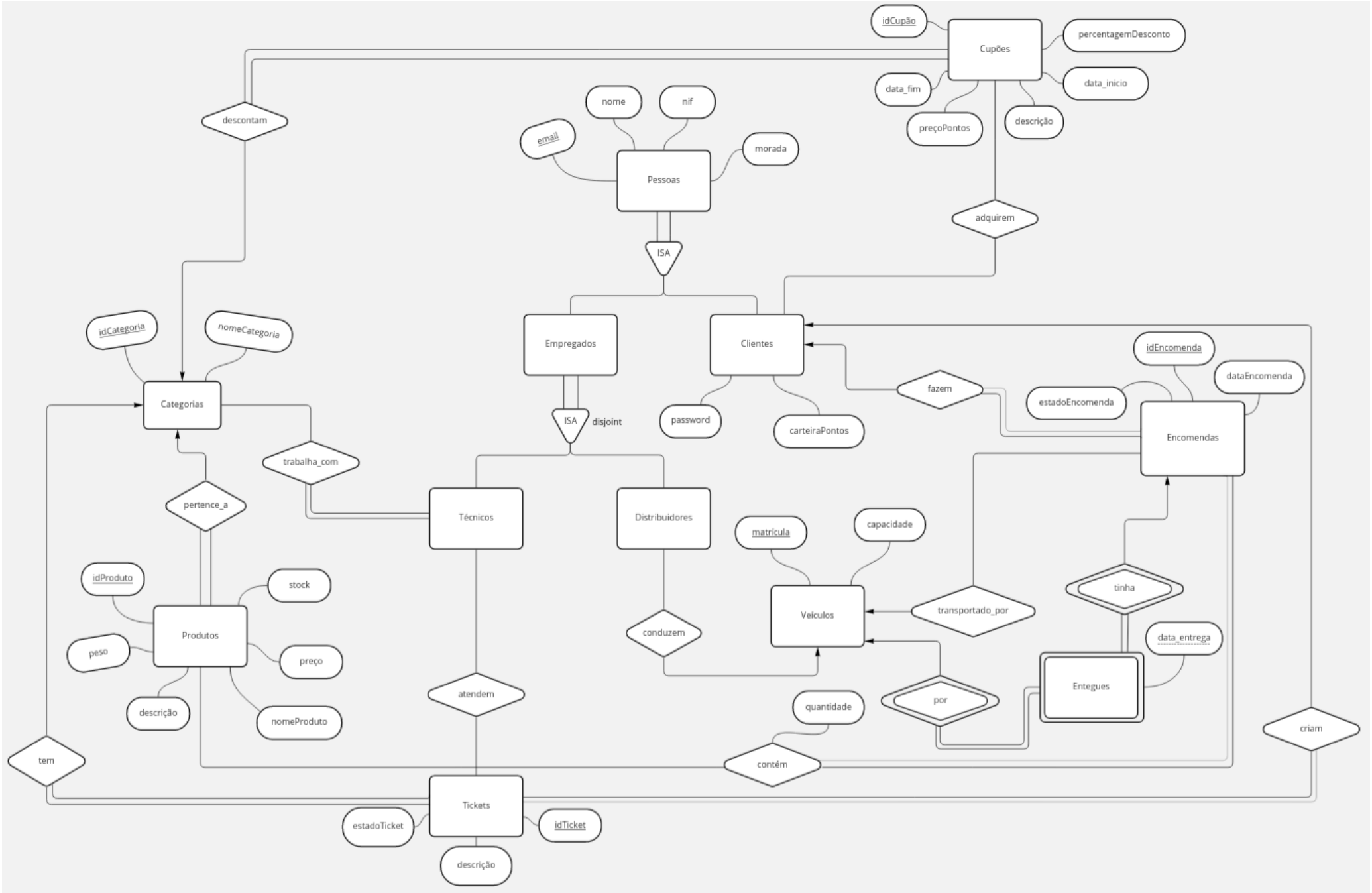
Cada produto terá uma categoria associada a si. Cada encomenda terá um veículo e distribuidor associados, bem como um tipo de produto (sendo especificada a quantidade que se pretende encomendar).

Os Técnicos estão encarregues de responder aos tickets (dúvidas ou queixas dos clientes) das categorias de produtos das quais eles estão encarregues.

Os distribuidores têm um veículo de entregas associado, estando cada encomenda atribuída a uma dupla distribuidor, veículo.

Pretende-se ainda guardar a matrícula do veículo, data de entrega e id das encomendas que já foram entregues.

Modelo ER


idCupão
Cupões
percentagemDesconto
data_fim
data_inicio
preçoPontos
descrição
descontam
nome
nif
email
morada
Pessoas
adquirem
ISA
idCategoria
nomeCategoria
Empregados
Clientes
idEncomenda
dataEncomenda
fazem
estadoEncomenda
Categorias
ISA
disjoint
password
carteiraPontos
Encomendas
trabalha_com
pertence_a
Técnicos
Distribuidores
matrícula
capacidade
idProduto
stock
tinha
Produtos
Veículos
transportado_por
peso
preço
conduzem
data_entrega
descrição
nomeProduto
atendem
Entegues
por
quantidade
criam
tem
contém
Tickets
estadoTicket
idTicket
descrição

Modelo Relacional

Esquema relacional para os conjuntos de entidades

Cupões (idCupão, preçoPontos, percentagemDesconto, data_início, data_fim, descrição, idCategoria)
- idCategoria - chave estrangeira referindo **Categorias**.

Pessoas (email, nome, nif, morada)

Empregados (email)
- email - chave estrangeira referindo **Pessoas**.

Técnicos (email)
- email - chave estrangeira referindo **Empregados**.

Distribuidores (email)
- email - chave estrangeira referindo **Empregados**.

Clientes (email, passwordCliente, carteiraPontos)
- email - chave estrangeira referindo **Pessoas**.

Categorias (idCategoria, nomeCategoria)

Produtos (idProduto, nomeProduto, peso, stock, preço, descrição, idCategoria)
- idCategoria - chave estrangeira referindo **Categorias**.

Veículos (matrícula, capacidade)

Encomendas (idEncomenda, dataEncomenda, estadoEncomenda, email)
- email - chave estrangeira referindo **Clientes**.

Tickets (idTicket, estadoTicket, descrição, idCategoria, email)
- idCategoria - chave estrangeira referindo **Categorias**.
- email - chave estrangeira referindo **Clientes**.

Entregues (matrícula, idEncomenda, data_entrega)
- idCategoria - chave estrangeira referindo **Categorias**.
- matrícula - chave estrangeira referindo **Veículos**.

Esquema relacional para os conjuntos de relações

adquirem (email, idCupão)
- email - chave estrangeira referindo **Clientes**.
- idCupão - chave estrangeira referindo **Cupões**.

trabalha_com (idCategoria, email)
- idCategoria - chave estrangeira referindo **Categorias**.
- email - chave estrangeira referindo **Técnicos**.

atendem (email, idTicket)
- email - chave estrangeira referindo **Técnicos**.
- idTicket - chave estrangeira referindo **Tickets**.

contém (idProduto, idEncomenda, quantidade)
- idProduto - chave estrangeira referindo **Produtos**.
- idEncomenda - chave estrangeira referindo **Encomendas**.

conduzem (email, matrícula)
- email - chave estrangeira referindo **Distribuidores**.
- matrícula - chave estrangeira referindo **Veículos**.

transportado_por (idEncomenda, matrícula)
- idEncomenda - chave estrangeira referindo **Encomendas**.
- matrícula - chave estrangeira referindo **Veículos**.

Consultas SQL

1. Para cada cliente, consultar o número de tickets que já se encontram no estado “fechado” numa categoria X de produtos.

```
select email, count (idTicket) tickets_fechados
from clientes inner join tickets using (email)
        inner join categorias using(idCategoria)
where nomeCategoria = ‘X’ and estadoTicket = 'fechado'
group by email
order by tickets_fechados desc;
```

Agrupar os clientes por email que já tenham criado tickets, cujo nome da categoria é dado por "X" e o estado do ticket encontra-se como “fechado”. Deste modo, para cada cliente (pelo seu email), contar o número de tickets referidos anteriormente.

2. Os técnicos que já responderam a tickets de todas as categorias, nas quais ele está atribuído.

```
select tecnicos.email, pessoas.nome
from tecnicos, pessoas
where tecnicos.email = pessoas.email and not exists (
        (select idCategoria
        from trabalha_com
        where trabalha_com.email = tecnicos.email)
        minus
        (select distinct idCategoria
        from atendem, tickets
        where atendem.email = tecnicos.email and
                atendem.idTicket = tickets.idTicket)
);
```

Pretende-se verificar para cada técnico se o conjunto das categorias nas quais atendeu tickets é igual ao conjunto de todas as categorias de produtos nas quais está encarregue. Deste modo, para satisfazer tal condição, é necessário que o resultado da operação retirar de todas as categorias que trabalha, as categorias nas quais o técnico já atendeu tickets, dê o conjunto vazio, ou seja, os conjuntos de categorias são iguais. Considere X o conjunto de todas as categorias e Y o conjunto das categorias nas quais o técnico já tratou de tickets. Se X - Y = {}, então o técnico já respondeu a tickets de todas as categorias que pertence.

3. Quais os produtos que nunca foram encomendados pelo cliente X, sendo que a categoria na qual pertencem nunca teve tickets abertos pelos clientes.

```
select nomeProduto, nomeCategoria
from produtos inner join categorias using(idCategoria)
where idProduto not in (
        select distinct idProduto
        from contem inner join encomendas using (idEncomenda)
                        inner join clientes using (email)
        where email = 'X'
) and idCategoria not in (
        select distinct idCategoria from tickets
)
```

Selecionam-se os produtos que nunca foram encomendados pelo cliente “X” (verificar que o idProduto não pertence ao resultado de todos os produtos que foram encomendados pelo cliente “X”) e que a categoria na qual pertence nunca teve tickets abertos (verificar que o idCategoria não consta no resultado dado pela seleção de todas as categorias em que foram criados tickets).

Triggers

1. Garante que o cliente tem pontos suficientes para comprar um cupão.

```
create or replace trigger comprar_cupao before insert on adquirem
        for each row
                declare cl_pontos number; cp_preco number; resultado number;
                begin
                        select carteiraPontos into cl_pontos from clientes
                        where clientes.email = :new.email;
                        select precoPontos into cp_preco from cupoes
                        where cupoes.idCupao = :new.idCupao;
                        resultado := cl_pontos - cp_preco;
                        if (resultado >= 0) then
                                update clientes set carteiraPontos = resultado
                                where clientes.email = :new.email;
                        else
                                Raise_Application_Error (
                                        -20100,  'Cliente não tem pontos suficientes para
                                comprar o cupão - ' || 'Carteira: ' || to_char(cl_pontos) ||
                                '; Preço: ' || to_char(cp_preco));
                        end if;
        end;
/
```

Antes de inserir um novo tuplo na relação **adquirem**, é necessário verificar que o cliente tem pontos suficientes para poder comprar o cupão que pretende. Deste modo, verifica-se se os pontos, resultantes do desconto do preço do cupão na carteira do cliente, é superior ou igual a zero. Caso seja verdade, então a compra sucede com sucesso e a carteira de pontos do cliente é atualizada para o valor resultante da diferença. Caso contrário, será lançada uma excepção.

2. Garante que cada empregado não pode ser distribuidor e técnico ao mesmo tempo.

```
create or replace trigger inserir_tecnico before insert on tecnicos
        for each row
                declare r number(1);
                begin
                        select count(*) into r from distribuidores
                        where email = :new.email;
                        if (r > 0) then
                                Raise_Application_Error (-20100, 'Empregado já está
        inserido como distribuidor.');
                        end if;
                end;
/
create or replace trigger inserir_distribuidor before insert on distribuidores
        for each row
                declare r number(1);
                begin
                        select count(*) into r from tecnicos
                        where email = :new.email;
                        if (r > 0) then
                                Raise_Application_Error (-20100, 'Empregado já está
        inserido como técnico.');
                        end if;
                end;
/
```

Antes de inserir um dado técnico/distribuidor, é necessário verificar se o mesmo não pertence à relação oposta, isto é, por exemplo, se um dado empregado já existir na relação **técnicos** (se o email do empregado, que se pretende inserir, estiver presente na relação), então não poderá ser inserido na relação **distribuidores** e um excepção será lançada.

3. Assegura que cada técnico irá responder apenas aos tickets que pertencem a uma das categorias nas quais estão encarregues de atender os pedidos.

```
create or replace trigger responder_ticket before insert on atendem
    for each row
        declare categoriaTicket number; ocorreCategoria number;
        begin
            select idCategoria into categoriaTicket from tickets
            where :new.idTicket = tickets.idTicket;
            select count(*) into ocorreCategoria from trabalha_com
            where categoriaTicket = trabalha_com.idCategoria
                and trabalha_com.email = :new.email;
            if (ocorreCategoria = 0) then
                Raise_Application_Error (
                -20100,
                Técnico não pertence à categoria do ticket (' ||
                categoriaTicket || ')';
            end if;
        end;
/
```

Antes de considerar um ticket como lido por um técnico (inserir na relação **atendem**), é necessário obter a categoria de produtos na qual o ticket foi aberto e verificar se a mesma ocorre no conjunto de categorias onde o técnico está encarregado de atender tickets dos clientes. Caso o técnico não esteja atribuído à categoria do ticket, então será lançada uma exceção a avisar.

4. Verifica se a quantidade de produtos de uma dada encomenda não excede stock. Caso respeite esta condição, então será retirada essa quantidade ao stock do produto.

```
create or replace trigger stock_produto before insert on contem
        for each row
        declare
        stockProduto number; resultado number;
        begin
                select stock into stockProduto from produtos
                where :new.idProduto = produtos.idProduto;
                resultado := stockProduto - :new.quantidade;
                if (resultado >= 0) then
                        update produtos
                        set stock = resultado
                        where produtos.idProduto = :new.idProduto;
                 else
                        Raise_Application_Error (
                        -20100, 'Quantidade excede o stock do produto - ' ||
                        'Stock: ' || to_char(stockProduto) ||
                        '; Quantidade: ' || :new.quantidade
                 );
                end if;
        end;
/
```

Antes de inserir a quantidade de um tipo de produto encomendado, é necessário obter o seu stock. De seguida, retira-se o número de produtos que se pretende comprar na encomenda e verificar se o valor é superior ou igual a zero, ou seja, se o número de unidades a comprar não excede o stock atual. Se não exceder, então pode ser comprado e atualiza-se o stock do produto para o resultado final da diferença calculada. Caso contrário, é lançada uma excepção.

5. Verifica se há capacidade disponível no veículo para poder transportar uma dada encomenda.

```
create or replace trigger peso_disponivel before insert on contem
    for each row
    declare
        veiculo_atribuido number; veiculo varchar2(8); p_total number;
        c_veiculo number; p_disponivel number; p_encomenda number;
        p_total_encomenda number;
    begin
        select count(*) into veiculo_atribuido from transportado_por
        where :new.idEncomenda = transportado_por.idEncomenda;
        if (veiculo_atribuido > 0) then
            select matricula into veiculo from transportado_por
            where :new.idEncomenda = transportado_por.idEncomenda;
            with encomendas_pesos as (
                select idEncomenda,
                        quantidade * peso p_por_encomenda
                from encomendas
                        inner join transportado_por using(idEncomenda)
                        inner join contem using(idEncomenda)
                        inner join produtos using(idProduto)
                where idEncomenda <> :new.idEncomenda
                        and matricula = veiculo
            )
            select sum(p_por_encomenda) into p_total
            from encomendas_pesos;
            select capacidade into c_veiculo
            from veiculos where matricula = veiculo;
            -- nvl é necessário pois sum(p_por_encomenda)
            -- pode retornar null
            p_disponivel := c_veiculo - nvl(p_total, 0);
            select peso into p_encomenda from produtos
            where idProduto = :new.idProduto;
            p_total_encomenda := p_encomenda * :new.quantidade;
```

```
                    if (p_total_encomenda > p_disponivel) then
                        Raise_Application_Error (
                        -20100, 'Veículo '|| veiculo || ' está cheio - ' ||
                            'Peso disponível: ' || p_disponivel ||
                            '; Pretendido: ' || to_char(p_total_encomenda)
                        );
                    end if;
                end if;
            end;
/
```

De modo a obter a capacidade atual do veículo é necessário saber em qual veículo a encomenda será transportada. Caso ainda não tenha nenhum veículo atribuído, a execução do trigger termina. Se já estiver com um veículo atribuído (em **transportado_por**) para ser entregue, então obtém-se a sua matrícula. De seguida, somam-se todos os pesos de todas as encomendas que têm esse mesmo veículo atribuído (tendo em consideração que o cálculo do peso é dado por peso do produto vezes a quantidade a adquirir, para cada encomenda). Além disso, também é necessário obter a capacidade máxima do veículo, o que leva ao cálculo da capacidade do veículo menos a soma dos pesos das encomendas, obtendo assim a capacidade atual do veículo. Por fim, após calcular o peso total da encomenda, ou seja, o peso da encomenda (aquela a inserir a sua quantidade na relação **contem**) vezes a quantidade do produto, verifica-se se esse valor obtido é superior à capacidade atual do veículo. Se essa condição for verdadeira, então lança-se uma excepção a informar que o veículo encontra-se cheio.

```
create or replace trigger peso_disponivel_com_quantidade
        before insert on transportado_por
        for each row
        declare
                ja_com_quantidade number; veiculo varchar2(8); p_total number;
                c_veiculo number; p_disponivel number;
                p_total_encomenda number;
        begin
                select count(*) into ja_com_quantidade from transportado_por
                where :new.idEncomenda = transportado_por.idEncomenda;
                if (ja_com_quantidade > 0) then
                        select matricula into veiculo from transportado_por
                        where :new.idEncomenda = transportado_por.idEncomenda;
                        with encomendas_pesos as (
                                select idEncomenda,
                                        quantidade * peso p_por_encomenda
                                from encomendas
                                        inner join transportado_por using(idEncomenda)
                                        inner join contem using(idEncomenda)
                                        inner join produtos using(idProduto)
                                where idEncomenda <> :new.idEncomenda
                                and matricula = veiculo
                        )
                        select sum(p_por_encomenda) into p_total
                        from encomendas_pesos;
                        select capacidade into c_veiculo from veiculos
                        where matricula = veiculo;
                        -- nvl é necessário pois sum(p_por_encomenda)
                        --pode retornar null
                        p_disponivel := c_veiculo - nvl(p_total, 0);
                        select quantidade * peso into p_total_encomenda
                        from contem inner join produtos using(idProduto)
                        where idEncomenda = :new.idEncomenda;
```

```
                    if (p_total_encomenda > p_disponivel) then
                    Raise_Application_Error (
                    -20100, 'Veículo '|| veiculo || ' está cheio - ' ||
                            'Peso disponível: ' || to_char(p_disponivel) ||
                            '; Pretendido: ' || to_char(p_total_encomenda)
                    );
                    end if;
            end if;
    end;
/
```

O procedimento, no geral, para o cálculo do peso é idêntico ao trigger **peso_disponivel**. Contudo, é necessário verificar inicialmente se para uma dada encomenda, se esta já se encontra com uma quantidade de produtos definida na relação **contem**. Caso já esteja, então será feita a verificação do peso disponível no veículo.

6. Garante que antes da alteração do estado da encomenda é necessário que a mesma tenha um veículo atribuído.

```
create or replace trigger estado_encomenda
        before update of estadoEncomenda on encomendas
        for each row
        declare r number(1);
        begin
                if (:new.estadoEncomenda = 'expedida'
                        or :new.estadoEncomenda = 'entregue') then
                        select count(*) into r from transportado_por
                        where idEncomenda = :new.idEncomenda;
                        if (r = 0) then
                                Raise_Application_Error (
                                -20100,  'Não é possível atualizar estado de
                                encomenda para ' || new.estadoEncomenda ||
                                ', visto que não tem um veículo atribuído'
                                );
                        end if;
                end if;
        end;
/
```

Se o estado da encomenda for para ‘expedida’ ou ‘entregue’, então será necessário verificar se a mesma tem um veículo atribuído na relação **transportado_por**.

7. Retira da tabela transportado_por uma dada encomenda, caso o seu estado tenha sido mudado para 'entregue'.

```
create or replace trigger encomenda_entregue
        after update of estado on encomendas
        for each row
                begin
                        if (:new.estadoEncomenda = 'entregue') then
                                insert into entregues (matricula, idEncomenda)
                                select matricula, idEncomenda from transportado_por
                                where transportado_por.idEncomenda =
                                :new.idEncomenda;
                                delete from transportado_por
                                where transportado_por.idEncomenda =
        :new.idEncomenda;
                        end if;
                end;
/
```

Após a atualização da relação **encomendas**, verificar se o estado da encomenda foi alterado para “entregue”. Se sim, então é necessário inserir na relação **entregues** a encomenda e removê-la da relação **transportado_por**.

8. Garante que não se insere uma encomenda na tabela transportado_por, sendo que já se encontra na tabela entregues.

```
create or replace trigger ja_entregue before insert on transportado_por
        for each row
        declare
                r number(1);
        begin
                select count(*) into r from entregues
                where :new.idEncomenda = idEncomenda;
                if (r > 0) then
                        Raise_Application_Error (
                        -20100, 'Encomenda ' || to_char(:new.idEncomenda) || ' já se
encontra entregue');
                end if;
        end;
/
```

Verifica se a encomenda a inserir em na relação **transportado_por** já se encontra na relação **entregues**. Se sim, então será lançada uma excepção.

Interface APEX

● Descrição

A aplicação GLVO foi desenvolvida no ORACLE Application Express (APEX), com recurso à instância que se encontra no servidor da universidade (servidor local).
Tem como finalidade a consulta, manipulação e inserção de dados de forma interativa na base de dados sobre a qual opera. Possui algumas funcionalidades, tais como obter informação sobre a validade dos cupões, tickets registados, pessoas registadas, categorias e produtos às quais pertencem e cupões adquiridos por um dados cliente.

● Racional

Para entrar na aplicação, há uma página de login para inserção das credenciais.
Após efetuar o login, existe uma página inicial, sob a qual estão indexadas (breadcrumbs) as diferentes páginas de interação na aplicação.
Seguem-se então as páginas criadas que definem a interface sobre a base de dados:

- Pessoas: página do tipo report e interativa com visualização da relação **pessoas**, contendo todos os seus atributos, estando subordinada à página inicial (parent entry) e com os breadcrumbs ativos.

- Tickets: página do tipo report e interativa de uma consulta sql **(1.)** que possibilita visualizar a informação base dos tickets, bem como o nome da categoria sobre a qual foram criados e o nome do cliente que o criou. Por ativação dos breadcrumbs, associou-se como página progenitora a Inicial e criou-se uma nova entrada de navegação associada à página inicial.

1. ```
   select idTicket as "Ticket",
          descricao as "Descrição",
          estadoTicket as "Estado",
          nomeCategoria as "Categoria",
          nome as "Cliente",
          email as "Email do Cliente"
   from tickets inner join categorias using(idCategoria)
          inner join pessoas using(email)
   ```
   ```

- Validade Cupão: página do tipo report e interativa de uma consulta sql **(1.)** que possibilita saber quantos dias falta até os cupões deixarem de ser válidos. Por ativação dos breadcrumbs, associou-se como página progenitora a Inicial e criou-se uma nova entrada de navegação associada à página inicial.

1. **select** idCupao **as** "Cupão",
   precoPontos **as** "Preço(€)",
   percentagemDesconto **as** "Desconto(%)",
   data_fim - **trunc**(**sysdate**) **as** "Validade(dias)",
   nomeCategoria **as** "Categoria"
   from cupoes **inner join** categorias **using**(idCategoria)

- Cupões Dados: página do tipo form, cujo report tem um form na tabela, com breadcrumb ativo, tendo a página inicial como ancestral. Além disso, também possui uma entrada de navegação para a página inicial. A tabela em questão é a dos cupões, tendo todos os seus atributos selecionados e o id do cupão como chave primária e preenchida com uma sequência já existente **(1.)**. Deste modo, para cada tuplo apresentado no report, existe um ícone em formato de lápis que redireciona para um form que permite editar ou remover esse tuplo da tabela. Também existe um botão a azul no canto superior direito que possibilita a criação de novos tuplos na tabela. Para substituir o id da categoria pelo seu nome, criou-se uma list of values from scratch denominada por “Categorias De Produtos”, por meio de uma sql query **(2.)**, do tipo dynamic. Por fim, em list of values, selecionou-se o atributo P32_IDCATEGORIA, alterou-se o seu type para select list, e em list of values como type shared component e associou-se a LOV “Categorias De Produtos”. Deste modo, quando for para selecionar a categoria, irá aparecer uma lista com os nomes das categorias, em vez do seu id.

1. **create sequence** seq_id_cupao
   start with 1
   increment by 1;

2. **select** nomeCategoria, idCategoria
   from categorias
   order by idCategoria

- Categorias de Produtos: página do tipo form e two page master detail para poder visualizar os produtos que pertencem a uma dada categoria e assim poder adicionar mais, se pretendido, ou então alterar o nome da categoria. Além disso, a página no master denomina-se por “Categorias de Produtos” e o detail por “Produtos”, os breadcrumbs como ativos, sendo a página inicial como parent entry e com uma entrada de navegação, estando subordinada à página inicial. Por fim, para a master table tem-se a tabela categorias, com todos os seus atributos selecionados, sendo o id da categoria como chave primária e form navigation order, e como detail table a tabela produtos, tendo rowid como chave primária e idcategoria como master detail foreign key para estabelecer a ligação entre as tabelas.

- Clientes: página do tipo report interativo com os breadcrumbs ativos, com a página inicial como ancestral e uma entrada de navegação subordinada à página inicial. Esta página permite visualizar o número de cupões adquiridos por um dado cliente e quais são eles. De seguida, criou-se um page Item do tipo select list com nome P42_CUPAO, tendo como região “Cupões Adquiridos” (região criada) e page action on selection como submit page. Além disso, em list of values desse mesmo page item, selecionou-se shared component e ID CUPAO (lista de valores criada e especificada em **(1.)** para obter os emails pelo nome do cliente) em list of values.
  Por fim, tem-se uma região do tipo classic report com a query **(2.)** para contar o número de cupões que um dado cliente adquiriu e na página principal, “Cupões adquiridos pelos Clientes”, tem-se uma query que lista os cupões adquiridos por um dado cliente selecionado **(3.)**.

1. **select** nome, email
   from pessoas **inner join** clientes **using(**email**)**

2. **select count(**idCupao**) "**Número de cupões adquiridos**"**
   from adquirem
   where email **=** :P42_CUPAO

3. **select** idcupao **"**Identificador**",** precoPontos **"**Preço**",** data_inicio **"**Início**",** data_fim **"**Fim**"**
   from ADQUIREM **inner join** cupoes **using(**idcupao**)**
   where email **=** :P42_CUPAO

- Responderam a todos os tickets: página do tipo report e interativa de uma consulta sql **(1. - semelhante à segunda consulta SQL)** que possibilita saber quais os técnicos que já responderam a tickets de todas as categorias nas quais está encarregue. Por ativação dos breadcrumbs, associou-se como página progenitora a Inicial e criou-se uma nova entrada de navegação associada à página inicial.

1. **select** tecnicos.email **as** "Email", pessoas.nome **as** "Nome"
   from tecnicos, pessoas
   where tecnicos.email = pessoas.email **and not exists** (
   (**select** idCategoria **from** trabalha_com
   where trabalha_com.email = tecnicos.email)
   minus
   (**select** distinct idCategoria **from** atendem, tickets
   where atendem.email = tecnicos.email
   and atendem.idTicket = tickets.idTicket)
)

- Tickets Fechados: página do tipo report interativo com os breadcrumbs ativos, com a página inicial como ancestral e uma entrada de navegação subordinada à página inicial. Esta página permite ver quais clientes têm tickets fechados, e quantos são, para uma dada categoria **(1. - semelhante à primeira consulta SQL)**. Criou-se uma simples região do tipo Static Content. De seguida, criou-se um item P44_CAT do tipo Select List com Label Categoria, Page Action on Selection como Submit Page e em Region “Categorias de Produtos”. Em List Of values, o Type marcado como Shared Component e em List Of Values tem-se “CATEGORIAS DE PRODUTOS” **(2.)**.

1. **select** email, **count** (idTicket) "Fechados"
   from clientes **inner join** tickets **using** (email)
   inner join categorias **using**(idCategoria)
   where nomeCategoria = :P44_CAT **and** estadoTicket = 'fechado'
   group by email **order by** "Fechados" **desc**

2. **select** nomeCategoria, nomeCategoria categoria
   from categorias **order by** nomeCategoria

- Sem Produtos e Sem Tickets Da Sua Categoria: página do tipo report interativo com os breadcrumbs ativos, com a página inicial como ancestral e uma entrada de navegação subordinada à página inicial. Esta página permite ver quais produtos, e sua categoria, que não foram comprados por um dado cliente e que não há tickets criados para a categoria na qual o produto pertence **(1. - semelhante à terceira consulta SQL)**. Criou-se uma simples região do tipo Static Content. De seguida, criou-se um item P46_CLIENTE do tipo Select List com Label Categoria, Page Action on Selection como Submit Page e em Region "Cliente". Em List Of values, o Type marcado como Shared Component e em List Of Values tem-se "CLIENTE EMAIL" **(2.)**.

1. **select** nomeProduto "Produto", nomeCategoria "Categoria"
   from produtos **inner join** categorias **using**(idCategoria)
   where idProduto **not in** (
   select distinct idProduto
   from contem **inner join** encomendas **using** (idEncomenda)
   inner join clientes **using** (email)
   where email = :P46_CLIENTE
) **and idCategoria not in** (
   select distinct idCategoria **from** tickets
)

2. **select** email, email cliente
   from clientes

- ## Mini manual de utilização

A aplicação possui uma página de login para autenticação do utilizador. Após introduzir o nome de utilizador e a palavra-passe, clica-se em “Log In” para entrar na aplicação. No lado esquerdo, possui uma lista com as diferentes páginas disponíveis, estando a inicial no topo. Se se pretender “minimizar” essa lista, no canto superior esquerdo existe um ícone (com três traços horizontais) para produzir tal efeito. Além disso, no canto superior direito existe um botão de sair da aplicação - “Log Out”.
Para cada página, caso se pretenda filtrar por algo (correspondência de nome de algo), basta escrever no retângulo branco acima da tabela de amostragem de dados (do lado direito de um ícone em formato de lupa) e clicar no botão “Go” ao lado desse campo. Além disso, em “Actions” é possível realizar outras operações, tais como criação de gráficos, ordenação, entre outras. Por fim, para cada pessoa, no lado esquerdo da sua informação existe um ícone em formato de lápis que possibilita ver apenas a informação dessa pessoa.
Segue-se então uma descrição breve das operações que se podem realizar para as diferentes páginas:

- Pessoas: nesta página é possível visualizar todas as pessoas registadas no sistema, mostrando para cada uma o seu email, nif, nome e morada.

- Tickets: possibilita saber quais os tickets que foram submetidos no sistema, mostrando para cada um o seu id, descrição (dúvida/problema), estado (aberto/fechado), categoria na qual se insere, e o nome e email do cliente que criou o ticket.

- Validade Cupão: descreve os cupões registados, mostrando para cada um o seu identificador, categoria pertencente, o seu preço em pontos, desconto na compra e validade em dias, ou seja, o tempo restante que falta até o cupão expirar.

- Cupões Dados: apresenta toda a informação dos cupões: identificador, preço em pontos, percentagem de desconto, data de início, data de fim e o seu identificador. Para editar um determinado cupão, basta clicar no ícone em formato de lápis no lado esquerdo da sua informação. Para cada informação, existe um campo onde é possível alterar o seu conteúdo. Para guardar as alterações efetuadas, basta clicar em “Apply Changes” no canto inferior direito. Caso se pretenda eliminar o cupão, é só clicar em “Delete”, ao lado de “Apply Changes”, ou se não quiser nenhuma das anteriores, então basta clicar em “Cancel” no canto inferior esquerdo.
  Por outro lado, para criar um novo cupão, clique em “Create”, sobre a tabela de amostragem da informação dos cupões. No menu de criação pode inserir os dados que descrevem o cupão (são todos obrigatórios, ou seja, nenhum pode ficar em branco; note que o identificador do cupão é gerado automaticamente). Para guardar a criação basta criar em “Create” no canto inferior direito, ou cancelar em “Cancel” no canto inferior esquerdo.

- Categorias de Produtos: caso se pretenda criar uma categoria, basta clicar em “Create” no canto superior direito e inserir o seu nome (o identificador é gerado automaticamente) e clicar em “Create” no canto inferior direito ou “Cancel” no canto inferior esquerdo. Para editar uma categoria, é só clicar no lápis do lado esquerdo do seu nome na tabela de listagem de categorias abaixo. Neste menu é possível alterar o nome da categoria ou remover (clicar em “Delete” no canto inferior direito, ao lado de “Save”). Também é possível alterar informação sobre um dado produto, clicando no campo que se pretende reescrever. Para adicionar um produto à categoria é só clicar em “Add Row” sobre a tabela de listagem dos produtos e inserir os dados. Para guardar as alterações feitas clica-se em “Save”, ao lado do botão delete, ou em “Cancel”, no lado esquerdo.

- Cliente: apresenta para um dado cliente o número de cupões adquiridos e quais são eles. Na secção “Cupões Adquiridos”, selecione no menu o nome do cliente. De seguida, em “Conta cupões” irá aparecer o número de cupões adquiridos e na tabela abaixo quais são eles, contendo informação sobre o seu identificador, preço, datas de início e fim da sua validade.

- Responderam a todos os tickets: apresenta quais os Técnicos que já responderam a tickets (estes poderão estar marcados como aberto ou fechado) de todas as categorias, listando para cada um o seu email e nome.

- Tickets Fechados: mostra quais clientes têm tickets fechados e quantos para uma dada categoria de produtos. Na secção “Categorias de Produtos”, pode selecionar a categoria que pretenda. Sobre essa mesma secção, serão mostrados os clientes, email e número de tickets fechados, para a categoria selecionada.

- Sem Produtos e Sem Tickets Da Sua Categoria: inicialmente são apresentados todos os produtos cuja categoria que pertencem não tem tickets criados pelos clientes. No menu em baixo, pode ser selecionado o email do cliente para filtrar pelos produtos que não foram encomendados por ele e também sem tickets criados na categoria que pertence.

Limitações

Relativamente aos cupões adquiridos pelos clientes, estes não podem ser selecionados deliberadamente pelo cliente durante a realização das suas compras, ou seja, serão aplicados automaticamente às encomendas cujo produto seja da mesma categoria que o cupão pertence.

Nas encomendas já entregues, não é possível registar o email do distribuidor que entregou a encomenda.

Opções tomadas

A participação do conjunto de entidades **Encomendas** na relação **transportado_por** é parcial, visto que no momento de criação de uma encomenda, a mesma poderá ainda não ter um veículo de transporte associado.

A participação do conjunto de entidades **Distribuidores** na relação **conduzem** é parcial, pois cada distribuidor poderá não ter um veículo associado num dado período de tempo (e.g. ocorrência de uma avaria num dos transportes).

Observações

- Tendo em conta a base de dados descrita na entrega do registo do tema do projeto, reparámos que ficaria com uma dimensão acrescida, isto é, para além das limitações definidas no enunciado do trabalho. Portanto, resolvemos simplificar as moradas em um atributo no conjunto de entidades **Pessoas** e retirar os fornecedores, a lista de compras e os horários dos empregados.

- A consulta 1. de SQL, sugerida na 1ª entrega do trabalho, foi substituída por uma outra que requer o uso de uma função de agregação, de modo a satisfazer os requisitos impostos no enunciado.

- Adição do atributo “nomeProduto” ao conjunto de entidades **Produtos** por questões de legibilidade.

- Renomeação do atributo “nome” para “nomeCategoria” no conjunto de entidades **Categorias** por questões de legibilidade.

- Renomeação do atributo “password” para “passwordCliente” no conjunto de entidades **Clientes** por questões de legibilidade.

- Renomeação do atributo “data” para “dataEncomenda” no conjunto de entidades **Encomendas** por questões de legibilidade.

- Remoção de “mas que já foram transportados em encomendas pelo distribuidor Y” da consulta 3. de SQL, uma vez que o modelo especificado não permite guardar de forma permanente o distribuidor que entregou uma dada encomenda.

- Adição do conjunto de entidades fraco **Entregues** com a finalidade de guardar as encomendas que já foram entregues. Especialmente necessário para a consulta 3. de SQL. Deste modo, utiliza-se a relação **transportado_por** para guardar as encomendas que têm um veículo atribuído e ainda não estão consideradas como entregues (o seu estado ainda não está marcado como entregue). A introdução deste conjunto de entidades foi adicionado no final da “Descrição da base de dados”.

- Adição da restrição “nas quais ele está atribuído” à consulta 2.