Universidade de Aveiro

Rent Buy Car

Base de Dados - 42532

Tiago Alvim – 95584
Vasco Costa – 97746

Junho – 2022

Professor Doutor – Carlos Manuel Azevedo Costa

Índice

Introdução[1]

Propunha-se com este trabalho realizar a implementação de uma aplicação gráfica com capacidade para interagir com uma base de dados externa. Para ser aplicada numa empresa de aluguer e venda de carros.

Seria possível a um cliente comprar e vender qualquer carro e alugar um conjunto destes que fosse destinado para tal fim. Registar clientes e funcionários. Como a compra e a venda do mesmo veículo em mais que uma ocasião é uma hipótese no mundo real esta foi tida em consideração. Ao aluguer encontra-se associado um registo para a anotação de eventuais danos.

Para isto foram utilizadas as ferramentas de *Microsoft SQL Server* e *Microsoft Visual Studio* de forma a realizar uma interface para utilização com a base de dados desenvolvida aplicando os conhecimentos adquiridos nas aulas.

Material a entregar

SQL

Na pasta entregue encontram-se mais 4 pastas. Em *Requisitos_diagramas* pode-se encontrar o percurso evolutivo do desenvolvimento do esquema conceptual da base de dados. Na pasta *Tabelas* é onde se encontram os ficheiros *.sql* para a criação da base de dados, *criar_base.sql*, e para a adição de *stored procedures(sp)*, *procedures.sql*, *triggers* em *triggers.sql* adição de alguns dados para propósitos de teste, mas com dados essenciais, nomeadamente as localizações, *meter_dados.sql*, e para funções contendo apenas uma, *funcs.sql*.

Para a execução por parte da docência poderá executar-se o ficheiro *CRIAR_TUDO.sql* onde está contemplada a criação de toda a base de dados e já foram executadas algumas operações para o teste geral da aplicação. Ou então é possível criar de raiz, utilizando primeiro o ficheiro *criar_base.sql* e depois *funcs.sql, triggers.sql, procedures.sql* e *meter_dados.sql*, não requerendo uma ordem especifica para executar.

Visual Studio

No interior da pasta *visual_studio/trab1* encontram-se os ficheiros de *Visual Basic*.

A *connection string* terá que ser alterada em todos os ficheiros dos *Forms* uma vez que a implementação da função, ***connect_to_bd()***, não ficou operacional, apresentando problemas na troca das páginas em que fechava a conexão. Assim em cada um dos *Form#_load* em cada *Form* terá que ser alterado o valor de ***CN*** para corresponder ao servidor do docente para a sua utilização.

- Form1.vb – linha 19
- Form2.vb – linha 20
- Form3.vb – linha 15
- Form4.vb – linha 12

[1] O esforço aplicado por cada um dos elementos:
- Tiago Alvim 95584 – 45%
- Vasco Costa 97746 – 55%

Vídeos

Em https://uapt33090-my.sharepoint.com/:v:/g/personal/vascovc_ua_pt/ETqbKu892TVEpbEAkm2eTWcBivYhgRoYkiP1R0OHIDH7dA?e=ReUK9T é possível visualizar-se o vídeo onde é demonstrado o funcionamento da interface estando também incluído na pasta o mesmo, com grande parte da demonstração dos métodos implementados e relevantes. Existem métodos que ao serem internos não foram possíveis de captar.

Análise de Requisitos

Na primeira reunião, na apresentação foi entregue um diagrama que apresentava falhas apresentando conceitos desnecessários e entidades repetidas que não iriam trazer maior riqueza e podiam ser generalizadas de outra forma. Levando a uma correção apresentada na *Figura 1* com o devido esquema relacional, *Figura 2*. Esta modificação consistiu em reduzir-se a entidade do online que não faria muito sentido devido à existência de balcões em que se poderia tomar um balcão especial que seria o online.

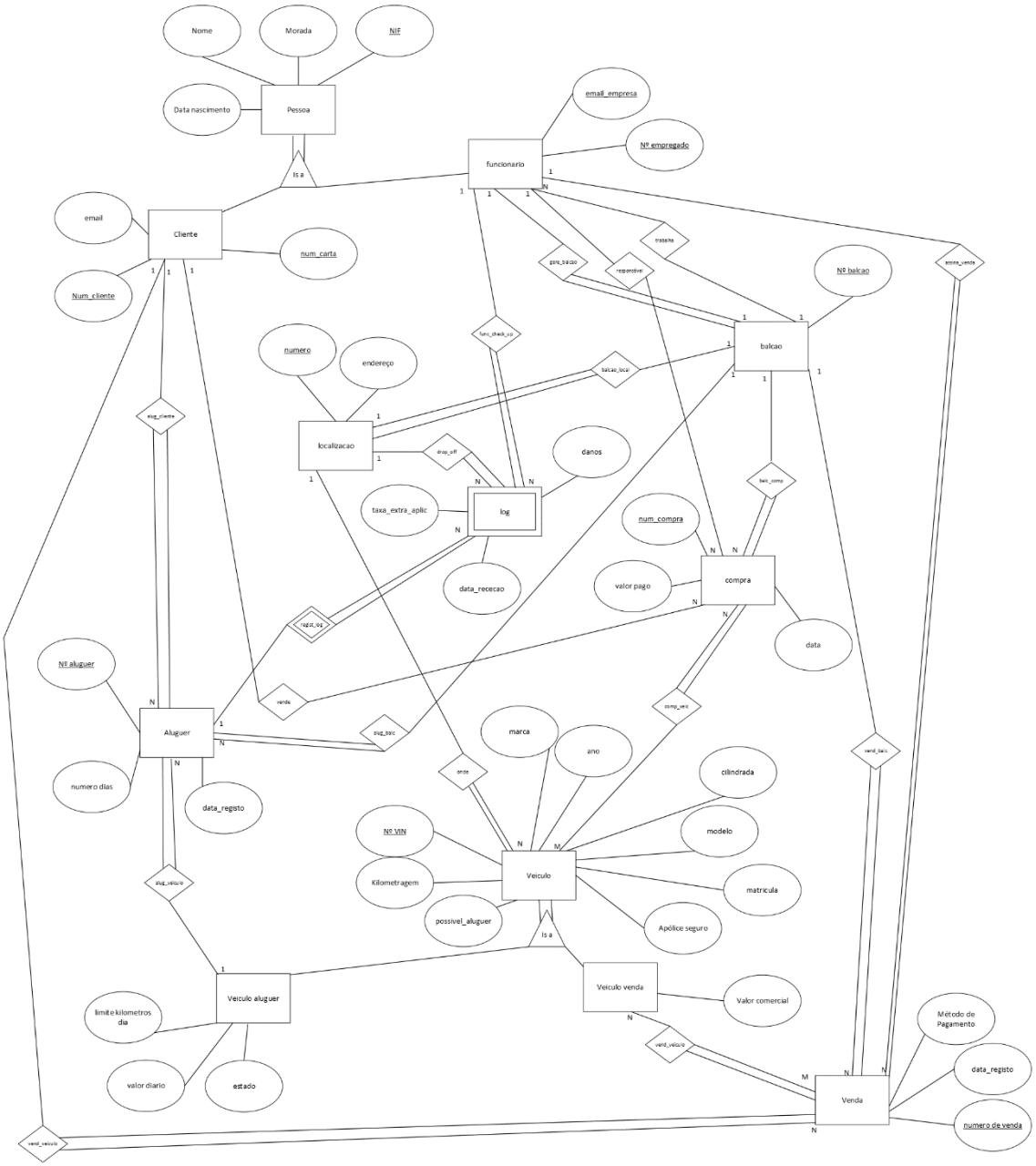

Figura 1 – Diagrama Entidade Relacionamento enviado por email após a reunião com as alterações sugeridas

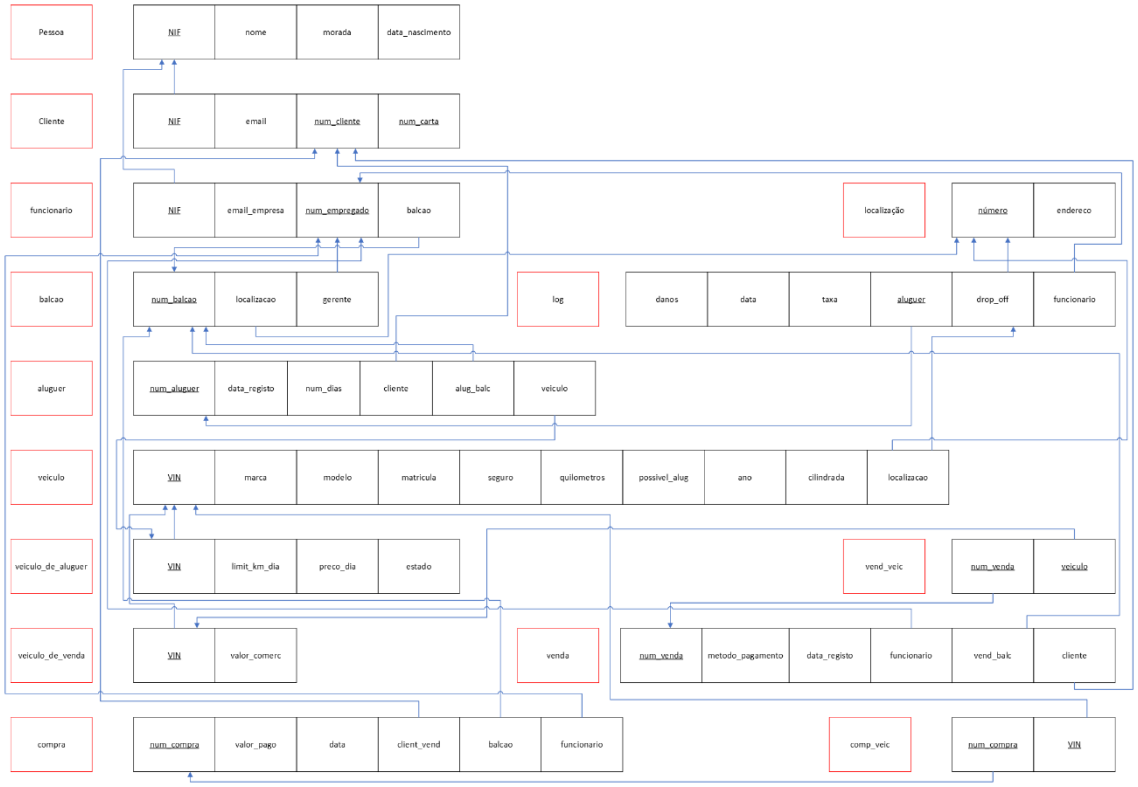

Figura 2 – Esquema Relacional enviado após a apresentação com as alterações efetuadas ao DER

As figuras encontram-se também apresentadas na pasta *figuras_relatorio* para uma visualização mais fácil e com maior resolução. Não sendo muito relevante devido a ter sofrido alterações que são apresentadas seguidamente.

Esquema Relacional na base de dados

Após o processo de evolução pelos diferentes esquemas chegou-se a um definitivo que se pode verificar na *Figura 3*. Este, no entanto, é diferente do enviado como correção uma vez que se verificou a existência e a possibilidade de simplificar entidades que não iriam fazer sentido, nomeadamente a junção de *Localização* com *Balcão*. Esta foi a lógica que fez mais sentido uma vez que os processos de interação iriam ser repetidos, pelo que um veículo se encontra num balcão por sua vez num local.

Pontos aqui a mencionar são nomeadamente a questão de a venda ser feita com base em *veiculo* e não em *veiculo_de_venda* isto devido à chave primária ser a mesma, assim como em *veiculo_de_aluguer*. Sendo apenas utilizadas para o acréscimo de informação que seja relevante somente para determinada ação, o aluguer ou a venda. A existência da tabela *veiculo_de_venda* é então questionável, no entanto, para a situação em que se realize maioritariamente alugueres não há necessidade da utilização de dados que seriam apenas relevantes para a sua venda, permitindo assim um desempenho maior pela redução do número de colunas. A justificação para a normalização encontra-se mais à frente, em

p_submit_log, por apresentar uma relação de dependência transitiva propositada numa situação em particular.

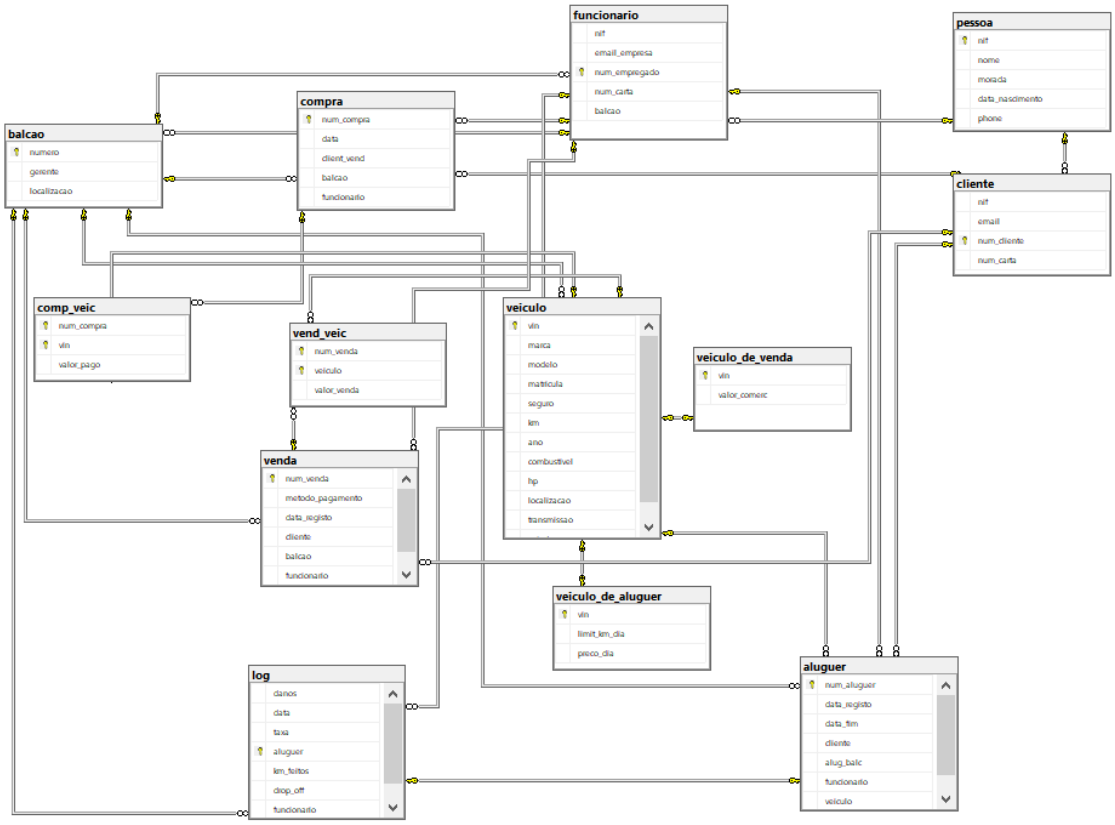

Figura 3 – Diagrama retirado de SQL Server com as ligações efetuadas entre as diferentes tabelas

Interface

Todo o código das interfaces foi desenvolvido em *Visual Basic*. Sendo apresentados 4 *forms* que incluem as funcionalidades que foram consideradas como essenciais. Tendo em vista a importância que é uma camada de abstração, tanto para o fornecimento de serviços assim como para segurança, a interação com a base de dados é maioritariamente feita pela utilização de *stored procedures*. Estas apresentam como vantagem já se encontrarem compiladas pelo que apresentam também um maior desempenho.

Em relação aos funcionários e à localização este conjunto deveria ter sido um aspeto a ter em consideração, nomeadamente a verificação de que se no balcão que é efetuado o registo é o balcão onde o funcionário trabalha. No entanto, poderá não ser errado para algumas estratégias de trabalho em que seja possível aos funcionários trabalharem em qualquer um dos balcões. Em adição faria mais sentido para o uso real a utilização de dados de *login* para cada um dos funcionários, resolvendo o ponto anterior colocado assim como evitando a colocação de um local de inserção do número de funcionário uma vez que isso estaria feito como *default*.

Não foram utilizados *indexes* uma vez que dada a escala do trabalho apresentado não o justifica. Contudo a inserção nos veículos para os seus estados, do tipo *clustered*, poderá ser desejada para tornar essa procura mais rápida uma vez que a venda de um veículo leva apenas à mudança de estado e não à sua remoção da lista.

Form1

Com esta página efetua-se o processo de aluguer, tanto o fornecimento do veículo assim como a sua devolução. No lado esquerdo é possível escolher um veículo utilizando-se filtros consoante se pretende uma determinada marca, modelo, combustível ou transmissão.

p_marcas

Esta é uma das *sp* utilizadas que segue a lógica de outras apresentadas para os outros parâmetros de escolha, modelo, ano, combustível[2], transmissão[3]. Em que se efetua a leitura para as *combobox* de apenas os restantes parâmetros que existem, por exemplo escolhendo a marca *Ford* só irão aparecer na *combobox* dos modelos os da marca, com o mesmo raciocínio para todos os parâmetros. Para isto são utilizadas variáveis globais em que a sua não seleção é considerada como *null*.

Não foi tido em consideração, no entanto a localização do veículo sendo apenas colocado para o ponto seguinte. Para a aplicação neste projeto, a criação dos filtros assim foi decidida por ser a opção mais fiável. No entanto, para larga escala consideramos que poderá ser um processo dispendioso estar sempre a efetuar a procura das restantes variáveis que apresentam veículos consoante a modificação pelo que será preferível deixar todas aparecerem e retornar uma lista nula para a sua combinação.

p_veiculos_aluguer

Faz uso da mesma lógica para o veículo utilizando todas as variáveis globais referidas para mostrar os veículos de aluguer. Com a adição de mostrar apenas os veículos numa dada localização. Uma vez estando no balcão 1 é mais provável pretender alugar um veículo que aí se encontre que um que esteja noutra localização.

[2] Este é um enumerado, em que 1 representa ser a gasolina, 2 a gasóleo, 3 elétrico. 4 GPL e 5 híbrido.
[3] Também é um enumerado em que 1 representa ser transmissão manual e 1 automática.

p_get_alug

Desta forma conseguem-se obter os alugueres que ainda não foram encerrados para mostrar na *listbox* da direita para se poder escolher qual se pretende encerrar. Ordenados pela data mais próxima de fim em cima, a inclusão de filtros ou de outros métodos de busca seriam uma adição que iria facilitar a procura. Podendo-se efetuar com apenas ligeiras modificações aos métodos anteriormente apresentados.

p_submit_log

Com este método pretende-se efetuar a conclusão do aluguer através da submissão de um ficheiro de *log*. No entanto, para facilidade de utilização e para fins mais práticos são feitas algumas alterações. É escrita a leitura do conta quilómetros que é apresentado agora, sendo introduzida a diferença de quilómetros no *log* para se ter a informação de quantos quilómetros foram realizados nesse aluguer. Após a introdução é calculado através de uma função, *func_price_to_pay*[4], o preço total que será pedido ao cliente. Este valor é substituído no ficheiro de *log* pelo valor da *taxa*, esta era introduzida como uma taxa que seria aplicada por danos ao veículo, sendo mais relevante armazenar o valor que foi arrecadado neste aluguer, colocando-se o veículo pronto a ser alugado para outra pessoa e na localização em que foi deixado.

Seria possível tornar este processo mais complexo introduzindo fases de limpeza do veículo e de manutenção, mas a lógica iria ser a mesma pelo que não foram aprofundadas. O cálculo do preço foi feito de forma arbitrária tendo em conta se o número de quilómetros percorridos é superior ao limite, assim como o número de dias, penalizando na situação de ser superior. Esta função deveria ser modificada consoante as normas na utilização.

Devido a esta dependência das colunas do *log* não se pode considerar que esteja para além da segunda forma normal. Apesar disso, isto foi feito intencionalmente de forma a poupar espaço e para ser mais interessante, continuando a verificarem-se muitos aspetos de formas normais de cardinalidade superiores, na 3ª forma normal para o restante conjunto.

Form2

Nesta interface pretende-se efetuar o processo de compra e de venda.

Apresentado à esquerda encontram-se os processos de venda de veículo que partem do mesmo princípio que muitos *sp* apresentados anteriormente no *Form1* para a filtragem, modificando para acrescentarem os veículos que se encontram disponíveis só para venda.

p_submit_venda

Para efetuar a venda de um automóvel o estado deste é alterado para 3[5], sendo inseridas as informações da venda na tabela *venda*, o método de pagamento utilizado, se às prestações ou uma vez só, a data da compra que é registada pelo sistema, o cliente que efetuou a compra, o balcão onde foi efetuada e o funcionário que registou a compra. No entanto, a tabela de *venda* cria a sua própria *primary key*, pelo que temos que a obter,

[4] Esta função foi criada para efetuar o processamento do valor a ser cobrado por aluguer. No entanto, a aritmética apresenta algumas falhas.

[5] Para o *estado* de um veículo foi decidido utilizar um enumerado, onde o estado 0 representa que o veículo esta disponível tanto para venda como para aluguer, o estado 1 que é apenas para venda, 2 que está a ser alugado neste momento e 3 que foi vendido.

utilizando os seus elementos para se poder inserir na tabela *vend_veic* devido à relação de N:M.

Ao efetuar-se a venda poderia fazer sentido apagar as linhas das tabelas de *veiculo_de_aluguer* e de *veiculo_de_venda*, no entanto foi decidido não se efetuar, estando essas linhas de código comentadas[6]. Também podendo eliminar-se na tabela *veiculo* a viatura vendida e introduzindo numa outra tabela de veículos vendidos com a alteração da chave de *vend_veic* para apontar para essa tabela.

p_submit_compra

Para efetuar a compra é primeiramente verificado se este veículo já tinha feito parte dos veículos da base de dados, caso não tenha sido, é inserido na tabela *veiculo* os seus dados e inserido nas tabelas de *veiculo_de_aluguer*, caso seja um veículo para efetuar o aluguer, e em *veiculo_de_venda*. Caso já tivesse estado na empresa, é então feito a sua atualização para o estado em que agora se encontra, atualização dos valores de aluguer e dos parâmetros da viatura, por exemplo da sua quilometragem atual.

Este procedimento apresenta uma grande dependência do anteriormente apresentado uma vez que se o anterior apaga algo este teria que inserir de novo ou fazer a passagem de dados de um veiculo de uma tabela de estado *vendido* para a nova.

Form3

Este *form* apresenta-se como funcionalidade para a gestão de pessoal, levando ao registo de clientes e de funcionários que apresentam parâmetros iguais que necessitam de ser preenchidos. Também é facultada a hipótese de alterar dados no funcionário assim como a sua demissão.

p_new_cliente

Previamente à criação de um novo cliente é verificada se já existe esta pessoa, um funcionário poderá eventualmente ser um cliente e por isso já se encontrar registado como pessoa faltando apenas os dados de cliente. Caso não seja esse o caso é registado como pessoa sendo acrescentados os parâmetros de clientes a ambos. A mesma lógica é aplicada ao caso dos funcionários. Sendo previamente verificado se esta pessoa já tinha sido apagada do sistema, ou seja, guardada numa tabela à parte, em que é removida dessa tabela.

p_demitir_func

Esta *procedure* efetua o despedimento de um dado funcionário, inserindo os seus dados numa tabela secundária de pessoas removidas e de funcionários despedidos, removendo-o das tabelas de dados ativos.

p_update_func

A atualização de dados faz sentido neste contexto pelo que é permitido efetuar-se ao funcionário, para o cliente seria igual pelo que se torna irrelevante para este contexto.

Form4

Foi escolhida a inclusão desta página para se poderem estudar *procedures* mais relevantes que poderiam ser aplicados para a obtenção de alguns dados estatísticos que fossem ser potencialmente relevantes para a administração e para a gestão da empresa. Os

[6] No ficheiro *procedures.sql*

métodos apresentados não são muito vastos tendo sido apenas concretizados os que foram sentidos como os que seriam potencialmente mais relevantes para fins práticos.

p_lucro_cliente

Desta forma é possível analisarem-se os clientes consoante o valor que gastam. Permitindo que haja uma outra gestão, seja para um tratamento VIP ou para a passagem a um plano de cliente de categoria mais elevada. Assim como em *p_num_alug_clientes* o número de alugueres que cada cliente já efetuou.

p_func_vendas

A uma empresa será muito útil verificar qual o número de vendas assim como o lucro que foi arrecadado para a verificação de se um funcionário está a corresponder às cotas mensais necessárias, sendo possível analisar em detalhe para um determinado balcão.

p_balcao_veiculo_alug

Verificando qual o modelo que é usualmente mais utilizado num determinado balcão permite uma melhor alocação de veículos, ao verificar-se que um determinado tipo de viatura é mais requisitado num local que outro, a troca de localização poderá ser desejável.

Triggers

Para aumentar a consistência da base de dados foi verificado que a inserção de *triggers* seria uma vantagem apesar dos custos extras associados. Foram utilizados na maioria com *after* devido a servirem como verificação de processos ou como adição a um processo normal de inserção, não apresentando vantagem a serem usados como *instead of*.

t_log

A inserção do ficheiro de *log* obriga sempre à verificação de que se o número de quilómetros agora apresentado é superior ao que estava antes, caso fosse menor haveria um aviso de adulteração no veículo.

Este seria um *trigger* que seria possível e recomendado para usar na modalidade de *instead of* em que se efetuava primeiramente a verificação dos quilómetros efetuados, colocando o valor como o que se desejaria inserir, os quilómetros realizados neste aluguer, voltando atrás caso fosse invalido, efetuando a inserção logo corretamente com todos os outros valores diminuindo se o custo com a atualização.

t_veiculo_alug

Este método é utilizado para verificar que uma viatura exclusiva a venda não é inserida por engano como veículo de aluguer. Não sendo necessário nesta aplicação uma vez que os procedimentos são efetuados por nós, no entanto, serve de acréscimo para a flexibilidade para o eventual uso e exploração por parte de outros desenvolvedores.

t_venda

De forma a colocar no estado de *vendido* após a sua venda, garantindo-se assim a sua não participação como sendo um veículo disponível para aluguer. Para mais uma vez, permitir uma maior flexibilidade.

t_aluguer

A verificação da data prevista para entrega do veículo ser após a data de registo do aluguer. Assim como a colocação da viatura num estado de *alugado* para não permitir que seja mostrado como disponível nem para ser alugado ou vendido neste momento.

t_idade

Apenas maiores de idade poderão efetuar o seu registo, seja tanto para trabalhador como para cliente.

t_funcionario

Um funcionário necessita de estar associado a um balcão que exista ou que esteja registado pelo que se efetua a sua verificação.

t_transmicao_combustivel

Uma vez que todos os carros elétricos ou híbridos não são possíveis de adquirir com uma caixa de velocidades manual, é constatado se ocorre esse lapso por parte do utilizador, corrigindo caso fosse uma inserção incorreta emitindo um alerta, não cancelando a transação uma vez que é possível corrigir automaticamente.

Considerações finais

Tendo em vista o ambiente educativo em que se insere a motivação deste trabalho, a criação de uma base de dados e a sua manipulação através de uma camada de abstração com o uso de uma interface gráfica para efetuar o mesmo, considera-se que foi respondido com sucesso no seu intrínseco. Poderiam ter sido feitas mais algumas interações com a base de dados, contudo estes iriam ser métodos que iriam fazer uso de lógicas muito semelhantes e que não iriam trazer nada de novo ou diferente ao já apresentado.

Para a aplicação num mundo empresarial iria ser necessário tornar a interface gráfica mais *user friendly* assim como a sua consistência para mitigar quaisquer erros que possam surgir na combinação de manipulações que iriam ser possíveis de estudar n a avaliação *beta* da aplicação. Ainda assim para uma empresa de baixa escala que pretendesse digitalizar a sua operação este projeto poderia responder a essas necessidades. Verificando-se a aplicação de toda a matéria desenvolvida no desenrolar da unidade curricular.