Relatório de Estágio

Rúben Silvério Figueiredo Rodrigues

Nº130221002

Escola Superior de Tecnologia de Setúbal - Instituto Politécnico de Setúbal

Mestrado em Informática de Gestão

Orientador EST – Nuno Pina

Júri – Cláudio Sapateiro

Júri – José Sena Pereira

Orientador RIGHTIT – Nuno Cardoso

Novembro de 2017

Agradecimentos

Antes de iniciar este relatório, não posso deixar de agradecer a todos aqueles que me ajudaram a concluir com sucesso esta fase do meu percurso académico.

Em primeiro lugar, quero agradecer à Right IT Services pela oportunidade que me proporcionam e por todo o apoio que recebi durante o meu estágio. Agradeço em especial ao meu orientador na empresa e grande *Team Leader*, Nuno Cardoso, pelo acompanhamento ao longo do meu trabalho e pelos valores profissionais que me transmitiu. Aproveito esta ocasião para também agradecer à restante equipa Right IT Services pelo apoio que me deram e por sempre se esforçarem por me fazer sentir parte da família. A vossa ajuda foi indispensável para o meu percurso. Nomeadamente Ralfe Elias, Diogo Antunes, Miguel Duarte, Hugo Pinto, João Reis, Pedro Pires, entre outros.

Agradeço também ao meu orientador de estágio, o professor Nuno Pina pela disponibilidade para ser meu orientador e pelo apoio ao longo do percurso.

Agradeço à minha família pelo seu apoio ao longo desta etapa, não só de estágio, mas por toda a carreira académica. Foi o vosso contributo que tornou tudo isto possível e por isso só tenho a agradecer.

Índice

Agradecimentos 2
Índice 3
Introdução 6
Apresentação Right IT Services 7
História 7
Quem somos e o que fazemos? 7
Missão: 7
O nosso ADN: 8
Objetivo: 8
Compromissos: 8
Fundamento Teórico 9
Contexto do Projeto 9
Salesforce 9
Tecnologias utilizadas 10
Force App Cloud.com 10
Apex 10
Visualforce 11
Lightning 11
JIRA 12
Trello 12
Bitbucket 13
Bamboo 14
HPI Webservice 14
Drawloop 14

Ambientes de desenvolvimento e de integração 14
Ambiente desenvolvimento (Sandbox) 15
Ambiente de Integração 15
Conceitos base a ter em conta 16
Objetos 16
Objetos Customizados 16
Salesforce Object Query Language (SOQL) e Salesforce Object Search Language (SOSL) 17
Relações 18
Triggers 19
Perfis (Profiles) 20
Segurança ao nível dos campos (Field-Level Security) 20
Programação em Apex 21
Código bulkified 21
Evitar consultas SOQL ou declarações DML dentro de ciclos For 22
Usando coleções, simplificação de consultas e For Loops eficientes 23
Como fazer consultas a grandes conjuntos de dados 23
Escrever testes para verificar grandes quantidades de conjunto de dados 24
Evitar hardocding Ids 24
Data Access Object (DAO) 24
Descrição Trabalho Realizado 25
Metodologia de desenvolvimento 25
CPQUserJourney 27
Módulo 1 - CPQUserJourneyVehicleInformation 32
Módulo 2 - CPQUserJourneyQuoteRates 36
Módulo 3 - CPQUserJourneyCustomerInfo 37
Módulo 4 - CPQUserJourneyPurchaseQuote 39

Módulo 5 - CPQUserJourneyContractPurchased 41
Cronograma geral do Projeto 43
Resultados e Discussão 44
Webgrafia.....47

Introdução

O presente relatório foi elaborado pelo aluno Rúben Silvério Figueiredo Rodrigues, número 150283011, a frequentar o segundo ano do mestrado em Informática de Gestão no Instituto Politécnico de Setúbal (IPS). Destina-se à unidade curricular Estágio/Projeto e tem como objetivo a transmissão do projeto desenvolvido durante o tempo de estágio e os seus resultados.

O estágio foi realizado na empresa Right IT Services, sediada em Lisboa, Portugal. Teve início a 02 de Fevereiro de 2017.

O projeto realizado consistiu no desenvolvimento de uma aplicação CRM.

Como referido, com este relatório é pretendido apresentar o projeto realizado e explicar o processo de desenvolvimento do mesmo. Começará por introduzir ao leitor a empresa Right IT Services, onde foi realizado o estágio, passando depois para uma descrição dos fundamentos teóricos do trabalho, contextualizando o leitor com o projeto e os seus componentes. Seguidamente, será descrito em mais detalhe o desenvolvimento do projeto, dividido em módulos para facilitar a compreensão.

Apresentação Right IT Services

História

A RIGHT IT Services é uma empresa Portuguesa sediada em Lisboa com mais de 10 anos de experiência na implementação de soluções globais de IT na Cloud. É uma empresa de base tecnológica, cujo *core business* incide sobre o desenvolvimento e implementação de software, com ênfase em CRM sobretudo em soluções baseadas na tecnologia Salesforce.com, tendo os seus clientes e Projetos espalhados pela Europa.

Figura 1 - Logotipo da Right IT Services

Quem somos e o que fazemos?

A Right IT Services garante soluções de IT, arquitetura de serviços, design e desenvolvimento de software e respetivos testes.

Missão:

Ajudar os nossos Clientes a maximizar o valor do seu negócio através de Inovação Tecnológica.

O nosso ADN:

Colocar os Clientes em primeiro lugar, investir na nossa equipa e recursos, gerar ideias e inovação, para garantir o sucesso e a constante evolução do seu Negócio.

Objetivo:

Ajudar os seus Clientes - A Right IT Services coloca ao dispor dos seus Clientes as competências que tem desenvolvido continuamente, possibilitando soluções à medida com custos reduzidos e segurança da informação, garantida por um conjunto de políticas, tecnologias e controlos ativados na *Cloud* para proteção dos dados. As soluções são desenvolvidas sob as melhores práticas de gestão de projetos de Tecnologias de Informação, o que permite aos seus Clientes minimizar os riscos inerentes aos investimentos em tecnologia e maximizar os benefícios das soluções tecnológicas.

Compromissos:

Entregar a melhor solução, com base na experiência, analisando os requisitos dentro do contexto das necessidades do Cliente;

Fornecer ao Cliente a informação certa, de forma a que tome melhores decisões;

Acrescentar Valor ao Cliente para garantir um Projeto de Sucesso;

Garantir transparência na relação com os Clientes e Parceiros.

Fundamento Teórico

Neste tópico será introduzido ao leitor o projeto desenvolvido no tempo de estágio de uma forma mais detalhada, apresentando os conceitos teóricos na sua base e os seus objetivos.

Contexto do Projeto

O projeto realizado consistiu no desenvolvimento de uma plataforma para uma empresa do Reino Unido que trata de seguros automóveis desenvolvida através da *Salesforce* com o objetivo de agilizar o processo *CPQ* (*Costumize*, *Price*, *Quote*). Adaptando as ferramentas de negócio deste cliente, passando de um sistema *legacy*, para uma plataforma mais robusta e moderna na Cloud.

Salesforce

Salesforce é uma empresa americana, fundada por Marc Benioff com sede em São Francisco que produz software *on demand* (software a pedido). Ganhou enorme notoriedade por ter produzido o *CRM* chamado *Sales Cloud*. O Fundador, revolucionou o mercado de CRM, criando uma solução Cloud.

Além desta solução, a empresa conta com outros produtos com foco em atendimento ao cliente, marketing, inteligência artificial, gestão de comunidades, criação de aplicativos entre outras frentes. As soluções da Salesforce são indicadas para PMEs (pequenas e médias empresas) e também para grandes corporações, de salientar que este CRM é dos mais poderosos disponíveis no mercado.

Figura 2 Logótipo da Salesforce

Tecnologias utilizadas

Force App Cloud.com

Force App Cloud.com é uma Plataforma como Serviço (*PaaS*), um ambiente completo de desenvolvimento e implementação na *cloud*, que permite aos programadores criar ***multitenants*** (uma única instância de software que é executada em um servidor e serve vários "inquilinos").

A Salesforce funciona em modelo de três camadas denominado de MVC (Model-View-Controller)

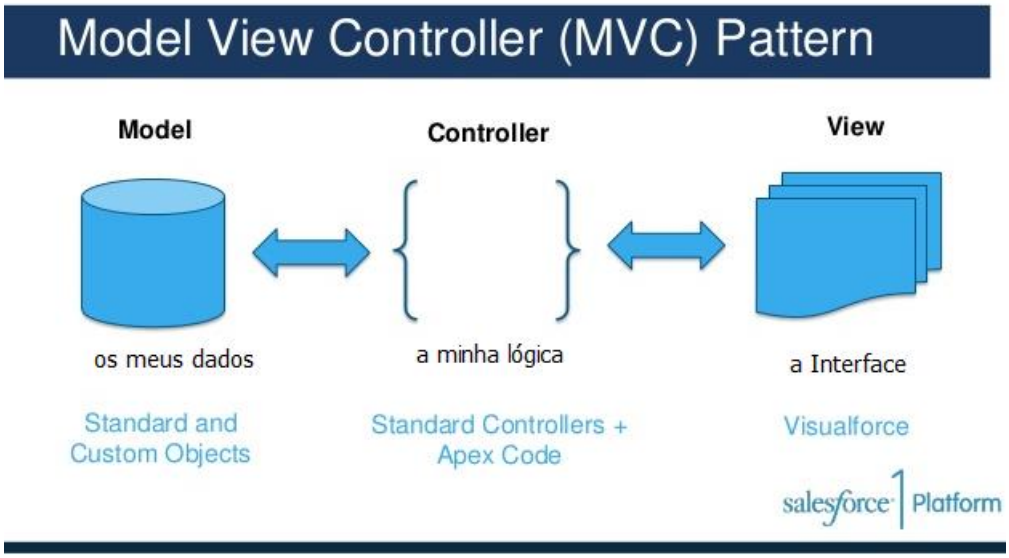


Figura 3 - Ilustração do Modelo MVC

Apex

Esta linguagem é fornecida pela plataforma ***Force.com*** para os programadores desenvolverem as suas aplicações, bastante semelhante ao **JAVA** e **C#**, onde estive em contato durante grande parte do percurso académico. Apex é uma linguagem de programação com tipologia forte e tal como o JAVA é orientada a objetos. Usando uma sintaxe parecida com Java que funciona de maneira semelhante aos procedimentos armazenados de banco de dados, o Apex permite que os programadores adicionem a lógica de negócios à maioria dos eventos do sistema,

incluindo cliques de botão, atualizações de registos relacionados e páginas ***Visualforce***.

Visualforce

Visualforce é a tecnologia de exibição de controle no **Force.com**. Isto é, o **front-end** com o qual os utilizadores interagem. É semelhante ao **HTML** e a maioria das anotações HTML podem ser usadas nas codificações das páginas **Visualforce**. Tal como no HTML, as páginas VF usam pares de componentes <> </> para definir componentes de layout e também para definir atributos.

Tanto o Apex e as páginas VisualForce "correm" na plataforma Force.com e não num servidor Salesforce.

Lightning

Em 2014, a Salesforce tornou público o *front-end* da sua plataforma, chamado Lightning. Esta estrutura baseada em componentes serve de base para as aplicações móvel Salesforce1, os clientes agora podem construir sobre isso também. O *Salesforce* foi construído nesta estrutura em 2015 ao lançar o *Lightning Design System*, uma estrutura constituída por HTML com estilo CSS padrão incorporado. Essa estrutura permite que os clientes criem seus próprios componentes para usar em suas instâncias internas ou vender no ***AppExchange***[1].

AppExchange[1] – é o mercado da Salesforce onde os utilizadores podem aceder para fazerem download e instalar aplicativos de software para serem usadas na plataforma Force.com e também pode ser usado para procurar consultores da cloud.

Uma das novas ferramentas lançadas é conhecida como o *Salesforce Lightning App Builder* para o desenvolvimento rápido de aplicações de interfaces web responsivas. Esta interface permite que diferentes telas sejam montadas com base em componentes *Lightning*. Isso pode ser usado como layouts para registos ou aplicativos específicos.

JIRA

O JIRA é uma ferramenta de gestão de projeto e para acompanhamento das tarefas e/ou erros (bugs) desenvolvido pela empresa *Australiana Atlassian.*

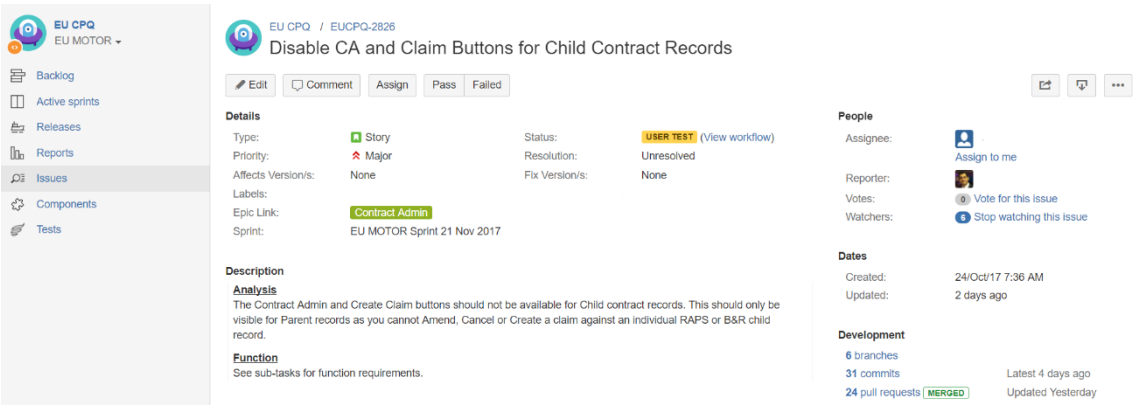


Figura 4 - Aspeto visual do JIRA

Trello

Outra ferramenta de colaboração utilizada para que todos possam acompanhar e coordenar o trabalho de todos, desde o Gestor do Projeto ao programador, Trello é uma aplicação de gestão de projeto baseado na web originalmente feito por Fog Creek Software em 2011. Trello utiliza o paradigma Kanban para gestão de projetos.

Os projetos são representados por *boards* (quadros), que contêm listas de tarefas. Cartões estes, que podem ser movidos de uma lista para outra para representar o progresso da tarefa sendo que os utilizadores podem ser inscritos nos cartões.

Na Figura 5 (alguns nomes foram removidos por motivos sigiloso), é possível ver no meu cartão (Rúben Rodrigues) que na semana decorrente de 4 de Abril de 2017 estive a trabalhar (até ao momento) em 2 *tickets* (EUCPQ-1287 e EUCPQ-1487). São usadas cores para representar o estado do ticket, sendo o Amarelo para quando se está a desenvolver a tarefa, Verde para concluído e Vermelho quando estamos bloqueados com uma outra tarefa.

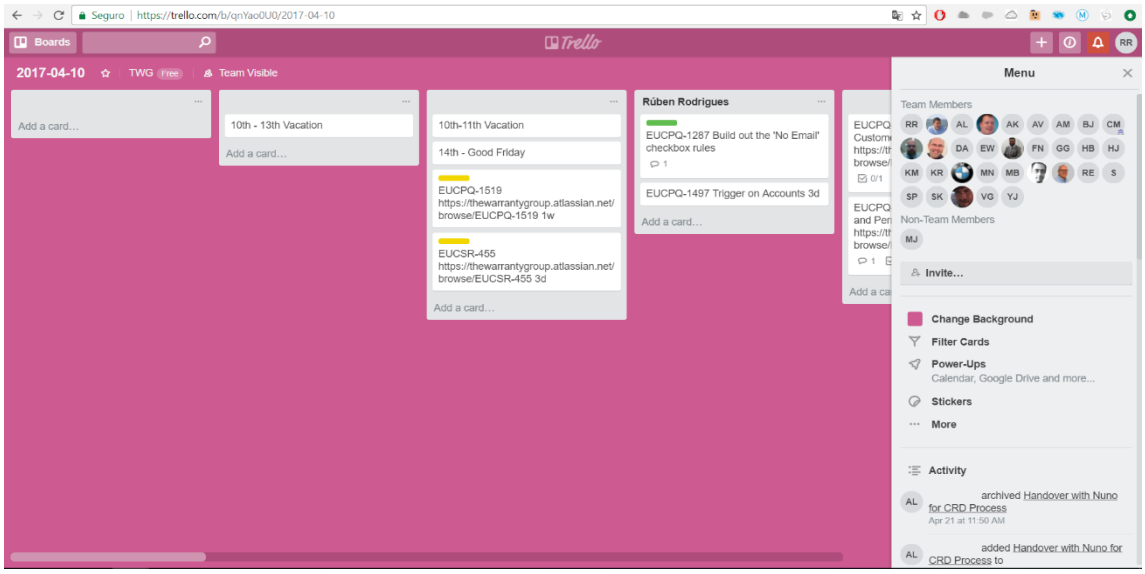


Figura 5 - Ilustração de uma das semanas no Trello

Bitbucket

O Bitbucket é um repositório de projetos controlado através do Mercurial, um sistema que serve para controlo de versões distribuído, sistema esse escrito em Python. O Bitbucket também suporta repositórios usando o sistema de controlo de versões Git. Está interligado ao Jira, o que facilita a ligação das "*user stories*" e o git.

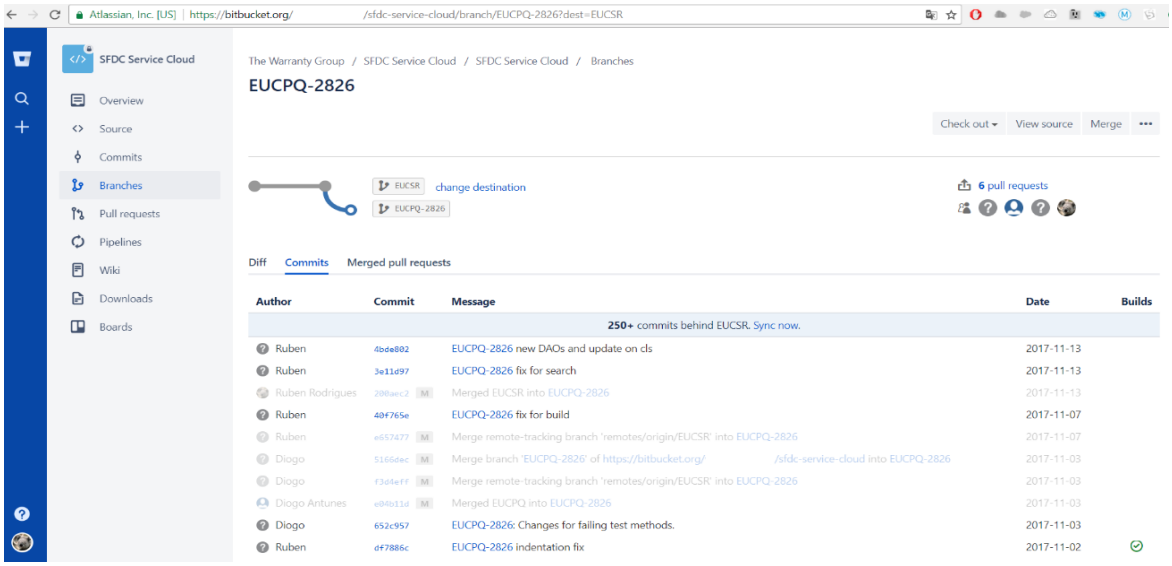


Figura 6 - Ilustração da interface do Bitbucket

Bamboo

Ferramenta contínua de integração utilizada para fazer os *deploys* (transição de software para outros ambientes).

HPI Webservice

HPI é um **Webservice** que retorna dados da HPI (*Hire Purchase inspection*) referentes a carros do Reino Unido. Este Webservice utiliza a arquitetura REST (*Representational State Transfer*), os formatos suportados do ***Request*** (Pedido) são SOAP (*Simple Object Access Protocol*) e XML (*eXtensible Marcus Language)*, e os de ***Response*** (Resposta) são o JSON, XML e SOAP.

Drawloop

O D*rawloop* é uma Plataforma para criação e automação de documentos na nuvem utilizado para a geração de documentos, fundindo os dados da Salesforce e automatização de documentos dinâmicos através da combinação do Microsoft Word, PowerPoint e outros.

Ambientes de desenvolvimento e de integração

Na terminologia Force.com, "ambientes" e "organizações" são sinónimos. Isto é, uma instância da plataforma Force.com é um ambiente e também uma organização, que na maioria das vezes se chamam apenas por "org". Existindo, portanto, vários tipos de ambientes: a org de produção, onde se encontra o produto final, ambiente de desenvolvimento onde cada programador desenvolve e realiza as tarefas e os devidos testes unitários do código, num ambiente isolado antes de serem passadas para os sistemas de integração onde serão testadas as funcionalidades. Podendo ser dado nomes as estes ambientes. Neste projeto o ambiente de desenvolvimento tinha como nome UKDEV, e os outros dois ambientes de integração CSRINT e CPMINT.

Ambiente desenvolvimento (*Sandbox*)

O termo *sandbox* é frequentemente usado para o desenvolvimento de serviços da Web para se referir a um ambiente réplica de produção para uso de desenvolvedores.

Normalmente, um programador desenvolve o seu código de acordo com a tarefa, que mais tarde irá ser validado, antes de migrá-lo para o ambiente de integração.

Neste projeto, a sandbox de desenvolvimento era a UKDEV, e as de integração eram a CSRINT e CPMINT.

Ambiente de Integração

Nestes ambientes (CPMINT e CSRINT) são efetuados os testes de sistema, das tarefas que migraram do ambiente de desenvolvimento (DEV), por parte da equipa de *testers.* Antes de ser migrado o código para produção é testado num ambiente réplica de produção, em QA (*Quality Assurance*).

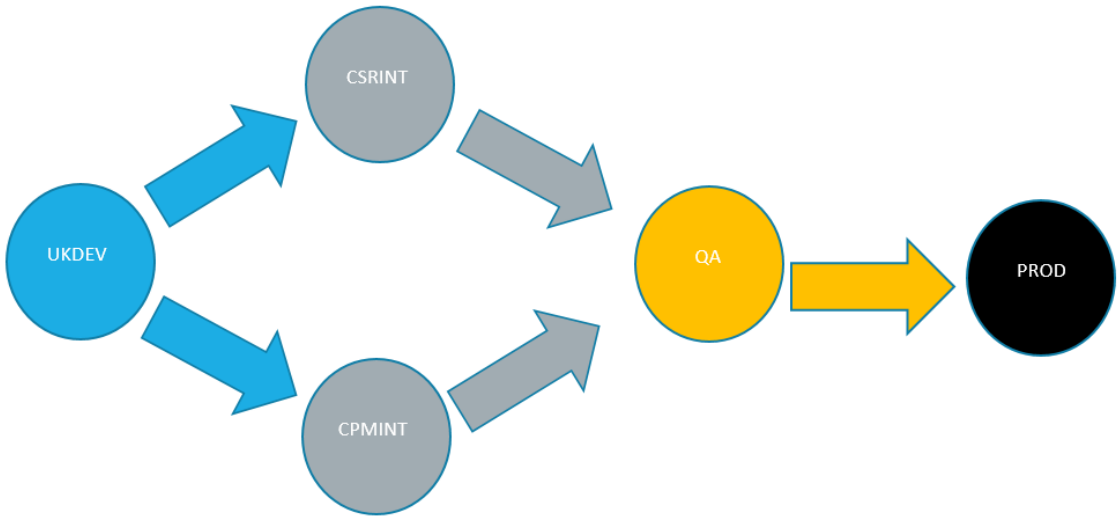


Figura 7 - Ilustração do processo dos ambientes desde o desenvolvimento até produção

Conceitos base a ter em conta

Objetos

De um modo geral, os objetos API (*Application Programming* Interface) representam tabelas na base de dados que contêm a informação da organização. Por exemplo, o objeto central no modelo de dados do Salesforce representa contas (*Accounts*) - empresas e organizações envolvidas com a empresa, como clientes, parceiros e concorrentes. O termo "registos" (*record*) descreve uma ocorrência específica de um objeto (como uma conta específica como "IBM" ou "United Airlines" que é representada por um objeto Conta). Um registo é análogo a uma linha em uma tabela de banco de dados.

Os objetos já criados por defeito pelo *Salesforce* são chamados de objetos padrão ou *Standard Objects*. Os novos objetos criados na organização são chamados de objetos customizados ou *Custom Objects*. O acesso aos objetos, sendo standard ou customizados, pode variar consoante o nível de configuração definido, tanto ao nível da configuração da organização, ao nível das permissões do utilizador, ao modelo de partilha de dados (*Sharing Model*).

Objetos Customizados

Objetos customizados são tabelas de base de dados personalizadas que permitem armazenar informações exclusivas para a organização.

Aplicativos de cliente com permissões suficientes podem invocar chamadas de API em objetos customizados existentes. É possível criar objetos customizados com a interface do utilizador ou usando o metadata WSDL (*Web Services Description Language)* com um aplicativo cliente ou ainda, usando o IDE Force.com.

Na API, os nomes de objetos customizados incluem um sufixo de dois sublinhados *(underscores)*, seguido de um "c" minúsculo. Por exemplo, um objeto customizado denominado "Contract" na interface do utilizador do Salesforce é Contract__c no WSDL da organização.

Salesforce Object Query Language (SOQL) e Salesforce Object Search Language (SOSL)

Uma consulta (*query*) SOQL é equivalente a uma instrução *SELECT* SQL (*Structured Query Language*) que serve de pesquisa à base de dados da org.

Exemplo: *SELECT Name FROM Account*. Nesta consulta estou a ir buscar todos os nomes do objeto *Account* existentes na base de dados.

O SOSL é uma maneira programática de realizar uma pesquisa baseada em texto em relação ao índice de pesquisa.

Exemplo: *FIND {John Smith} IN Name Fields RETURNING Account (name, phone)*. Aqui estou a procurar o nome John Smith no campo *name* em uma Account ou várias retornando apenas os campos *name* e *phone*.

Quando usar SOQL ou SOSL?

As consultas SOQL devem ser usadas quando se sabe em que objeto se encontram os dados e queremos retornar:

- Dados de um único objeto ou de vários objetos que estão relacionados um ao outro

- Contar o número de registos que vão ao encontro dos critérios especificados

- Ordenar resultados como parte da pesquisa; retornar dados de campos como, *number, date* e *checkbox.*

Usar o SOSL quando não se sabe em qual objeto ou campo os dados se encontram, e queremos:

- Recuperar dados para um termo específico que sabemos que existe dentro de um campo. Como o SOSL pode "tokenizar" vários termos dentro de um campo e criar um índice de pesquisa a partir deste, as buscas SOSL são mais rápidas e podem retornar resultados mais relevantes.

- Recuperar vários objetos e campos de forma eficiente onde os objetos podem ou não estar relacionados um com o outro.

Relações

Relações associam objetos com outros objetos. Por exemplo, a relação pode vincular a um objeto customizado a um objeto standard em uma lista relacionada, como a ligação a um objeto customizado chamado *Bugs* para *Cases* para rastrear defeitos de produtos associados a *Cases* de clientes.

Tipos de relações

Master-detail (1:n)

Uma relação pai-filho em que o objeto master controla certos comportamentos do objeto do *detail*:

- Quando um registo do objeto master é excluído, os registos do *detail* relacionados também são excluídos.

- O campo *owner* no objeto de *detail* não está disponível e é definido automaticamente para o proprietário(*owner*) do seu registo *master* associado. Objetos customizados no lado do *detail* de uma relação *master-detail* não podem ter regras de compartilhamento (*sharing rules)*, compartilhamento manual ou filas, pois estes requerem o campo Proprietário.

O registo do *detail* herda as configurações de compartilhamento e segurança do seu registo *master*.

Many-to-Many

É possível usar as relações *master-detail* para modelar relações de muito-para-muitos entre quaisquer dois objetos. Esta relação permite que cada registo de um dado objeto esteja ligado a vários registos de outro objeto e vice-versa. Por exemplo, cria-se um objeto customizado chamado "Bug" que se relaciona com o objeto de *Case standard*, de modo que um erro(*bug*) possa estar relacionado a vários casos(*cases*) e um caso(*case*) também pode estar relacionado a múltiplos erros(*bugs*).

Lookup (1:n)

Este tipo de relação liga dois objetos, mas não tem efeito sobre exclusão ou segurança. Ao contrário dos campos de *master-detail*, os campos de *lookup* não são necessários. Quando se define uma relação de *lookup*, os dados de um objeto podem aparecer como uma lista relacionada em layouts de página para o outro objeto.

As relações alteram a convenção de nomenclatura. Por exemplo, no caso de objetos customizados: Quote_Dealer_Product__r.Dealer_Product__c, acrescenta-se dois sublinhados (*underscores)* e um "r" seguido de "." para o campo pretendido

A tabela a seguir resume se um objeto *standard* pode ser:

- O *master* em uma relação *master-detail* com um objeto customizados. Relações *master-detail* envolvem exclusões em cascata e regras de compartilhamento que o objeto "pai" controla.

- Se um objeto customizado pode ter uma pesquisa(*lookup*) para o objeto *standard*.

- Estendido com campos customizados.

Standard Object	Master-Detail	Lookup	Custom Fields
Account	Yes	Yes	Yes
Campaign	Yes	Yes	Yes
Case	Yes	Yes	Yes
Contact	Yes	Yes	Yes
Contract	Yes	Yes	Yes
Event	No	No	Yes
Lead	No	No	Yes
Opportunity	Yes	Yes	Yes
Solution	Yes	Yes	Yes
Task	No	No	Yes
User	No	Yes	Yes

Tabela 1- Tabela que define que tipo de relações um objeto standard pode ter e se permite campos customizados

Triggers

Apex pode ser invocado usando *triggers*. Os *Apex triggers* permitem executar ações personalizadas antes ou depois de alterações nos registos do *Salesforce*, como inserções, atualizações ou exclusões. Um disparador é o código *Apex* que executa

antes ou depois dos seguintes tipos de operações: *insert, update, delete, merge, upsert, undelete.*

Por exemplo, é possível ter um *trigger que* executa antes que os registos de um objeto sejam inseridos na base de dados, depois que os registos sejam excluídos, ou mesmo após a gravação ser recuperado da Lixeira (*Recycle Bin*).

Existem dois tipos de *triggers:*

Before Triggers são usados para atualizar ou validar valores de registo antes de serem salvos na base de dados.

After Triggers são usados para aceder aos valores dos campos definidos pelo sistema (como o campo *Id* ou *LastModifiedDate* de um registo) e afetar as alterações em outros registos, como "disparar" eventos assíncronos com uma *queue* (fila).

Perfis (Profiles)

Os perfis definem como os utilizadores acedem a objetos e dados, e o que eles podem fazer dentro da aplicação. Quando se cria utilizadores, atribui-se um perfil a cada um. Também é possível criar perfis customizados assim como alterar os existentes na Org.

Segurança ao nível dos campos (Field-Level Security)

As configurações ao nível dos campos permite, restringir os acessos de visibilidade e de edição dos campos aos utilizadores.

Os campos que os utilizadores veem nas páginas são uma combinação de layouts de página e configurações de segurança ao nível dos campos. As configurações de acesso de campo mais restritivas dos dois sempre se aplicam. Por exemplo, pode existir um campo que é necessário em um *layout* de página, mas é apenas de leitura nas configurações de segurança de nível de campo. A segurança no nível de campo substitui o layout da página, então o campo permanece somente leitura.

É possível definir segurança ao nível dos campos de dois modos:

- Para múltiplos campos num único *Permissions Set* ou por Perfil

- Para um único campo em vários Perfis

Programação em Apex

Alguns aspetos importantes a ter em conta quando se programa em Apex na plataforma Force.com. Tal como acontece com qualquer linguagem de programação, existem, portanto, princípios e práticas recomendas para a codificação, que ajudam a escrever um código não só eficiente como também escalável.

Código *bulkified*

Refere-se ao conceito de garantir que o código lida corretamente com mais de um registo por vez. Quando um lote de registos inicia Apex, uma única instância desse código Apex é executada, mas precisa lidar com todos os registos nesse lote dado. Por exemplo, um *trigger* poderia ser invocado por uma chamada Force.com SOAP API que inseriu um lote de registos. Então, se um lote de registos invoca o mesmo código Apex, todos esses registos precisam ser processados como um volume, para escrever código escalável e evitar bater nos limites do governador.

Aqui está um exemplo de código mal escrito que apenas lida com um registo:

```
trigger accountTestTrggr on Account (before insert, before update) {

    //This only handles the first record in the Trigger.new collection
    //But if more than one Account initiated this trigger, those additional records
    //will not be processed
    Account acct = Trigger.new[0];
    List<Contact> contacts = [select id, salutation, firstname, lastname, email
                 from Contact where accountId = :acct.Id];

}
```

Figura 8 - Ilustração de código não bulkified

A questão é que apenas um registo da *Account* é tratado porque o código acede apenas ao primeiro registo na coleção Trigger.new usando a sintaxe Trigger.new [0]. Em vez disso, o *trigger* deve lidar adequadamente com toda a coleção de *Accounts* na coleção Trigger.new (Figura 9).

```
trigger accountTestTrggr on Account (before insert, before update) {

   List<String> accountNames = new List<String>{};

   //Loop through all records in the Trigger.new collection
   for(Account a: Trigger.new){
      //Concatenate the Name and billingState into the Description field
      a.Description = a.Name + ':' + a.BillingState
   }

}
```

Figura 9 - Ilustração de código bulkified

Evitar consultas SOQL ou declarações DML dentro de ciclos For

Em Salesforce existe um limite regulador (*governor limits*) que impõe um determinado número máximo de instruções DML (*insert, update, delete, undelete*). Quando essas operações são colocadas dentro de um *loop for*, as operações da base de dados são invocadas uma vez por iteração do *loop*, tornando muito fácil alcançar esses limites. Para evitar isto, deve se fazer qualquer operação DML antes dos ciclos *for.*

```
trigger accountTestTrggr on Account (before insert, before update) {
  //This queries all Contacts related to the incoming Account records in a single SOQL query.
  //This is also an example of how to use child relationships in SOQL
  List<Account> accountsWithContacts = [select id, name, (select id, salutation, description,
                                                           firstname, lastname, email from
Contacts)
                                                           from Account where Id
IN :Trigger.newMap.keySet()];

  List<Contact> contactsToUpdate = new List<Contact>{};
  // For loop to iterate through all the queried Account records
  for(Account a: accountsWithContacts){
     // Use the child relationships dot syntax to access the related Contacts
     for(Contact c: a.Contacts){
      System.debug('Contact Id[' + c.Id + '], FirstName[' + c.firstname + '], LastName[' +
c.lastname +']');
      c.Description=c.salutation + ' ' + c.firstName + ' ' + c.lastname;
      contactsToUpdate.add(c);
     }
   }

   //Now outside the FOR Loop, perform a single Update DML statement.
   update contactsToUpdate;
}
```

Figura 10 - Ilustração de como proceder a declarações DML

Usando coleções, simplificação de consultas e For Loops eficientes

É importante usar Coleções Apex para consultar dados de forma eficiente e armazenar os dados na memória. Uma combinação de usar coleções e racionalizar consultas SOQL pode ajudar substancialmente a escrever código Apex eficiente e evitar os limites do governador.

Como fazer consultas a grandes conjuntos de dados

O número total de registos que podem ser retornados por consultas do SOQL em um pedido é de 50.000. Se retornar um grande conjunto de consultas exceder limite de *heap*, então uma consulta SOQL *for loop* deve ser usada em vez.

Isto é, em vez de fazer como é descrito na Figura 11, fazer como é mostrado na Figura 12.

```
//A runtime exception is thrown if this query returns enough records to exceed your heap limit.
Account[] accts = [SELECT id FROM account];
```

Figura 11- Ilustração de como não fazer consultas a um enorme quantidade de dados

```
// Use this format for efficiency if you are executing DML statements
// within the for loop.  Be careful not to exceed the 150 DML statement limit.

Account[] accts = new Account[];

for (List<Account> acct : [SELECT id, name FROM account
                           WHERE name LIKE 'Acme']) {
    // Your logic here
    accts.add(acct);
}

update accts;
```

Figura 12 - Ilustração de como fazer consultas a um enorme quantidade de dados

Escrever testes para verificar grandes quantidades de conjunto de dados

Como o código Apex é executado em massa, é essencial ter cenários de teste para verificar se o Apex foi desenvolvido para lidar com grandes conjuntos de dados e não apenas únicos registos.

Evitar *hardocding* Ids

Ao migrar o código Apex entre *sandboxes* e ambientes de produção, ou instalar os pacotes Force.com AppExchange, é essencial evitar identificações de *hardcoding* no código Apex. Ao fazê-lo, se os IDs de registo mudarem entre ambientes, a lógica pode identificar dinamicamente os dados adequados para operar contra e não falhar.

Data Access Object (DAO)

É um padrão para persistência de dados, que permite separar regras de negócio das regras de acesso à base de dados.

No projeto foram utilizadas classes para cada objeto de modo a separar o acesso de dados que são necessários, em termos de objetos específicos do domínio e tipos de dados.

Exemplo: AccountDAO (classe)

```
public static Account getAccountInfoById(Id accountID){
    return [
        SELECT Name, AccountNumber, currencyISOCode, Id, FirstName, LastName, Default_Payment_Method__c,
            BillingStreet, BillingState, BillingCity, BillingCountry, Branding__r.Name,
            BillingPostalCode, PersonMobilePhone, LastModifiedDate, ISFSW__c,
            Parent.Id, RecordType.DeveloperName, Dealer_Qualifier__c, Branding__r.Client_Intro_Text__c,
                Opt_out_of_all_Marketing__c,
            Branding__r.Vehicle_Type_Covered__c, Branding__r.Display_VIN__c, Country__c, Branding__r.Vehicle_Source_System__c
        FROM Account
        WHERE Id = :accountID
    ];
}
```

Figura 13 - Ilustração de um método DAO ao objeto Account

Descrição Trabalho Realizado

Nesta secção será descrito o desenvolvimento do projeto em mais pormenor e como cada tecnologia e ferramenta foi utilizada.

O projeto é constituído por 2 grandes *streams*: CPQ (*Configure Price Quote*) e CSR (*Corporate Social Responsibility*).

A sigla *Configure Price Quote* (CPQ) é um termo usado no setor de negócios para empresas (B2B – Business to Business) para descrever sistemas de software que ajudam as empresas a citar produtos complexos e configuráveis.

Devido ao grau de complexidade do projeto, apenas irei descrever uma das componentes da stream do projeto no qual estive inserido, deixando de parte a stream de CSR.

Metodologia de desenvolvimento

O desenvolvimento do projeto teve como base o método Ágil. **Método ágil** é uma expressão que define um conjunto de metodologias utilizadas no desenvolvimento de software. Grande parte dos métodos ágeis tentam minimizar o risco pelo desenvolvimento do software em curtos períodos, chamados de iteração, os quais gastam tipicamente menos de uma semana a até quatro, neste caso do projeto eram de quatro semanas (com alguns meses em exceção, onde chegaram a atingir cinco semanas). Cada uma destas iterações é como um projeto de software em miniatura de seu próprio, e inclui todas as tarefas necessárias para implantar o mini-incremento da nova funcionalidade: planeamento, análise de requisitos, projeto, codificação, teste e documentação.

Enquanto que, em um processo convencional, cada iteração não está necessariamente focada em adicionar um novo conjunto significativo de funcionalidades, um projeto de software ágil busca a capacidade de implantar uma nova versão do software ao fim de cada iteração, etapa a qual a equipa responsável reavalia as prioridades do projeto.

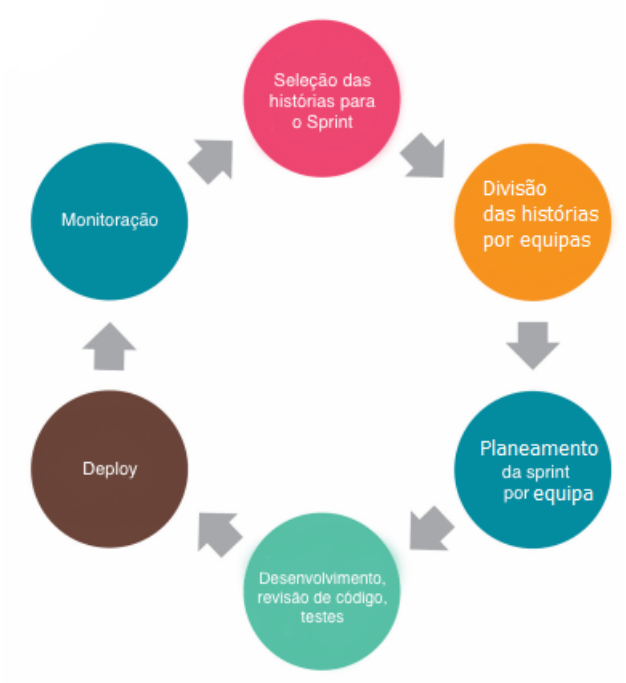


Figura 14 - Ilustração da metodologia

1 – Seleção das *User Stories* (histórias) para o Sprint: ao início de cada sprint, o nosso PO (Product Owner, pessoa responsável pelas decisões que agregam valor ao nosso produto) define as histórias que entrarão em nosso sprint de acordo com o planeamento estratégico.

2 – Divisão das historias por equipas: Antes do planeamento por equipa, o PO já com as histórias selecionadas realiza a divisão e define a prioridade com o líder de cada equipa. Neste projeto, as equipas estão divididas por *streams*, tal como já referi.

3 – Planeamento da sprint por equipa: cada equipa contém um *Technical Lead* (líder técnico), sendo este responsável por dar o tempo estimado para a duração de cada tarefa e assignar ao programador.

4 – Desenvolvimento, revisão de código, testes: para acompanhar o andamento das tarefas são realizadas reuniões diárias, onde cada *technical lead* fala o que foi realizado no dia anterior pela sua equipa e o que planeiam trabalhar durante o dia. Utilizamos um *branch* criado afeto a cada *stream*, usando o nome da *streams* e o número da *User Story* (exemplo: EUCPQ-1234) para desenvolver cada história e, finalizada a User Story, é feito um *pull-request*, através do Gitbucket, para que o

tech lead possa fazer o *code-review* (revisão do código) e fazer o merge para os respetivos ambientes de integração.

5 – *Deploy*: Usando o Git para controlo de versões, quando é feito o "merge" das histórias aprovadas pelo *code-reviewer*, dá-se inicia o processo de testes de aceitação em uma cópia do nosso ambiente de produção (ambientes de integração). Quando todos os testes são aprovados, é feito o *deploy* no ambiente de produção que automaticamente aplica as alterações em todos os servidores, gerando uma nova versão do software.

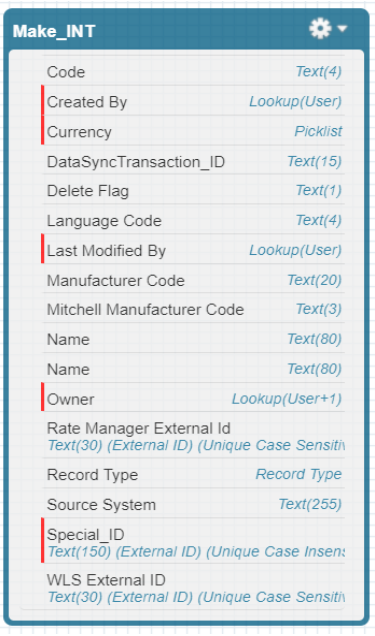
6 - Monitoração: com a história (*User Story*) em produção, é acompanhado pela respetiva equipa dos *Business Analysts (*BAs) através dos relatórios para verificar se houve algum impacto, positivo ou negativo.

CPQUserJourney

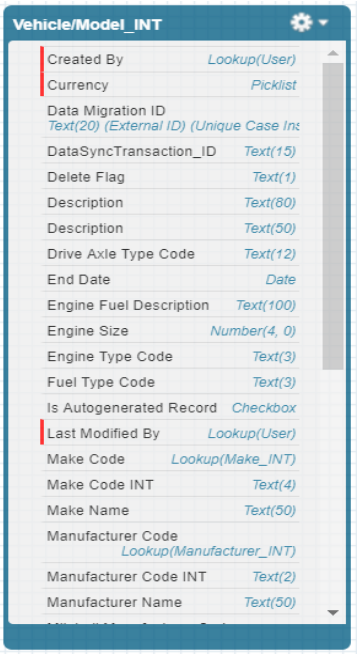
A CPQUserJourney é um conjunto de páginas onde o utilizador é guiado através de introdução de valores e alguns cliques é guiado através do processo de compra de um seguro, desde a introdução dos dados do veículo, a obtenção de rates, dados pessoais e a confirmação da aquisição.

Os objetos utilizados foram os seguintes:

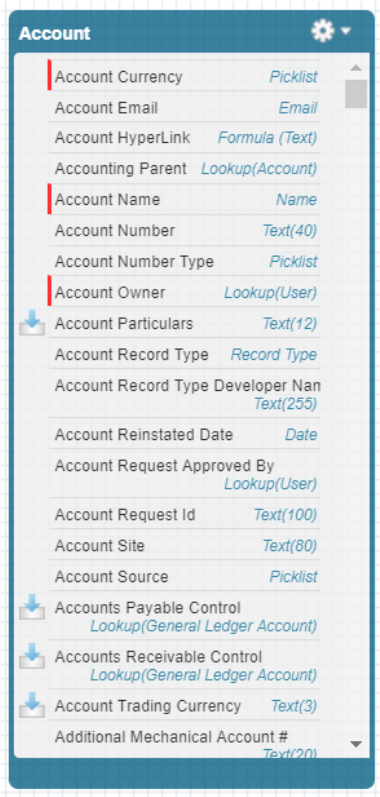
- Make_INT__c (objeto referente ao fabricante do veículo)



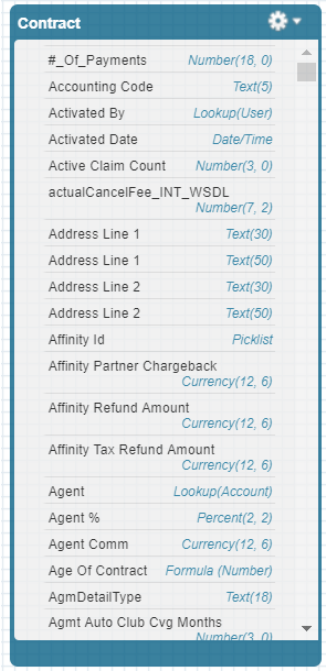
- Vehicle_INT__c (o objeto com a informação do veículo)



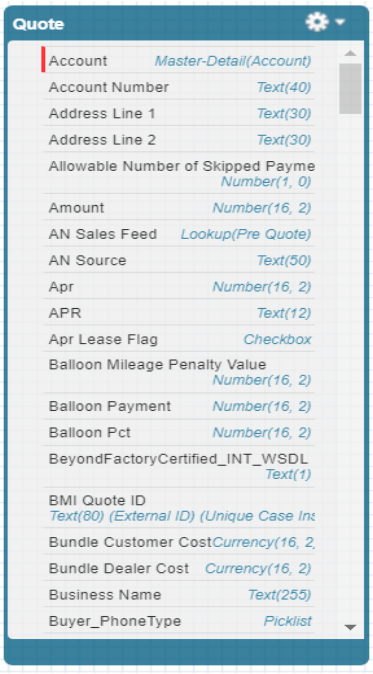
- Account (objeto do *Dealer* - revendedor)



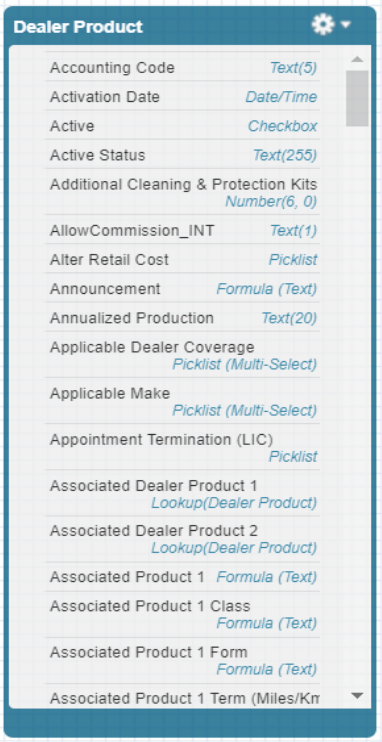
- Contract__c (objeto do Contracto com a informação referente ao mesmo)



- Quote__c (objeto que contém a informação da *Quote* para o cliente)

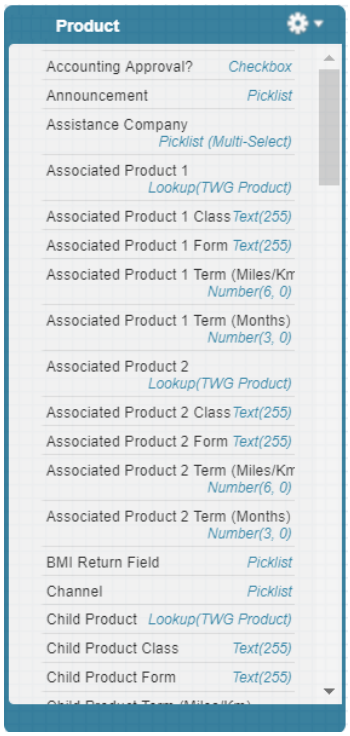


- Dealer_Product__c (objeto que contém a informação referente do produto a cada *Dealer* - revendedor)



- Quote_Dealer_Product__c (objeto que contém informação de um determinado produto, cliente para obtenção de *rates* – taxas)

- ***_Product__c (objeto que contém a informação do produto)



***_Product__c: este objeto contém o nome do cliente e como tal foi escondido por motivos de sigilo

- MulesoftRatesWebService (*Webservice* para obtenção de *rates*)

Irei explicar cada um destas etapas mais em pormenor com exemplos em código e respetiva páginas *Visualforce*.

Módulo 1 - CPQUserJourneyVehicleInformation

Nesta página (*Visualforce Page* - VF) é pedido ao utilizador que selecione um revendedor (Dealer – objeto *Account*), como mostra a Figura 16. Esta pesquisa é feita através de um *lookup* ao objeto Account pelo *DAO* ***public static Account getAccountInfoById(Id accountID)***

```
public static Account getAccountInfoById(Id accountID){
    return [
        SELECT Name, AccountNumber, currencyISOCode, Id, FirstName, LastName, Default_Payment_Method__c,
            BillingStreet, BillingState, BillingCity, BillingCountry, Branding__r.Name,
            BillingPostalCode, PersonMobilePhone, LastModifiedDate, ISFSW__c,
            Parent.Id, RecordType.DeveloperName, Dealer_Qualifier__c, Branding__r.Client_Intro_Text__c,
                Opt_out_of_all_Marketing__c,
            Branding__r.Vehicle_Type_Covered__c, Branding__r.Display_VIN__c, Country__c, Branding__r.Vehicle_Source_System__c
        FROM Account
        WHERE Id = :accountID
    ];
}
```

Figura 15 - Ilustração do método DAO

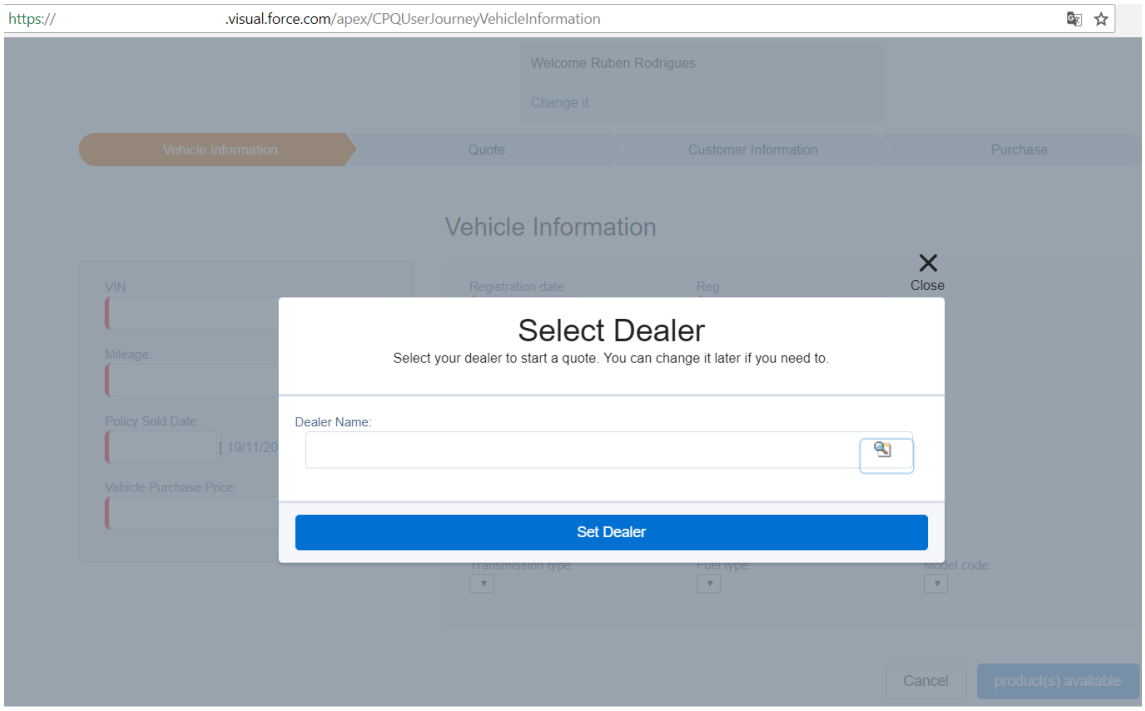


Figura 16 - Lookup para a Account

Após a escolha do *Dealer*, no apex é chamado o método **updateAccount()** que vai permitir colocar o ID do Dealer(Account) no Url através do método **getParameters**() do tipo ***PageReference*** (é um objeto no salesforce, usado para navegar entre uma página diferente ou Url como resultado de um método de ação [*action method*]. É uma referência a uma instância de uma página)

```
public Pagereference updateAccount(){
    PageReference pr = new PageReference('/apex/CPQUserJourneyVehicleInformation');
    pr.getParameters().put('acc',helpQuote.Account__c);
    acc.id = helpQuote.Account__c;
    pr.setRedirect(true);
    return pr;
}
```

Figura 17 – método que vai colocar no URL o id da Account

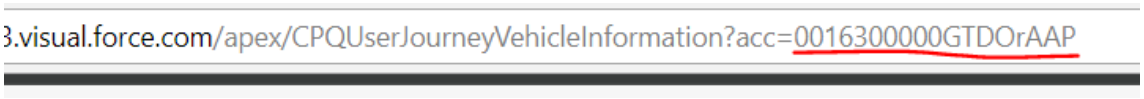


Figura 18 – Ilustração do id no URL

De seguida é pedido ao utilizador que preencha os campos referentes ao Veículo através de **inputfields** definidos no código da página Visualforce. Após preenchimento corretamente dos dados o utilizador ao clicar no botão "*Get Data*" é apresentado a informação existente referente ao veículo com os dados introduzidos:

- **VIN** (Vehicle Idenfication Number)
- **REG** (Vehicle Registration Number)
- ***Mileage*** (Quilometragem)
- ***Policy Sold Date*** (data de venda da apólice)
- ***Vehicle Purchase Price*** (Preço de compra do veículo).

Este botão "*Get Data*" chama *no apex o* método ***retrieveVehicleInfo()*** que irá através da chamada ao *WebService* **HPIVehicleInfoWebService** receber os dados referentes ao dado veículo e retornar as *rates*(taxas) existentes, através da resposta do WebService **MulesoftRatesWebService**. Irá verificar também a existência de *Dealer Products* (objeto *Dealer_Product__c*) referentes ao *Dealer* anteriormente selecionado.

```
public void retrieveVehicleInfo(){
    if(acc.BillingCountry == TWG_Constants.USA){
        UserInputVehicle.VIN__c = searchStringOrReg;
        VINDecoder.VinDecodeResponseWrapper response = VINDecoder.decodeVin(UserInputVehicle,acc,true);
        List<Vehicle__c> vehicleList = response.vehicleList;
        if(!vehicleList.isEmpty()){
            UserInputVehicle = vehicleList[0];
        }
        showVinSection =true;
    }else{
        getRegDecoding();
        getDealerProducts();
        Boolean oneInAllDealerProducts = true;
        for (Dealer_Product__c dealerProduct: dealerProducts)
        {
            if (dealerProduct.ProductClass_INT__c != TWG_Constants.PRODUCTCLASS_INT_IS_ONE)
            {
                oneInAllDealerProducts = false;
            }
        }
        if(UserInputVehicle.Model_Code__c == null && oneInAllDealerProducts && UserInputVehicle.Fuel_Type__c != null
            && UserInputVehicle.CC__c != null && UserInputVehicle.Make__c != null && UserInputVehicle.Model__c != null
            && oneInAllDealerProducts)
        {
            UserInputVehicle.Model_Code__c = TWG_Constants.DEFAULT_MODEL_CODE;
        }
        getRatesWebService();
        numbOfProducts = dealerProducts.size();
    }
}
```

Figura 19 – método para retornar os dados referente ao Vehicle através da chamada do Webservice

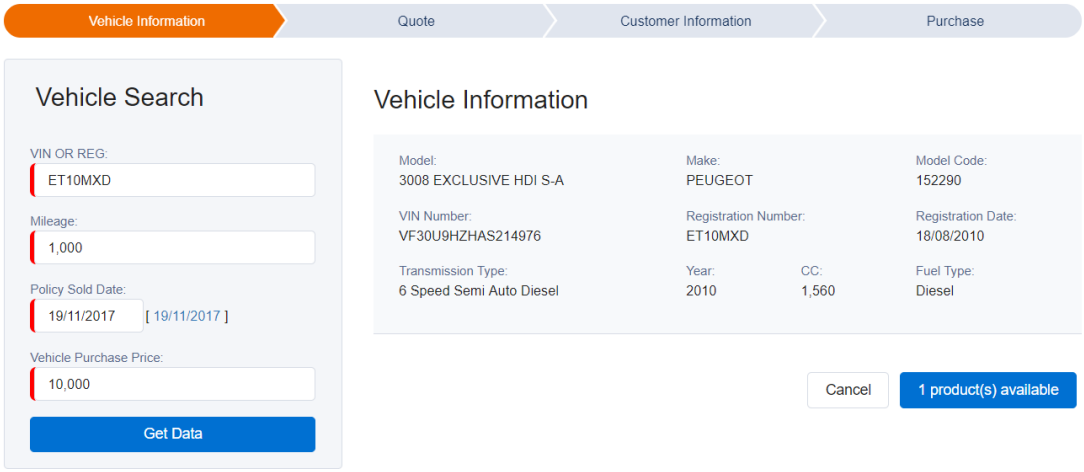


Figura 20 – Ilustração após obtenção dos dados

É então indicado ao utilizador o número de Produtos e a existência de *rates* no botão (Figura 20). Botão esse que fará o utilizador navegar para a seguinte página através do método **getRates()** (Figura 21), retornando no URL o Id do veículo acompanhado com o id da Account.

```
public Pagereference getRates(){
    PageReference pr = new PageReference('/apex/CPQUserJourneyQuoteRates');
    if(acc.id != null){
        if(contactId == null)pr.getParameters().put('acc',Acc.id);
        helpQuote.Account__c=acc.id;
    }
    else {
        ApexPages.addMessage(new ApexPages.Message(ApexPages.Severity.ERROR,Label.No_Valid_Account_Selected));
        return null;
    }
    if(UserInputVehicle.Odometer__c==null){
        ApexPages.addMessage(new ApexPages.Message(ApexPages.Severity.ERROR,Label.Fill_In_The_Mileage));
    }
    if(UserInputVehicle.Policy_sold_date__c==null){
        ApexPages.addMessage(new ApexPages.Message(ApexPages.Severity.ERROR,Label.Fill_In_The_Policy_Sold_Date));
    }
    if(UserInputVehicle.Vehicle_Purchase_Price__c==null){
        ApexPages.addMessage(new ApexPages.Message(ApexPages.Severity.ERROR,Label.Fill_In_The_Purchase_Price));
    }
    if (ApexPages.hasMessages()) return null;
    Upsert UserInputVehicle;
    pr.getParameters().put('veh',UserInputVehicle.id);
    pr.setRedirect(true);
    return pr;
}
```

Figura 21 – Ilustração do método que fará navegar para a pagina CPQUserJourneyQuoteRates

Módulo 2 - CPQUserJourneyQuoteRates

Nesta página (Figura 22) é apresentado ao utilizador as *rates* disponíveis. Estas *rates* são retornadas pelo o método **getRatesWebService()**, onde é feito a chamada ao *Webservice* **MulesoftRatesWebService** para enviar o pedido (*Request*) com os parâmetros dos objetos da *Account, Vehicle__c* e *Dealer_Product__c*.

```
public void getRatesWebService(){
    Integer i = 0;
    List<MulesoftRatesWebService.MulesoftRatesCollection> rcList = new List<MulesoftRatesWebService.MulesoftRatesCollection>(
        );
    if(dealerProducts.size()==0){
        ApexPages.addMessage(new ApexPages.Message(ApexPages.Severity.ERROR, Label.No_product_found));
        return;
    }
    try
    {
        rcList = MulesoftRatesWebService.sendRequest(acc, UserInputVehicle, dealerProducts);
        ratesResponse = MulesoftRatesWebService.ratesResponse.getBody();
        dealerProducts = processResponse(ratesResponse, dealerProducts);
    }
    catch (Exception e)
    {
        error = new TWG_System_Error__c();
        error.Error_Message__c = Label.Unable_To_Retrieve_Rates;
    }
```

Figura 22 – Ilustração do método que invoca o Webservice para obtenção das rates

Após a escolha da *rate* a ditar no contrato (Figura 23), o utilizador grava a sua escolha e é enviado para a página **CPQUserJourneyCustomerInfo**, onde é apresentado campos a preencher referentes ao utilizador, assim como, os valores a vincular no contrato com a aplicação das *rates*. Os cálculos são efetuados de acordo com a resposta do WebService **CPQReratingServiceResponse.**

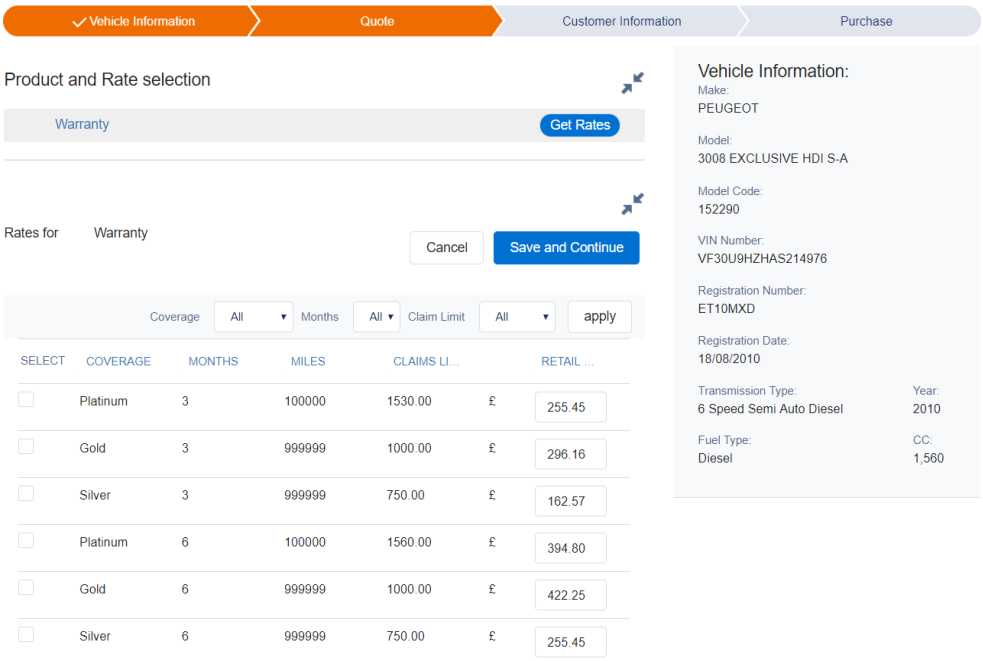


Figura 23 – Ilustração da página contendo as rates

Módulo 3 - CPQUserJourneyCustomerInfo

Esta página destina-se à introdução dos dados pessoais referentes do utilizador assim como, a apresentação dos custos do contrato calculados pela *rate* escolhida anteriormente (Figura 26). Ao clicar em "*Save and Continue*" é executada uma função escrita em JavaScript (***PurchaseQuoteAndGenerateDocuments* –** Figura 24) para invocar 2 métodos no Apex (**saveAndContinue** e **saveAndContinueRedirect** – Figura 25)

```
function PurchaseQuoteAndGenerateDocuments(){
    SaveQuote();
    redirectQuote();
}
```

Figura 24 – método PurchaseQuoteAndGenerateDocuments

```
                <apex:commandButton styleClass="btn slds-button slds-x-small-button--horizontal
                slds-button--brand slds-button--last"
                    status="processing" rerender="DummyPanel" value="Save and Continue"
                    onclick="PurchaseQuoteAndGenerateDocuments()"
                />
            </div>
        </div>
    </div>

    <apex:actionFunction action="{!saveAndContinue}" name="SaveQuote" rerender="dummyPanel" status="
    processing">

    </apex:actionFunction>
    <apex:actionFunction action="{!saveAndContinueRedirect}" name="redirectQuote" rerender="msg" status="
    processing">
    </apex:actionFunction>
```

Figura 25 – Invocação dos métodos no apex

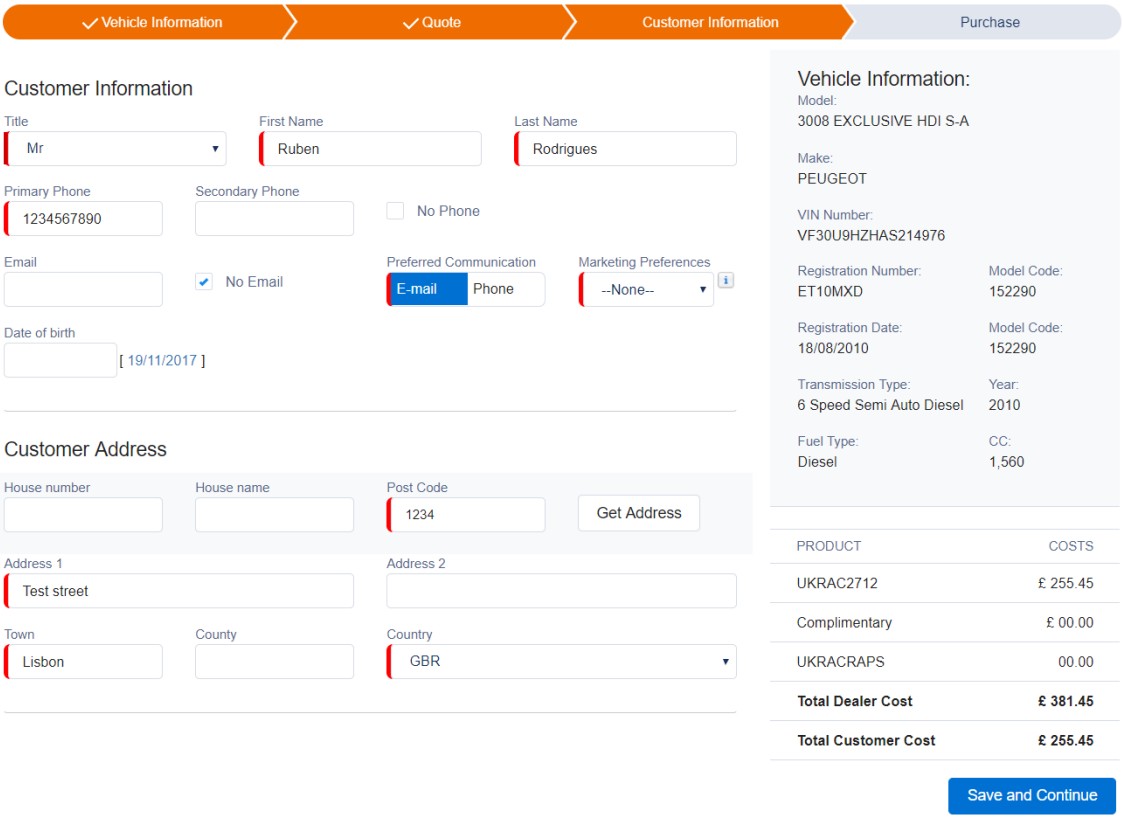


Figura 26 – Ilustração da página com dados preenchidos

Módulo 4 - CPQUserJourneyPurchaseQuote

Após introdução correta dos dados é apresentada a página **CPQUserJourneyPurchaseQuote** (Figura 29) com os dados do contrato a que o utilizador está prestes a acordar, bem como, um documento ilustrativo mais em detalhe. Documento este gerado pelo o Drawloop através do método **generateContractDocuments()** (Figura 27).

```
public void generateContractDocuments(){
    List<Contract__c> contractsForDocuments = new List<Contract__c>();
    for(Contract__c con : contractsToInsert){
        if(!con.Complimentary__c && !con.Bundled__c)contractsForDocuments.add(con);

    }
    DrawloopApexDocumentGenerationService.massMerge(contractsForDocuments);
}
```

Figura 27 – Método generateContractDocuments

No método **massMerge(contractsForDocuments)** da classe **DrawloopApexDocumentGenerationService**, o que está a ser feito é a recolha do objeto para o qual se está a criar o documento (neste caso para o objeto Contract__c) e pegar nos campos deste objeto para preencher o documento (Figura 28).

```
public static void massMerge(List<sObject> newObjects){
    if(newObjects != null && newObjects.size()>0){
        String objectName=newObjects[0].id.getSObjectType().getDescribe().getName();
        objectName = objectName.substring(0,objectName.length()-3);
                Loop.loopMessage lm = new Loop.loopMessage();
                Drawloop_DDP_Settings_for_Apex__c DDPSettings = Drawloop_DDP_Settings_for_Apex__c.getValues(objectName);
                Map<string, string> variables;
                variables = new Map<string, string> { 'deploy' => DDPSettings.DDP_Delivery_ID__c };
                for(sObject tmpobject : newObjects){
                    lm.requests.add(new Loop.loopMessage.loopMessageRequest(tmpobject.id,DDPSettings.DDP_ID__c,variables));
                }
                Lm.sendAllRequests();

    }
}
```

Figura 28 – método do Drawloop massMerge(List<sObject newObjects)

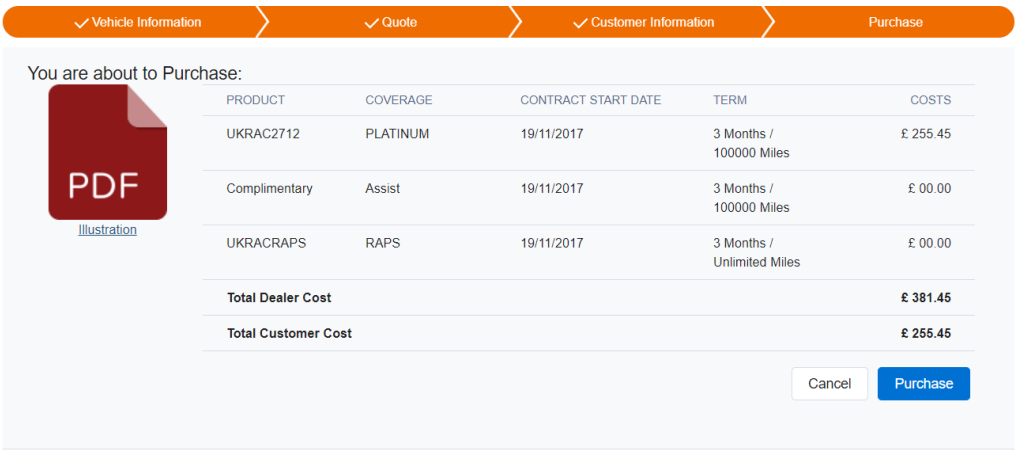
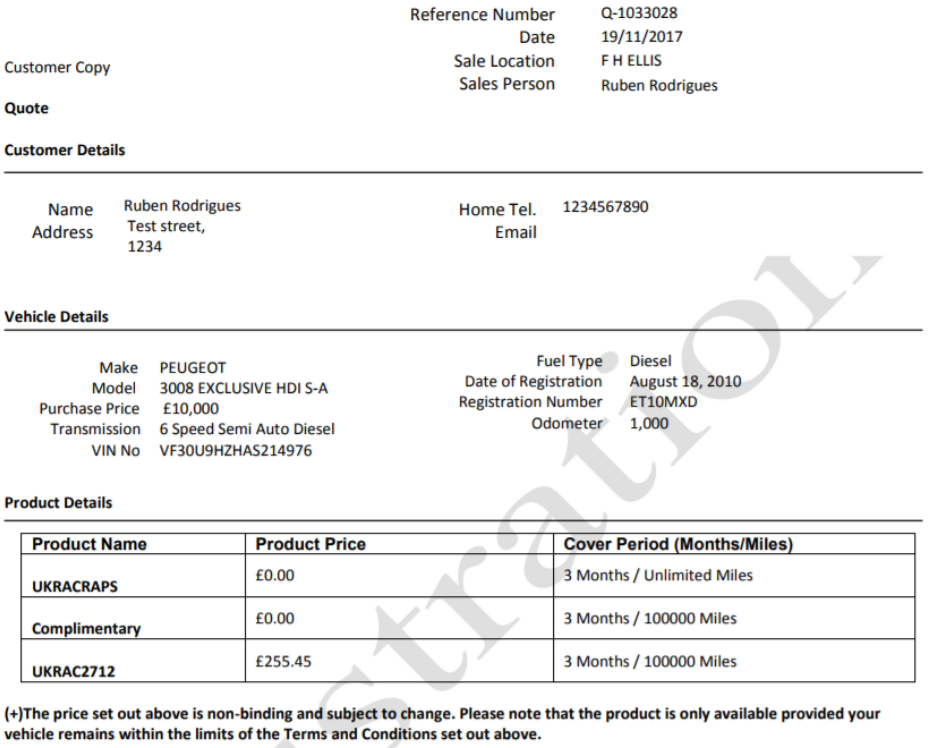


Figura 29 – Apresentação dos valores do contrato



Reference Number Q-1033028
Date 19/11/2017
Sale Location F H ELLIS
Sales Person Ruben Rodrigues

Customer Copy

Quote

Customer Details

Name Ruben Rodrigues
Address Test street, 1234
Home Tel. 1234567890
Email

Vehicle Details

Make PEUGEOT
Model 3008 EXCLUSIVE HDI S-A
Purchase Price £10,000
Transmission 6 Speed Semi Auto Diesel
VIN No VF30U9HZHAS214976
Fuel Type Diesel
Date of Registration August 18, 2010
Registration Number ET10MXD
Odometer 1,000

Product Details

Product Name	Product Price	Cover Period (Months/Miles)
UKRACRAPS	£0.00	3 Months / Unlimited Miles
Complimentary	£0.00	3 Months / 100000 Miles
UKRAC2712	£255.45	3 Months / 100000 Miles

(+)The price set out above is non-binding and subject to change. Please note that the product is only available provided your vehicle remains within the limits of the Terms and Conditions set out above.

Figura 30 - Ilustração do contrato

Módulo 5 - CPQUserJourneyContractPurchased

Após a confirmação do utilizador para o acordo do contrato, através do botão "**Purchase**", é encaminhado para a página **CPQUserJourneyContractPurchased**. Esta página serve apenas de suporte para a criação do documento final, que representa o contrato. Este documento, mais uma vez, é gerado através do Drawloop pelo o método **manualGenerateContractDocument()**. Neste caso é usado a notação **@remoteAction** para servir de suporte ao método apex usado na página Visualforce (Figura 32) para ser chamado via JavaScript (Figura 32)

```
@remoteAction
public static void manualGenerateContractDocument(String contractId){
    Contract__c contractToGenerateDocuments = ContractDAO.getContractById(contractId);
    DrawloopApexDocumentGenerationService.massMerge(contractToGenerateDocuments);
}
```

Figura 31 – método que invoca o Drawloop para criação do documento

```
</div>
<apex:commandbutton styleclass="btn slds-button
slds-x-small-button--horizontal slds-button--brand  slds-button--last
"
                    onclick="deleteContractDocument({!qdp.contractObject.id});
                        generateAttachmentManually({!qdp.contractObject.id})"
                    value="{!$Label.Generate_Documents}" rerender="dummyPanel"></
                    apex:commandbutton>
```

Figura 32 – método JavaScript usado na Visualforce

```
function generateAttachmentManually(id) {
    $(id).find(".pdfIMG").hide();
    $(id).find(".loadingPDF").show();
    CPQUserJourneyController.manualGenerateContractDocument(
        id.id,
        function(result, event) {
            handleManualResult(result, event, id);
        }
    );
}
```

Figura 33 – método manualGenerateContractDocument usado no apex

Cronograma geral do Projeto

Como é visível na Figura 34 no inicio do estágio com data de início a 14/02/2017, o projeto encontrava-se na Sprint 12 e foi desenvolvido e acompanhando até à data 14/11/2017 (conclusão do estágio). Antes de ter sido integrado no projeto tive formação durante 1mês em Salesforce nas vertentes de *Developer* e de *Admin*. Posteriormente, e através da empresa, adquiri uma importante certificação em ***Salesforce Certified Platform Developer I***, o que foi uma mais valia em termos de aquisição de conhecimento e permitiu, também, uma melhor compreensão para o acompanhamento e desenvolvimento do projeto inserido.

Nome da Tarefa	Início	Término	Duração (dias)
UKCPQ Sprint 12	14/02/2017	07/03/2017	21
UKCPQ Sprint 13	08/03/2017	11/04/2017	34
UKCPQ Sprint 14	12/04/2017	02/05/2017	20
EU MOTOR Sprint 15	03/05/2017	06/06/2017	34
EU MOTOR Sprint 16	07/06/2017	11/07/2017	34
EU MOTOR Sprint 17	12/07/2017	08/08/2017	27
EU MOTOR Sprint 18	09/08/2017	05/09/2017	27
EU MOTOR Sprint 19	06/09/2017	26/09/2017	20
EU MOTOR Sprint 20	27/09/2017	24/10/2017	27
EU MOTOR Sprint 21	25/10/2017	14/11/2017	20

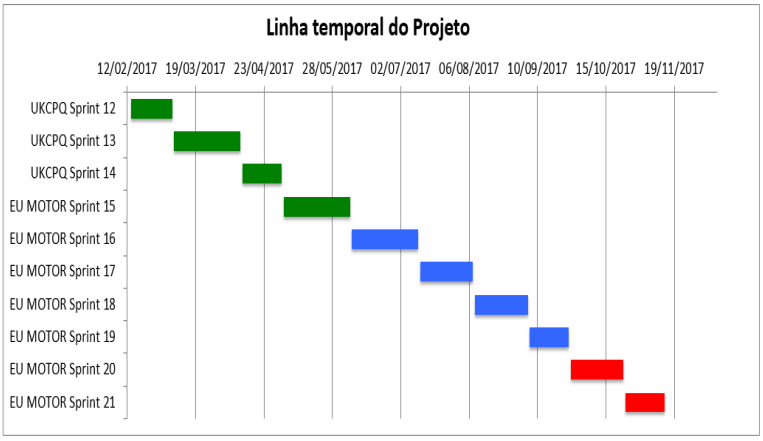
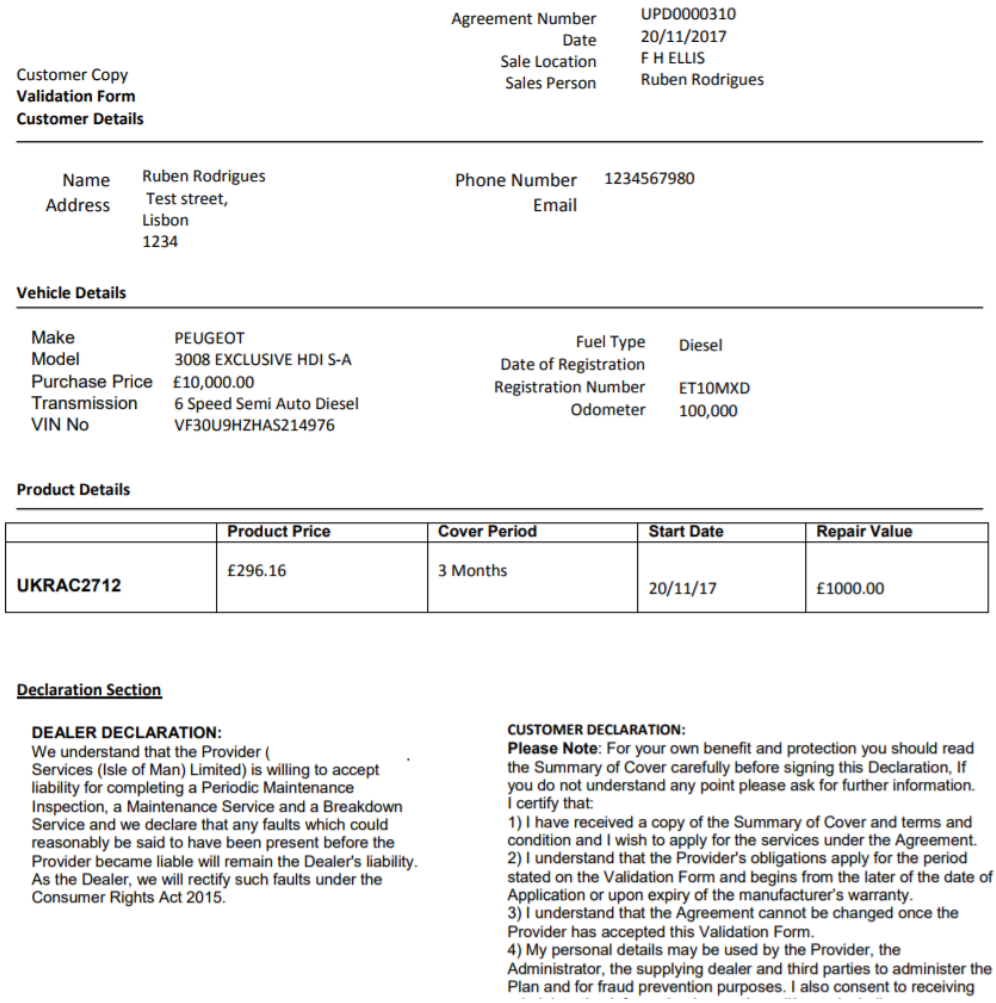


Figura 34 - Ilustração do tempo afeto ao projeto por sprints durante o período do estágio

Resultados e Discussão

Neste tópico será apresentado o resultado final do projeto desenvolvido. Posteriormente, serão também apontados alguns problemas que surgiram e como foram resolvidos.

Após a conclusão da CPQUserJourney é gerado o contrato através do *Drawloop*, como é mostrado na Figura 35.



Customer Copy
Validation Form
Customer Details

Agreement Number UPD0000310
Date 20/11/2017
Sale Location F H ELLIS
Sales Person Ruben Rodrigues

Name Ruben Rodrigues
Address Test street, Lisbon 1234
Phone Number 1234567980
Email

Vehicle Details

Make PEUGEOT
Model 3008 EXCLUSIVE HDI S-A
Purchase Price £10,000.00
Transmission 6 Speed Semi Auto Diesel
VIN No VF30U9HZHAS214976
Fuel Type Diesel
Date of Registration
Registration Number ET10MXD
Odometer 100,000

Product Details

	Product Price	Cover Period	Start Date	Repair Value
UKRAC2712	£296.16	3 Months	20/11/17	£1000.00

Declaration Section

DEALER DECLARATION:
We understand that the Provider (Services (Isle of Man) Limited) is willing to accept liability for completing a Periodic Maintenance Inspection, a Maintenance Service and a Breakdown Service and we declare that any faults which could reasonably be said to have been present before the Provider became liable will remain the Dealer's liability. As the Dealer, we will rectify such faults under the Consumer Rights Act 2015.

CUSTOMER DECLARATION:
Please Note: For your own benefit and protection you should read the Summary of Cover carefully before signing this Declaration, If you do not understand any point please ask for further information.
I certify that:
1) I have received a copy of the Summary of Cover and terms and condition and I wish to apply for the services under the Agreement.
2) I understand that the Provider's obligations apply for the period stated on the Validation Form and begins from the later of the date of Application or upon expiry of the manufacturer's warranty.
3) I understand that the Agreement cannot be changed once the Provider has accepted this Validation Form.
4) My personal details may be used by the Provider, the Administrator, the supplying dealer and third parties to administer the Plan and for fraud prevention purposes. I also consent to receiving

Figura 35 - Resultado final após conclusão da CPQUserJourney

As maiores dificuldades encontradas foram, devido à complexidade do projeto, ao inicio a adaptação foi complicada e foi necessário um esforço enorme da minha parte para conseguir acompanhar tudo e todos. Também como são vários programadores a trabalhar no mesmo projeto, por vezes ficamos bloqueados nas nossas tarefas devido ao facto de outros programadores estarem a trabalhar na mesma classe (apesar de tarefas diferentes), o que pode influenciar os prazos de entrega das tarefas.

Conclusões

Concluo assim a apresentação do projeto realizado no meu estágio curricular. Espero ter conseguido expor da melhor forma o trabalho desenvolvido ao longo destes 9 meses.

Foi uma experiência bastante enriquecedora, não só a nível profissional como também pessoal. Tendo sido a minha primeira experiência num projeto profissional, a verdade é que a realidade académica é bastante diferente da realidade empresarial, daí ser tão importante este contato. Permitiu-me também crescer como pessoa e adquirir qualidades de extrema importância no mundo do profissional e conhecimentos até à data ainda não adquiridos ou consolidados.

Como seria de esperar surgiram alguns desafios. Um deles foi a adaptação a programação em uma plataforma CRM, tendo o estudo para a certificação em ***Salesforce Certified Platform Developer I*** ajudado bastante. Mais uma vez agradeço à Right IT Services por esta oportunidade e por todo o apoio que me foi fornecido ao longo destes meses. Outro desafio com que me deparei foi o desconhecimento de grande parte das tecnologias com que trabalhei. Devido a isso tive de aumentar os meus esforços para conseguir desenvolver e acompanhar o projeto de qualidade no tempo disponível. Ao longo do percurso académico desenvolvi diversos projetos e em todos eles existia alguma pressão, não só de cumprir o tempo delimitado, mas também de conseguir alcançar os objetivos. Contudo desenvolver um projeto numa empresa, aumenta bastante a pressão e também o desejo de ultrapassar as metas estabelecidas a nível pessoal. Ao longo do estágio, senti essa pressão. Mas creio que é algo com que todos passamos e temos de aprender a lidar e com certeza este estágio ajudou-me neste sentido.

Durante estes nove meses, tentei em todas as ocasiões dar o meu melhor e espero ter conseguido acrescentar algum valor à Right IT Services com o trabalho que desenvolvi.

O meu balanço global do estágio é bastante positivo e posso afirmar que consegui alcançar os meus objetivos.

Webgrafia

https://developer.salesforce.com/page/An_Introduction_to_Environments

https://en.wikipedia.org/wiki/Salesforce.com#Salesforce

https://developer.salesforce.com/docs/atlas.en-us.apexcode.meta/apexcode/apex_writing.htm

https://developer.salesforce.com/docs/atlas.en-us.apexcode.meta/apexcode/apex_continuation_overview.htm

https://pt.wikipedia.org/wiki/Bitbucket

https://pt.wikipedia.org/wiki/Trello

https://en.wikipedia.org/wiki/Salesforce.com#Salesforce

https://pt.wikipedia.org/wiki/Jira

http://searchsalesforce.techtarget.com/definition/Salesforce-AppExchange