

**UNIVERSIDADE TÉCNICA DE ANGOLA**

**FACULDADE DE ENGENHARIAS**

**DEPARTAMENTO DE ENSINO E INVESTIGAÇÃO DAS TECNOLOGIAS DE INFORMAÇÃO E COMUNICAÇÃO DEITIC**

**Sistema de Gestão de Serviços Da Oficina DARTIN**

SARA NGUEVE JEREMIAS CELESTINO

**LUANDA-ANGOLA**

**OUTUBRO/2018**

**UNIVERSIDADE TÉCNICA DE ANGOLA**

**FACULDADE DE ENGENHARIAS**

**DEPARTAMENTO DE ENSINO E INVESTIGAÇÃO DAS TECNOLOGIAS DE INFORMAÇÃO E COMUNICAÇÃO DEITIC**

**CURSO DE ENGENHARIA INFORMÁTICA**

**Sistema de Gestão de Serviços Da Oficina DARTIN**

**SARA NGUEVE JEREMIAS CELESTINO**

*Trabalho de Fim de curso apresentado a Universidade Técnica de Angola, como requisito para obtenção do título de licenciado em Engenharia Informática.*

**Orientador: Msc Amiraldes Xavier**

**LUANDA-ANGOLA**

**OUTUBRO/2018**

**SARA NGUEVE JEREMIAS CELESTINO**

**Sistema de Gestão de Serviços Da Oficina DARTIN**

**Aprovado: \_\_\_\_\_/\_\_\_\_\_/\_\_\_\_\_**

|  |
| --- |
| BANCA EXAMINADORA  Presidente do Júri………………………………………… Assinatura………………………  Primeiro Vogal…………………………………………… Assinatura………………………  Segundo Vogal……………………………………………Assinatura………………………. |

# DEDICATÓRIA

Ao meu pai “***Ferreira Celestino***” por mostrar que apenas acreditando e trabalhando conseguimos o sucesso na vida.

A minha mãe “***Avelina Jeremias***” por me ensinar que somente com a humildade é que nos tornamos grandes homens.

Aos meus Irmão (Cipriano, Ruth, Abel, Isabella e Florinda).

# AGRADECIMENTO

À Deus o Altíssimo – pois sem ele, desde já não estaria escrevendo isso, “Shalon Shalin Shalem”.

Aos meus pais por tudo, pelo sacrifício feito para que enfim eu chegasse aqui, pois sois os únicos responsáveis de ser o que hoje sou.

Ao meu querido Tutor Msc Amiraldes Xavier, que com sua doce e incrível paciência, sabedoria e profissionalismo soube guiar-me nesta trajetória difícil.

Ao Eng. Romário Meireles por contribuir com sua paciência para a realização do mesmo.

A todos os professores que de forma directa contribuíram com a sua sabedoria para a consolidação da minha formação.

Ao meu Irmão querido Cipriano Celestino e sua esposa Adelina Celestino que com muita paciência apoiaram-me incondicionalmente em toda a minha trajetória até chegar aqui.

Aos meus colegas, em especial a turma EIM4.1 (2017), pelo apoio que me foi fornecido nessa dura e longa caminhada!

A todos que de forma directa ou indirecta contribuíram para o sucesso deste trabalho.

**Muito obrigada…**

# EPÍGRAFE

*“Apesar de nossos defeitos, precisamos enxergar que somos pérolas únicas no teatro da vida e entender que não existem pessoas de sucesso ou pessoas fracassadas. O que existe são pessoas que lutam pelos seus sonhos ou desistem deles”*

***Augusto Cury***

# RESUMO

Os Softwares de gestão são hoje solicitados por muitas empresas para o seu melhor funcionamento, a criação de sistemas de gestão tornou-se uma necessidade que o nível de exigência de qualidade dos mesmos tem aumentado a cada dia que passa.

O objectivo deste trabalho é criar um software de gestão de Serviço para a Oficina da Empresa DARTIN, pois o mesmo será uma ferramenta ideal e imprescindível para aumentar a qualidade do serviço prestado bem como a sua produtividade, garantindo assim uma melhor gestão da oficina, na gestão de clientes, orçamentos de clientes, gestão de viaturas, plano de manutenção das viaturas com respectivos agendamentos. E estas são apenas algumas das funcionalidades que decerto vão agilizar o seu controlo administrativo, e rentabilizar as suas operações.

Portanto, quem administra uma Oficina vai perceber que a medida que o negócio vai crescendo ele perceberá que tem de fazer um melhor acompanhamento do que está acontecendo, seja para o levantamento do que realmente se lucrou ou mesmo para levantar seu custo operacional, isto num nível mais avançado; então pensou-se no desenvolvimento do referido Software para a Gestão de Serviços para a Oficina da Empresa DARTIN.

***Palavras-chave: Software de Gestão, Oficina.***

# ABSTRACT

The management software is now required by many companies for its better functioning, the creation of management systems has become a necessity that the level of quality requirement of them has increased with each passing day.

The objective of this work is to create a service management software for the DARTIN Company Workshop, as it will be an ideal and indispensable tool to increase the quality of the service rendered as well as its productivity, thus guaranteeing a better management of the workshop, in the customer management, customer budgets, vehicle management, vehicle maintenance plan with respective schedules. And these are just some of the features that will certainly streamline your administrative control and monetize your operations.

Therefore, who administers a workshop will realize that as the business grows, it will realize that it has to do a better follow-up of what is happening, whether it is for the survey of what was actually profited or even to raise its operational cost, this at a level more advanced; We then thought of developing the aforementioned Service Management Software for the DARTIN Enterprise Workshop.

Key-Words: Management Software, Workshop.

**ÍNDICE**

[DEDICATÓRIA i](#_Toc521413279)

[AGRADECIMENTO ii](#_Toc521413280)

[EPÍGRAFE iii](#_Toc521413281)

[RESUMO iv](#_Toc521413282)

[ABSTRACT v](#_Toc521413283)

[ÍNDICE DE FIGURAS ix](#_Toc521413284)

[ÍNDICE DE TABELAS x](#_Toc521413285)

[1. INTRODUÇÃO 1](#_Toc521413286)

[1.1 Definição do problema 1](#_Toc521413287)

[1.2 Justificativa 2](#_Toc521413288)

[1.3 Objectivos 2](#_Toc521413289)

[**1.3.1** **Objectivo Geral** 2](#_Toc521413290)

[**1.3.2** **Objectivos específicos:** 2](#_Toc521413291)

[1.4 Organização do trabalho 2](#_Toc521413292)

[2. FUNDAMENTAÇÃO TEÓRICA 4](#_Toc521413293)

[2.1 Sistema de Gestão 4](#_Toc521413294)

[**2.1.1** **Vantagens de um sistema de gestão** 4](#_Toc521413295)

[**2.1.2** **Sistemas de Gestão para Oficinas** 5](#_Toc521413296)

[2.2 Engenharia de Software 8](#_Toc521413297)

[**2.2.1** **Requisitos do Sistema** 8](#_Toc521413298)

[**2.2.2** **Modelação de Sistemas** 9](#_Toc521413299)

[2.3 Base de Dados 10](#_Toc521413300)

[**2.3.1** **Sistemas de base de dados** 11](#_Toc521413301)

[**2.3.2** **Modelo Entidade Relacionamento (MER)** 11](#_Toc521413302)

[**2.3.3** **SQL** 12](#_Toc521413303)

[2.4 Processo de desenvolvimento de sistemas 13](#_Toc521413304)

[**2.4.1** **Processo Incremental** 15](#_Toc521413305)

[2.5 Qualidade de Software 16](#_Toc521413306)

[2.6 Arquitectura de Software 17](#_Toc521413307)

[**2.6.1** **Linguagem de Programação** 17](#_Toc521413308)

[**2.6.2** **Linguagem C#** 18](#_Toc521413309)

[**2.6.3** **Programação em camadas** 18](#_Toc521413310)

[2.7 Linguagens e Tecnologias Web 19](#_Toc521413311)

[**2.7.1** **ASP.NET** 20](#_Toc521413312)

[**2.7.2** **CSS** 20](#_Toc521413313)

[3. METODOLOGIA 22](#_Toc521413314)

[3.1 Metodologia de Investigação Científica (Metodologia de Pesquisa) 22](#_Toc521413315)

[3.2 Descrição do Campo de estudo 22](#_Toc521413316)

[3.3 Processo de Desenvolvimento 22](#_Toc521413317)

[3.4 Análise de Requisitos 23](#_Toc521413318)

[**3.4.1** **Requisitos Funcionais:** 23](#_Toc521413319)

[**3.4.2** **Requisitos não funcionais** 25](#_Toc521413320)

[**3.4.3** **Regras de negócio do nosso sistema** 25](#_Toc521413321)

[3.5 Diagramas UML do nosso Sistema 26](#_Toc521413322)

[***3.5.1*** **Diagrama de Casos de Uso** 26](#_Toc521413323)

[**3.5.2** **Descrição dos principais casos de uso do nosso sistema** 27](#_Toc521413324)

[**3.5.3** **Matriz de Rastreabilidade** 30](#_Toc521413325)

[**3.5.4** **Diagrama de Actividades** 30](#_Toc521413326)

[**3.5.5** **Diagrama de Sequência** 33](#_Toc521413327)

[**3.5.6** **Diagrama de Classe** 35](#_Toc521413328)

[**3.5.7.** **Diagrama de Instalação** 36](#_Toc521413329)

[**3.5.8.** **Modelo conceitual** 36](#_Toc521413330)

[**3.5.9.** **Modelo lógico** 37](#_Toc521413331)

[3.6. Ferramentas utilizadas 38](#_Toc521413332)

[4. RESULTADOS 40](#_Toc521413333)

[5. CONCLUSÃO 44](#_Toc521413334)

[5.1 Recomendações 44](#_Toc521413335)

[6. REFERÊNCIAS BIBLIOGRÁFICAS 45](#_Toc521413336)

# ÍNDICE DE FIGURAS

[Figura 2. 1 Sistemas de gestão para Oficinas Mecânicas 6](#_Toc525714540)

[Figura 2. 2 Sistemas de Gestão Para Oficinas Integrada 7](#_Toc525714541)

[Figura 2. 3 Processo Incremental 15](#_Toc525714542)

[Figura 2. 4 Arquitetura de Software 17](#_Toc525714543)

[Figura 3. 1 Diagrama de Caso de Uso do projecto. 27](#_Toc525714574)

[Figura 3. 2 Diagrama de Actividade Cadastrar Cliente. 31](#_Toc525714575)

[Figura 3. 3 Diagrama de Actividade Visualizar Serviço. 32](#_Toc525714576)

[Figura 3. 4 Diagrama de Sequência Cadastrar Cliente. 33](#_Toc525714577)

[Figura 3. 5 Diagrama de Sequência Visualizar Serviço. 34](#_Toc525714578)

[Figura 3. 6 Diagrama de Classe 35](#_Toc525714579)

[Figura 3. 7 Diagrama de instalação do sistema. 36](#_Toc525714580)

[Figura 3. 8 Modelo Conceitual 37](#_Toc525714581)

[Figura 3. 9 Diagrama Entidade Relacional 38](#_Toc525714582)

[Figura 4. 1 Página inicial 41](#_Toc525591609)

[Figura 4. 2 Cadastro de solicitações de Serviços 41](#_Toc525591610)

[Figura 4. 3 Cadastro se Usuários 42](#_Toc525591611)

[Figura 4. 4 Login 43](#_Toc525591612)

[Figura 4. 5 Cadastro de Clientes 43](#_Toc525591613)

# ÍNDICE DE TABELAS

[Tabela 3. 1 Requisitos Funcionais 24](#_Toc525591699)

[Tabela 3. 2 Requisitos Não Funcionais 25](#_Toc525591700)

[Tabela 3. 3 Regras de Negócio 26](#_Toc525591701)

[Tabela 3. 4Descrição do Caso de Uso Cadastrar Cliente 27](#_Toc525591702)

[Tabela 3. 5Descrição do Caso Atribuir serviço ao mecânico 28](#_Toc525591703)

[Tabela 3. 6 Descrição do Caso de Uso Visualizar Serviço 28](#_Toc525591704)

[Tabela 3. 7 Descrição do Caso de Uso Solicitar Serviço 29](#_Toc525591705)

[Tabela 3. 8 Descrição do Caso de Uso Registrar Pagamento. 29](#_Toc525591706)

[Tabela 3. 9 Matriz de rastreabilidade 30](#_Toc525591707)

# INTRODUÇÃO

As oficinas mecânicas hoje em dia, contam como auxílio de softwares para gerenciar o ambiente de trabalho e melhorar o atendimento ao cliente e contribuindo muito para o aumento da produtividade e lucratividade do negócio. Pois administrar uma oficina exige evolução constante de toda equipa envolvida. Para se adaptar ao mercado actual, é preciso que a empresa seja bem organizada e que se desenvolva uma boa estratégia para o funcionamento interno.

Então é imprescindível que uma oficina mecânica tenha o completo controle das informações referentes aos veículos que estão em processo de manutenção ou reparo dentro do seu espaço. Este tipo de organização deve estar presente desde o inicio do atendimento, na abertura da ordem de serviço, até a finalização do trabalho do cliente.

Basicamente serão os programas de computador que farão toda a gestão da oficina, ou seja, o controle do agendamento do cliente, o fluxo de veículos, o estoque, a parte administrativa, os pagamentos, as reclamações e muitos outros requisitos importantes para optimizar o trabalho do empresário da reparação.

## **Definição do problema**

Durante a entrevista feita com o gestor da Empresa DARTIN, notou-se que a gestão de informação, é feita através do uso dos papéis, o que por vezes causa muita demora na procura do processo dos clientes, e devido a esta demora, que na maior parte das vezes causa a perda de tempo em que poderiam ter gasto em mais outras funções o que consequentemente causa a insatisfação dos clientes. Daí surge a necessidade de implementar esse sistema de gestão para a mesma.

Tendo em conta aos problemas aqui apresentados, formulou-se as seguintes questões de pesquisa para o nosso trabalho:

* **Será possível com a implementação do sistema resolver problemas da empresa, na organização e registros de dados?**
* **Será que os meios a serem usados irão ajudar na gestão e evitar a perda de dados na Empresa?**

## **Justificativa**

Com um Sistema de Gestão, haverá uma facilidade na definição dos serviços, concernente a gestão da empresa em geral, uma vez que tudo estará automatizado, facilitando e orientado assim os funcionários da oficina. Também compensará muito utilizar um sistema de gestão, pois é bastante ágil na consulta de respostas e esclarecimentos, melhorará o fluxo de entrada e saída de mercadorias e ajudará a ganhar tempo na execução de tarefas e na elaboração de orçamentos. Isto é, rapidez na resolução dos serviços da empresa de dentro para fora.

E pessoalmente, com esta prática, também me trará melhorias na minha aptidão no que concerne a programação.

## **Objectivos**

### **Objectivo Geral**

* Desenvolver uma página Web para um Sistema de Gestão de Serviço da Oficina DARTIN.

### **Objectivos específicos:**

* Proceder ao levantamento e análise de requisitos; completo com diagramas de classes e esquemas conceituais de base de dados.
* Proceder a Modelação do sistema utilizando a linguagem UML;
* Desenvolver o sistema;
* Testar os módulos desenvolvidos.

## **Organização do trabalho**

Para melhor enquadramento e situação no relatório, o presente trabalho foi organizado por capítulos, como se segue a abaixo:

* **Capítulo 1 –** Neste capítulo é feita uma breve introdução sobre o trabalho realizado apresentado elementos importantes como a problemática, justificativa e os objectivos a serem atingidos.
* **Capítulo 2 -** Neste capítulo abordar-se-á a fundamentação teórica onde será reservado a abordagem mais teórica sobre o projecto desenvolvido, apresentando algumas tecnologias envolvidas no projecto, arquitecturas e metodologias usadas para a construção de softwares.
* **Capítulo 3** - Este capítulo apresenta a análise metodológica, onde são apresentadas as técnicas, os procedimentos e os métodos de pesquisa usados para a elaboração deste projecto, apresentamos os diagramas UML em função da análise que foi feita, fizemos a descrição das tecnologias envolvidas.
* **Capítulo 4 -** Neste capítulo é apresentado o nosso projecto de forma funcional, a interface juntamente com alguma explicação do funcionamento do mesmo.

# FUNDAMENTAÇÃO TEÓRICA

## **Sistema de Gestão**

Para Rodriguez y Rodrigues (2002) a gestão de um modo abrangente, tem a ver com a forma como os relacionamentos entre as pessoas se estabelecem na busca de um objetivo comum.

A apresentação de forma estruturada e organizada de como ocorre a integração entre os seus sistemas internos, formais e informais, que fazem com que seja assegurado o atendimento às estratégias de negócio suportadas pelas pessoas dentro de uma Organização formal de poder e sistemas.

Um sistema de gestão é um programa de computador que ajuda a cuidar das atividades de uma empresa. Ele é um software inteligente e que tem como objetivo facilitar as atividades do dia a dia, automatizando o máximo de processos quanto for possível.

Os resultados apresentados por um sistema de gestão extrapolam o caráter financeiro. Pois podemos obter alguns benefícios da implementação de um SG, que, por conseguinte, poderão ser representados por meio de indicadores de qualidade:

* Maior segurança para os funcionários;
* Melhoria nos índices de satisfação interna;
* Aumento da qualidade de vida na comunidade na qual a empresa atua.

### **Vantagens de um sistema de gestão**

Com a delimitação de indicadores e o acompanhamento constante das métricas relacionadas à implantação de um sistema de gestão em um negócio, diversas vantagens podem ser observadas. Eis as principais:

* Aumenta a transparência;
* Diminui os riscos de acidentes de trabalho;
* Reduz a burocracia no trâmite de processos;
* Aprimora o clima organizacional da empresa;
* Reduz os danos causados ao meio ambiente;
* Torna o negócio mais competitivo e mais próximo da excelência;
* Padroniza processos em consonância aos padrões internacionais;
* Possibilita um ambiente de trabalho mais seguro, agradável e produtivo;
* Fortalece a percepção de marca tanto entre o público interno, quanto entre o externo.

### **Sistemas de Gestão para Oficinas**

Os Sistema de Gestão de Serviços para Oficinas de um modo geral é uma ferramenta muito útil pois, é possível obter informações actualizadas sobre o negócio, controlar os recursos e informações extras valiosas para auxiliar na tomada de decisões, isto tudo em tempo real.

Mas apesar da reconhecida existência no mercado de diversos sistemas de gestão de oficinas mecânicas, tais soluções nem sempre são adotadas pelos gestores daquele ramo de negócio, tendo em vista sua alta complexidade de operação e / ou os altos custos de aquisição, licença de uso e manutenção. Assim, pretende-se com este projeto demonstrar que pequenas oficinas mecânicas, diferentemente de grandes centros automotivos, também podem dispor de um sistema de gestão prático (pela facilidade de interação com o usuário), gratuito ou de custo extremamente reduzido e eficiente. Ou seja, mais controle, qualidade e produtividade no negócio com menos pessoal, tempo, tecnologia e recursos financeiros.

Mais a baixo apresentamos dois (2) exemplos de como funcionam alguns sistemas de Gestão para oficinas que serviram como base de estudo para os trabalhos relacionados:

* **PROGRAMA MECÂNICA PREMIUM**

É um software de fácil manuseio, muito prático e intuitivo com um visual limpo e de fácil compreensão para usuários iniciantes. Entre os recursos oferecidos estão: cadastro de usuários, cadastro de perfil (acessos dos vendedores), cadastro de fornecedores, cadastro de vendedores (com foto), cadastro completo de produtos, cadastro de imagens do produto, controle de desconto por produto, cadastro de produto por grade de cor e tamanho, busca de produtos por código, nome, etc, imagem do produto na tela de venda durante o pedido, estorno e trocas de produtos, tabelas de preços ajustáveis pelo usuário, consulta de preços rápida, tabela resumida e completa para vendedores externos, controle de comissões (automático), controle de comissão por produto diferenciado e controle diário do caixa (fechamento);



Figura 2. 1 Sistemas de gestão para Oficinas Mecânicas

* **OFICINA INTEGRADA**

A Oficina Integrada é um Software para controle e gerenciamento de reparadoras de veículos. Controla desde a entrada do veículo na oficina até a saída. O Sistema Integrado controla ordem de serviço e situação do veículo na oficina. Seus clientes podem consultar os serviços feitos pelo site da sua oficina.

O Oficina Integrada é um software para reparadores de veículos. O sistema foi desenvolvido pela MundomidiaLtda, empresa de tecnologia da informação.

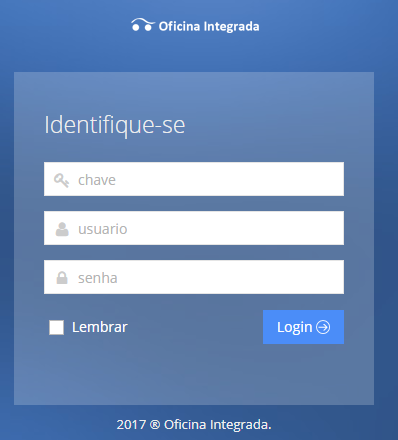


Figura 2. 2 Sistemas de Gestão Para Oficinas Integrada

A mesma dispõe das seguintes funcionalidades: Controle de estoque de peças; Ordens de Serviços com alertas aos funcionários; Gestão financeira, NFe e controle de fluxo de caixa; Cálculo de comissões por setores; Integração com o site da oficina onde o cliente pode acompanhar a Ordem de Serviço. O administrador pode ter acesso completo de onde estiver com tablet ou smartphone.

A Oficina Integrada é o sinônimo de Modernidade, facilidade e organização.

As duas soluções acima, apesar de não serem testada por nós, apresentam boas funcionalidades que facilitam a gestão das oficinas. Mas uma das grandes desvantagens é que não estão disponíveis para o nosso país e as informações requisitos utilizados para o desenvolvimento dos mesmos são bastante genéricos e viradas para o público brasileiro. Sendo assim a nossa solução é uma solução virada para o nosso país e que esta a ser desenvolvida não de forma genérica, mas sim tendo em conta a uma entidade real.

## **Engenharia de Software**

É um conjunto integrado de métodos e ferramentas utilizadas para especificar, projetar, implementar e manter um sistema.

A Engenharia de Software é a área da computação que estabelece uma abordagem sistemática de desenvolvimento de software com qualidade envolvendo processos, técnicas e ferramentas apropriadas para uma ampla gama de aplicações, considerando prazos, restrições e recursos disponíveis (PRESSMAN, 2011).

A criação da Engenharia de Software surgiu no intuito de contornar a crise do software, dando um tratamento de engenharia ao desenvolvimento de sistemas complexos caracterizados por um conjunto de componentes abstratos (estrutura de dados e algoritmos) encapsulados na forma de procedimentos, funções, módulos, objetos ou agentes e interconectados entre si, compondo a arquitetura do software, devendo ser executados em sistemas computacionais.

### **Requisitos do Sistema**

Para Silva e Videira (2001), um requisito é uma funcionalidade ou condição que o sistema deverá possuir. Para os identificar adequadamente, é aplicado um conjunto de técnicas de modo a obter a percepção detalhada daquilo que o sistema deverá efectuar.

E eles podem ser extraídos realizando reuniões com os interessados, a elaboração de questionários, a observação das actividades e do funcionamento do dia-a-dia, a recolha e análise de documentação diversa, a elaboração de pequenos protótipos do sistema que permitam validar mais facilmente a percepção obtida (seguindo o princípio que "uma imagem vale mais do que mil palavras"). Deve-se ter a preocupação de encontrar a melhor solução, pois às vezes aquilo que o utilizador pede não é sempre o que ele necessita (este facto está relacionado com o seu desconhecimento do que se pode obter de um sistema de informação). Outra questão a considerar tem a ver com a importância de identificar não apenas as funcionalidades actuais, mas sobretudo determinar a situação futura a atingir, (Silva e Videira, 2001).

#### Tipo de Requisitos

Esses requisitos também são impostos pelos diversos stakeholders do software e estão normalmente relacionados a interfaces com o usuário, capacidades, consumo de recursos e escalas de tempo.

Bennet, McRobbe Farmer (1999), identificam as seguintes categorias de requisitos:

* Requisitos funcionais - descrevem o que um sistema faz ou é esperado que faça. Estes são os requisitos que inicialmente serão levantados, abrangendo a descrição de processamentos a efectuar pelo sistema, entradas (inputs) e saídas (outputs) de informação em papel ou no ecrã que derivam da interacção com pessoas e outros sistemas.
* Requisitos não funcionais - relacionados com as características qualitativas do sistema, descrevendo a qualidade com que o sistema deverá fornecer os requisitos funcionais. Abrange medidas de desempenho como, por exemplo, tempos de resposta, volume de dados ou considerações de segurança.

### **Modelação de Sistemas**

Para Lancaster (2000), como parte dos requisitos do sistema e da actividade de projectos, o sistema precisa ser modelado como um conjunto de componentes e de relações entre esses componentes. Isso é, normalmente, ilustrado graficamente em um modelo de arquitectura de sistema, que proporciona ao leitor uma visão geral da organização do sistema.

#### Linguagem UML

UML é a sigla de Unified Modelling Language, que pode ser traduzido por Linguagem de Modelação Unificada. A UML é uma linguagem que utiliza uma notação padrão para especificar, construir, visualizar e documentar sistemas de informação orientados por objectos segundo (Nunes e O´Neill).

Particularmente no que tange à engenharia de software, a linguagem UML pode ser utilizada para modelar todas as etapas do processo de desenvolvimento de software, bem como produzir todos os artefatos de software necessários à documentação dessas etapas, segundo (Gudwin, 2015)

Um modelo em UML é constituído por um conjunto de diagramas que representam aspectos complementares de um sistema de informação. E para cada um destes diagramas são utilizados símbolos que representam os elementos que estão a ser modelados (abstracções) e linhas que relacionam esses elementos. Os símbolos e as linhas têm significado específico e possuem formas distintas, constituindo uma forma de notação, segundo (Nunes e O´Neill).

Para (Nunes e O´Neill), a UML disponibiliza o seguinte conjunto mais importantes de diagramas:

* **Diagrama de Use Case -** serve para identificar as fronteiras do sistema e descrever os serviços (use cases) que devem ser disponibilizados a cada um dos diversos utilizadores (actores);
* **Diagrama de Classes -** através do qual descrevemos a estrutura de informação (classes e suas relações) que é utilizada no sistema;
* **Diagrama de Actividade -** pode ser utilizado para descrever cada um dos use cases, realçando o encadeamento de actividades realizadas por cada um dos objectos do sistema, numa óptica de fluxo de trabalho (work-flow).
* **Diagrama de Sequência -** O diagrama de sequência é um diagrama de interacção que realça a ordem cronológica das mensagens entre objectos.

## **Base de Dados**

Para Coelho (2011) base de dados é um local onde pode ser guardada informação. A informação pode ser consultada, alterada, apagada, na totalidade ou parcialmente, através de uma aplicação conhecida como Sistema de Gestão de Base de Dados (SGBD), também chamada simplesmente de Base de Dados (BD).

Uma das principais finalidades de uma base de dados é que as informações que elas contêm sirvam para uma variedade de aplicações distintas. Para isso, é importante que a base de dados seja “substancialmente não-redundante, isto é, possuir o mínimo de duplicidade de dados idênticos, de preferência nenhuma” (Rowley, 2002, p. 125).

### **Sistemas de base de dados**

Um sistema de base de dados tenta baixar os custos de manutenção através da separação entre a forma como os dados são percebidos pelo programador e a forma como esses dados são armazenados fisicamente.

De acordo com DATE (2004), um sistema de base de dados é “um sistema computadorizado cuja finalidade geral é armazenar informações e permitir que os usuários busquem e atualizem essas informações quando as solicitar”. Para o autor um sistema de base de dados é composto por dados, hardware, software e usuários.

Para Um Sistema Gerenciador de Base de Dados (SGBD) é uma coleção de programas que permitem aos usuários criarem e manipularem uma base de dados. Um SGBD é, assim, um sistema de software de propósito geral que facilita o processo de definir, construir e manipular bases de dados de diversas aplicações:

Definir uma base de dados envolve a especificação de tipos de dados a serem armazenados na base de dados. Construir uma base de dados é o processo de armazenar os dados em algum meio que seja controlado pelo SGBD.

Manipular uma base de dados indica a utilização de funções como a de consulta, para recuperar dados específicos, modificação da base de dados para refletir mudanças no mini-mundo (inserções, atualizações e remoções), e geração de relatórios.

### **Modelo Entidade Relacionamento (MER)**

Segundo Takai, Italiano e Ferreira (2005) O MER é um modelo de dados conceitual de alto-nível, ou seja, seus conceitos foram projetados para serem compreensíveis a usuários, descartando detalhes de como os dados são armazenados.

E é actualmente usado principalmente durante o processo de projeto da base de dados. Segundo (Silva) O MER, ou simplesmente ER, foi desenvolvido com o objetivo de facilitar o projeto de base de dados por meio de um modelo independente de implementação e de fácil compreensão por parte do usuário

Os relacionamentos são descritos através da cardinalidade, que indica como as instâncias das entidades se relacionam. Os tipos utilizados na modelagem são (KORTH, SILBERCHATZ e SUDARSHAN, 2006):

* **Um-para-um (1:1):** uma instância em “A” está associada com no máximo uma instância em “B”, e uma instância em “B” está associada com no máximo uma instância em “A”;
* **Um-para-muitos (1:n):** uma instância em “A” está associada a qualquer número de instâncias em “B”, e uma instância em “B”, todavia, pode estar associado a no máximo uma instância em “A”;
* **Muitos-para-muitos (n:n):** uma instância em “A” está associada a qualquer número de instâncias em “B” e vice-versa. Alguns autores preferem chamar esta cardinalidade de m:n, por considerar que podem representar valores diferentes.

### **SQL**

A Structured Query Language (SQL) ou Linguagem de Consulta Estruturada foi criada pela IBM Research, no início da década de 1970, para o protótipo de um sistema de banco de dados chamado System R (DATE, 2004).

Apesar de conhecida como uma “linguagem de consulta”, a SQL oferece também recursos para definir a estrutura dos dados, atualizar, incluir, excluir e alterar dados, especificar restrições de integridade e outros recursos mais (SILBERSCHATZ; KORTH; SUDARSHAN, 1999).

Ainda nesta senda, de acordo com DATE (2004), a SQL possui, entre outros, os seguintes componentes:

**Linguagem de Definição de Dados (“Data DefinitionLanguage” - DDL):** é utilizada pelo DBA e projetistas de base de dados para definir seus esquemas. O SGBD tem um compilador para processar descrições em DDL e construir a descrição do esquema armazenado no catálogo;

**Linguagem de Manipulação de Dados (“Data Manipulation Language” - DML):** uma vez que o esquema é compilado e a base de dados preenchida com dados, os usuários têm que ter algum modo de manipular os dados. Manipulações comuns como recuperação, inserção, remoção e modificação de dados são realizadas pela DML (Takai, Italiano e Ferreira, 2005).

Segundo Battisti (2000), todas as instruções SQL são conduzidas com um único comando que contem uma descrição completa da informação exigida. Ao escrever uma instrução SQL, você não deve se preocupar em como os dados são recuperados, mas somente com o conteúdo do conjunto de dados. Este é o principal objectivo do SQL.

## **Processo de desenvolvimento de sistemas**

Ainda que os processos tenham de ser definidos caso a caso, de maneira geral, o ciclo de vida de um software envolve, pelo menos, as seguintes fases (Falbo e Barcellos, 2011):

* **Planeamento:** O objetivo do planejamento de projeto é fornecer uma estrutura que possibilite ao gerente fazer estimativas razoáveis de recursos, custos e prazos. Uma vez estabelecido o escopo de software, com os requisitos esboçados, uma proposta de desenvolvimento deve ser elaborada, isto é, um plano de projeto deve ser elaborado configurando o processo a ser utilizado no desenvolvimento de software. À medida que o projeto progride, o planejamento deve ser detalhado e atualizado regularmente. Pelo menos ao final de cada uma das fases do desenvolvimento (análise e especificação de requisitos, projeto, implementação e testes), o planejamento como um todo deve ser revisto e o planejamento da etapa seguinte deve ser detalhado. O planejamento e o acompanhamento do progresso fazem parte do processo de gerência de projeto.
* **Análise e Especificação de Requisitos:** Nesta fase, o processo de levantamento de requisitos é intensificado. O escopo deve ser refinado e os requisitos mais bem definidos. Para entender a natureza do software a ser construído, o engenheiro de software tem de compreender o domínio do problema, bem como a funcionalidade e o comportamento esperados. Uma vez capturados os requisitos do sistema a ser desenvolvido, estes devem ser modelados, avaliados e documentados. Uma parte vital desta fase é a construção de um modelo descrevendo o que o software tem de fazer (e não como fazê-lo).
* **Projeto:** Esta fase é responsável por incorporar requisitos tecnológicos aos requisitos essenciais do sistema, modelados na fase anterior e, portanto, requer que a plataforma de implementação seja conhecida. Basicamente, envolve duas grandes etapas: projeto da arquitetura do sistema e projeto detalhado. O objetivo da primeira etapa é definir a arquitetura geral do software, tendo por base o modelo construído na fase de análise de requisitos. Essa arquitetura deve descrever a estrutura de nível mais alto da aplicação e identificar seus principais componentes. O propósito do projeto detalhado é detalhar o projeto do software para cada componente identificado na etapa anterior. Os componentes de software devem ser sucessivamente refinados em níveis maiores de detalhamento, até que possam ser codificados e testados.
* **Implementação:** O projeto deve ser traduzido para uma forma passível de execução pela máquina. A fase de implementação realiza esta tarefa, isto é, cada unidade de software do projeto detalhado é implementada.
* **Testes:** inclui diversos níveis de testes, a saber, teste de unidade, teste de integração e teste de sistema. Inicialmente, cada unidade de software implementada deve ser testada e os resultados documentados. A seguir, os diversos componentes devem ser integrados sucessivamente até se obter o sistema. Finalmente, o sistema como um todo deve ser testado.
* **Entrega e Implantação:** uma vez testado, o software deve ser colocado em produção. Para tal, contudo, é necessário treinar os usuários, configurar o ambiente de produção e, muitas vezes, converter bases de dados. O propósito desta fase é estabelecer que o software satisfaz os requisitos dos usuários. Isto é feito instalando o software e conduzindo testes de aceitação. Quando o software tiver demonstrado prover as capacidades requeridas, ele pode ser aceito e a operação iniciada.
* **Operação:** nesta fase, o software é utilizado pelos usuários no ambiente de produção.
* **Manutenção:** Indubitavelmente, o software sofrerá mudanças após ter sido entregue para o usuário. Alterações ocorrerão porque erros foram encontrados, porque o software precisa ser adaptado para acomodar mudanças em seu ambiente externo, ou porque o cliente necessita de funcionalidade adicional ou aumento de desempenho. Muitas vezes, dependendo do tipo e porte da manutenção necessária, essa fase pode requerer a definição de um novo processo, onde cada uma das fases precedentes é reaplicada no contexto de um software existente ao invés de um novo.

### **Processo Incremental**

Para Silva e Videira (2001) a noção de processo incremental corresponde à ideia de “aumentar (ou alargar) pouco-a-pouco” o âmbito do sistema. Uma boa imagem para este atributo é a de uma mansão que foi construída por sucessivos incrementos a partir de uma primeira casa com apenas duas divisões.

Segundo Lancaster (2000), a abordagem do desenvolvimento incremental foi sugerida por Mills [Millset al., 1980] como um meio de reduzir o ´retrabalho´ no processo de desenvolvimento e de proporcionar aos clientes algumas experiencias de adiar decisões sobre seus requisitos detalhados, até que eles tenham alguma experiencia com o sistema.

Em um processo incremental, os clientes identificam, em um esboço, as funções a serem fornecidas pelo sistema. Eles identificam quais funções são mais importantes e quais são menos importantes para eles. Em seguida é definida uma série de estágios de entrega, com cada estágio fornecendo um subconjunto das funcionalidades do sistema. A alocação de funções aos estágios depende da prioridade da função. As funções prioritárias são entregues primeiramente ao cliente, (Lancaster, 2000).

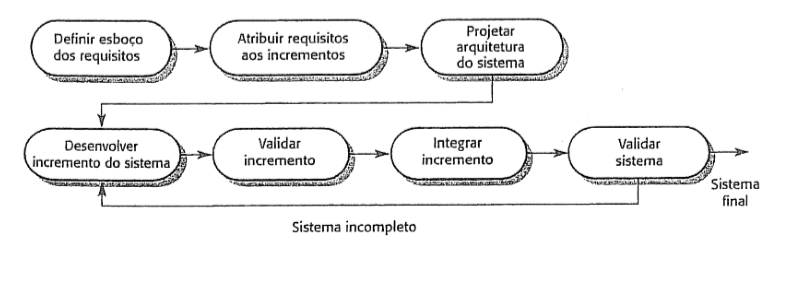


Figura 2. 3 Processo Incremental

**Fonte: Vasconcelos, Rouiller, Machado, Medeiros(2006)**

Segundo Vasconcelos, Rouiller, Machado, Medeiros(2006),as principais vantagens do modelo incremental são:

* A funcionalidade do sistema estará disponível mais cedo, pois ela é entregue a partir dos incrementos;
* Incrementos iniciais agem como um protótipo para ajudar a elicitar requisitos para incrementos finais;
* Diminuem-se os riscos de falhas no projeto como um todo;
* Os serviços de prioridade mais alta do sistema tendem a receber mais testes.

## **Qualidade de Software**

Qualidade de software é a conformidade dos requisitos funcionais e de desempenho explicitamente declarados, a padrões de desenvolvimento claramente documentados e a características implícitas que são esperadas de todo software profissionalmente desenvolvido (Pressman, 1995)

A avaliação da qualidade de software pode ser realizada em dois momentos: durante a geração do software e após este estar pronto para o uso, chamando esses dois momentos, respectivamente, de processo e produto.

No processo procura-se avaliar de que forma o software está sendo desenvolvido, identificando práticas que possam conduzir a problemas na qualidade do produto e desenvolvendo e/ou utilizando métodos e ferramentas que evitem esses problemas. Já no produto concluído, procura-se avaliar a sua qualidade a fim de identificar deficiências e limitações em sua aplicabilidade como um produto final.

E de acordo com Shiba (1997), deve-se considerar alguns aspectos para se obter a qualidade do produto:

* **Funcionalidade:** identifica os procedimentos de funcionamento de um produto;
* **Confiabilidade:** o produto não deve apresentar problemas junto aos clientes, caso contrário, o fornecedor deverá resolvê-los;
* **Usabilidade:** deve-se testar o máximo possível o produto e constatar o resultado como satisfatório;
* **Eficiência:** comprovação, pelo cliente, de sua satisfação com o produto;
* **Manutenibilidade:** garantia de correções dos problemas. 6. Portabilidade: o produto muda de ambiente e a operação ocorre da mesma forma satisfatória.

## **Arquitectura de Software**

Uma arquitetura de software envolve a descrição de elementos arquiteturais dos quais os sistemas serão construídos, interações entre esses elementos, padrões que guiam suas composições e restrições sobre estes padrões segundo. (PFLEEGER, 1998).

A arquitetura de um software consiste na definição de seus componentes, as propriedades externamente visíveis destes elementos e os relacionamentos entre eles, enfatizando a separação dos interesses. (BASS, 2003).

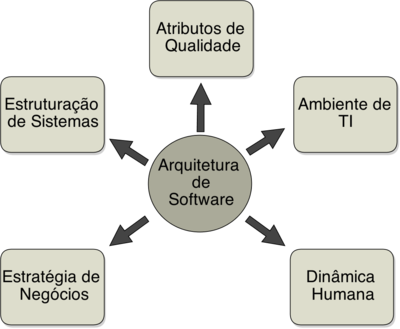


Figura 2. 4 Arquitetura de Software

**Fonte: lobotech.com.br**

### **Linguagem de Programação**

Na programação de computadores, uma linguagem de programação serve como meio de comunicação entre o indivíduo que deseja resolver um determinado problema e o computador.

Uma Linguagem de Programação é uma linguagem destinada a ser usada por uma pessoa para expressar um processo através do qual um computador pode resolver um problema. Três categorias principais de linguagem de programação têm sido desenvolvidas: linguagem de máquina, linguagens assembly e linguagens de alto nível. Onde apenas os primeiros computadores somente podiam ser programados usando linguagem de máquina. Linguagem essa que usa sequências de zeros e uns (bits) que representam instruções precisas para computação e acessos de dados.

### **Linguagem C#**

Neste capítulo falaremos de uma das linguagens usadas no desenvolvimento do nosso trabalho, que é a linguagem C#.

O C# é uma linguagem orientada a objetos com a qual podemos criar classes que podem ser utilizadas por outras linguagens como, por exemplo, o Visual Basic. Uma característica importante é que ainda é possível utilizar os componentes COM, facilitando assim uma rápida migração para um ambiente de desenvolvimento de alto nível sem precisar reescrever todas as aplicações que você possui.

Segundo Bittuti(), o C# é uma linguagem da Microsoft, apresentada juntamente com o Framework .NET. O C# foi construído com base nos conceitos de Orientação a objectos.

Para (Ferguson, Patterson, Beres, Boutquin, Gupta) 2003, o C# proporciona vários operadores e que lhe permitem escrever expressões matemáticas e de bits. Muitos (Mas nem todos) destes operadores podem ser redefinidos. Permitindo-lhe mudar a forma em que trabalham estes operadores.

C# admite uma larga lista de expressões que lhe permitem definir varias rotas de execução dentro do código. As instruções de fluxo de controlo que usam palavras chave como *if, switch, while, for, break e continue* permitem ao código ramificar-se por caminhos diferentes. Dependendo dos valores de suas variáveis. As classes podem conter códigos e dados. Cada membro de uma classe tem algo chamado âmbito de acessibilidade. Que define a visibilidade do membro com respeito a outros objectos. C# admite os âmbitos de acessibilidade *public, protected, internal e private* (Ferguson, Patterson, Beres, Boutquin, Gupta, 2003)

### **Programação em camadas**

Para este ponto é importante frisar que usaremos a programação em 3 camadas. Pois muitas são as vantagens em usar este modelo, em relação a facilidade de gerenciamento e actualização das aplicações.

A ideia básica do modelo em 3 camadas é retirar as Regras do Negócio, da aplicação Cliente e centraliza-las em um determinado ponto, o qual é chamado de Servidor de Aplicações. O acesso à base de dado é feito através de regras contidas no Servidor de Aplicações. Ao centralizar as Regras de Negócio em um único ponto, fica mais fácil a actualização das mesmas (Battisti, 2000).

Para Battisti (2000), todo o acesso do cliente, aos dados do servidor de base de dados, é feito de acordo com as regras contidas no Servidor de Aplicações. O cliente não tem acesso aos dados do servidor de Base de dados, sem antes passar pelo servidor de aplicações. Com isso, as três (3) camadas são as seguintes:

* **Camada de Apresentação:** Continua no programa instalado. Alterações na Interface do programa ainda irão gerar a necessidade de actualizar a aplicação em todas as estações de trabalho, onde a aplicação estiver sendo utilizada. Porém cabe ressaltar que alterações na interface são menos frequentes nas regras de negócio;
* **Camada de Lógica de Negócio:** São regras do negócio, as quais determinam de que aneira os dados serão utilizados e manipulados. Desta maneira, quando uma regar do negócio for alterada, basta actualizá-la no Servidor de Aplicações. Apos a actualização, todos os usuários passarão a ter acesso à nova versão, sem que seja necessário reinstalar o programa em cada um dos computadores da rede;
* **Camada de Acesso a Dados:** Nesta camada temos o servidor de Base de Dados, no qual reside toda a informação necessária para o funcionamento da aplicação. Cabe reforçar que os dados são acessados somente através do Servidor de Aplicação, e não directamente pela aplicação cliente.

## **Linguagens e Tecnologias Web**

Linguagens Web, é um ponto do desenvolvimento web que abrange todas as linguagens de programação web, as mais usadas hoje são:

**HTML -***HyperText Markup Language***:** é a linguagem usada para descrever e definir o conteúdo de uma página Web em um formato estruturado. A linguagem de programação HTML libera permissão a máquinas diferentes de ler documentos da Internet, por meio do protocole HTTP, dando acesso a documentos de um único endereço na rede (chamado protocole HTTP, dando acesso a documentos de um único endereço na rede (chamado URL). (KIOSKEA.NET, 2008).

**Java script:**o Java script tem a possibilidade de criar pequenos detalhes (programas) em uma página web ou em programas orientados a objetos.

É uma linguagem de script que quando agregado nos tag's Html, permite também modernizar (incrementar) a apresentação e interatividade de páginas Web.

Características Java Script segundo SANTOS (2006):

* Permite acessar os objetos do browser;
* Código integrado na página HTML;
* Código interpretado pelo browser no momento da execução;
* Códigos de programação simples, mas para aplicações limitadas;
* Confidencialidade do código é nulo (Código é visível por ser client side),
* Criar efeitos especiais nas paginas e definir interatividades com o usuário.

### **ASP.NET**

Falando um pouco do AS.NET, esta plataforma introduz uma camada de abstração que nos permite trabalhar com valores obtidos através de pedidos http no lado servidor. Uma das vantagens decorrentes da utilização desta plataforma reside no facto de esta conseguir transformar eventos cliente (gerados do browser) em eventos servidor que podem ser tratados através da adição de código escrito numa das linguagens da plataforma .NET (tudo isto de forma quase transparente para o programador). (Abreu, 2013).

Bem, segundo Abreu (2013) a geração de eventos servidor é conseguida através da utilização de formulários e controlos servidor (elementos anotados com o atributo runat = “server”), que geram o código HTML da página carregada no browser e que são responsáveis por iniciar um novo pedido a partir do cliente para a própria pagina no lado servidor. E estas operações são designadas por POSTBACKS. A plataforma garante ainda que os estados dos controlos no lado servidor são mantidos (e actualizados, quando necessário) entre pedidos.

### **CSS**

Então surgiu o CSS, que é uma outra linguagem, separada do HTML, como objetivo único de cuidar da estilização da página. A vantagem é que o CSS é bem mais robusto que o HTML para estilização, como veremos. Mas, principalmente, escrever formatação visual misturado com conteúdo de texto no HTML se mostrou algo bem impraticável. O CSS resolve isso separando as coisas; regras de estilo não aparecem mais no HTML, apenas no CSS.

Uma folha de estilo CSS (folha de estilo em cascata) não e XHTML, mas sim, um conjunto informações sobre a formatação e exibição dos componentes do layout de uma pagina WEB. Ela e um código separado que altera as características da pagina. O CSS deve ser utilizado para liberar do XHTML ou do HTML o peso da responsabilidade da apresentação.

# METODOLOGIA

Actualmente nas empresas é necessário que se tenha algum nível de processo visando como objectivo a qualidade no desenvolvimento de software.

## **Metodologia de Investigação Científica (Metodologia de Pesquisa)**

O tipo de pesquisa usada neste trabalho foi a Entrevista e levantamento bibliográfico.

Pois ela vai além da observação dos factos e fenômenos, e faz uma coleta do que ocorre na realidade a ser pesquisada. Depois disso, elas são analisadas e seus dados são interpretados com base em uma fundamentação teórica sólida com o desígnio de elucidar o problema pesquisado.

## **Descrição do Campo de estudo**

O campo de estudo foi realizado na empresa Dartin, localizada em Viana/Zango I, a mesma que oferece serviços de oficina auto e frio industrial, constituída por 20 funcionários dos quais estão divididos por áreas específicas:

* Área mecânica;
* Área de frio;
* Área de lavagem de automóveis;
* Área de canalização;
* Área de bate-chapa;
* Área de electricidade doméstica;

## **Processo de Desenvolvimento**

Para esse sistema seguiu-se a seguinte fase de desenvolvimento:

* **Levantamento de requisitos:** Neste estágio fez-se a recolha de informações relativas ao projecto que se queria desenvolver. Estas informações foram adquiridas na sua maioria de entrevistas feitas com o gestor e funcionários da empresa para o desenvolvimento do software em questão.
* **Análise de requisitos:** A Análise de Requisitos é a primeira actividade técnica no desenvolvimento do software, e pode ser entendida como responsável por definir os serviços que um sistema deve realizar, sua interface com os demais elementos e sob quais restrições o sistema deve operar. Os requisitos dos sistemas devem estabelecer o que o sistema deve fazer ao invés de como isto será feito.
* **Projecto:** nesta fase determinamos como o sistema funcionará para atender aos requisitos, de acordo aos recursos tecnológicos existentes.
* **Implementação:** nesta fase, ocorre a codificação do sistema, através do uso de uma ou mais linguagens de programação.
* **Testes:** Esta é a fase em que os testes para verificação do sistema construído são desenvolvidos, levando-se em conta a especificação feita na fase de projecto. O principal produto dessa fase é o relatório de testes, contendo informações sobre erros detectados no software.
* **Implementação:** Esta é a fase em que o sistema é empacotado, distribuído e instalado no ambiente do usuário, os manuais do sistema são escritos, os arquivos carregados, os dados são importados para o sistema e os usuários são treinados para utilizar o sistema correspondente.

## **Análise de Requisitos**

A análise de requisitos é fundamental para o desenvolvimento de sistemas, pois trata de descobrir o que o cliente quer com o sistema. Esta mesma análise está associada ao processo de descoberta das operações que o sistema deve realizar e quais são as restrições que existirão no mesmo. Esta análise recebe o nome de Requisitos Funcionais e Requisitos não Funcionais.

### **Requisitos Funcionais:**

Um requisito é tratado como funcional quando descreve um serviço ou função que o sistema deve realizar.

Para o nosso sistema extraiu-se os seguintes requisitos funcionais:

|  |  |
| --- | --- |
| ID | Descrição |
| RF1 | Cadastrar clientes; |
| RF2 | Cadastrar funcionários; |
| RF3 | Permitir a inclusão, alteração e a exclusão de clientes; |
| RF4 | Permitir a inclusão, alteração e exclusão de produtos/serviços; |
| RF5 | Permitir o cadastro de veículos; |
| RF6 | Permitir o cadastro de serviços; |
| RF7 | Possuir um controle de acesso; |
| RF8 | Fornecer relatórios; |
| RF9 | Permitir a solicitar serviços; |
| RF10 | Permitir a verificação do estado de serviço; |
| RF11 | Permitir a atribuição de serviços aos específicos funcionários; |
| RF12 | Permitir a desabilitação de serviços; |
| RF13 | Permitir o registro de pagamentos; |
| RF14 | Calcular o preço de serviços; |
| RF15 | Permitir a visualização serviços; |
| RF16 | Permitir a consulta do estado do serviço; |

Tabela 3. 1 Requisitos Funcionais

### **Requisitos não funcionais**

Os Requisitos não funcionais estão relacionados ao uso da aplicação em termos de segurança, usabilidade, desempenho, disponibilidade e portabilidade. Para o nosso sistema extraiu-se os seguintes requisitos não funcionais:

|  |  |
| --- | --- |
| Segurança | * Deverá constar no sistema a funcionalidade de login; |
| * O sistema deve permitir que cada conta de utilizador tenha um nível de acesso; |
| * Só os usuários autorizados deveram acessar o sistema; |
| * O sistema deverá também fazer um controlo de fluxo de caixa de contabilidade; |
| Usabilidade | * O sistema dará um alerta a empresa quando um cliente estiver completando aniversário. |
| * O sistema será de fácil uso pois os menus deverão ser de fácil acesso. As telas poderão se adaptar aos formatos de várias telas devido a utilização do bootstrap |
| * A aplicação dará uma espécie de alerta, quando chegasse o tempo de fazer a manutenção da viatura, gerador ou aparelhos de ar condicionados; |
| Desempenho | * O tempo de resposta não deve ultrapassar 5 segundos; |
| * O sistema deverá ser capaz de ter um tempo de respostas alto para emitir as facturas; |
| Portabilidade | * O sistema deve rodar em qualquer navegador; |
| Confiabilidade | * O sistema deve ter um plano de contingência que permita a recuperação de dados; * O sistema não deve permitir a duplicidade de dados; |

Tabela 3. 2 Requisitos Não Funcionais

### **Regras de negócio do nosso sistema**

As regras de negócio definem como uma empresa funciona, normalmente são políticas, validações, restrições ou condições específicas. Tais regras podem determinar um comportamento do sistema. No quadro abaixo estão descritas todas regras de negócio.

|  |  |
| --- | --- |
| Código | Descrição |
| RN -01 | Obrigatoriedade no prazo de entrega; |
| RN -02 | É obrigatório que o cliente faça o pagamento de 70% do valor total dos serviços requisitados; |
| RN -03 | Apos a entrega só é considerada a reclamação no prazo de duas semanas no máximo; |

Tabela 3. 3 Regras de Negócio

## **Diagramas UML do nosso Sistema**

Neste tópico falaremos sobre os diagramas UML usados para a elaboração do sistema.

### **Diagrama de Casos de Uso**

O Diagrama de Casos de Uso tem o objectivo de auxiliar a comunicação entre os analistas e o cliente. Um diagrama de Caso de Uso descreve um cenário que mostra as funcionalidades do sistema do ponto de vista do usuário. O cliente deve ver no diagrama de Casos de Uso as principais funcionalidades de seu sistema.

O diagrama de Caso de Uso é representado por:

* Actores;
* Casos de Uso.

O relacionamento entre estes elementos. pode se dar por:

* + Associações entre actores e Casos de Uso:
  + Generalizações entre actores;
  + Generalizações, extends e includes entre os casos de uso.

Para esse sistema, como mostra a figura 6, foi criado o seguinte diagrama abaixo ilustrado, composto ele por quatro (3) actores, dez (10) casos de uso, treze (12) relacionamentos.

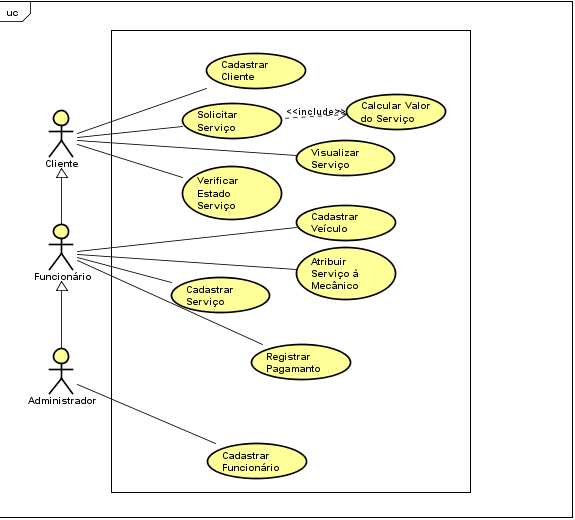


Figura 3. 1 Diagrama de Caso de Uso do projecto.

### **Descrição dos principais casos de uso do nosso sistema**

Na tabela 3.4 apresentamos a descrição do caso de uso Cadastrar Clientes.

|  |  |
| --- | --- |
| UC:UC2: | Cadastrar Cliente |
| Pré-Condição | * Estar logado. |
| Pós-Condição | * O usuário deverá estar cadastrado no sistema. |
| Fluxo Principal | * Inserir os dados; * Submeter dados; * Validar dados; * Guardar dados; * Informar o cliente sobre a operação. |
| Fluxo Alternativo | Erro no ponto 3: Dados inválidos;  4- Informar o utilizador sobre o erro e retomar ao ponto 1. |

Tabela 3. 4Descrição do Caso de Uso Cadastrar Cliente

Na tabela 3.5 apresentamos a descrição do caso de uso atribuir serviços ao mecânico.

|  |  |
| --- | --- |
| UC: UC1 | Atribuir Serviço ao Mecânico |
| Actor | Mecânico Chefe |
| Pré-Condição | * O Usuário deve estar logado; |
| Pós-condição | * Um serviço deve ser atribuído ao mecânico; |
| Fluxo Principal | * Seleccionar/Verificar Serviço; * Disponibilidade do mecânico; * Inserir dados; * Validar dados; * Guardar dados, * Informar o usuário sobre a operação. |
| Fluxo Alternativo | Erro no ponto 4: Dados inválidos;  5- Informar o utilizador sobre o erro e retomar ao ponto 3. |

Tabela 3. 5Descrição do Caso Atribuir serviço ao mecânico

Na tabela 3.6 apresentamos a descrição do caso de uso de como um serviço é visualizado.

|  |  |
| --- | --- |
| UC: UC3 | Visualizar Serviço |
| Actores | * Anonimo, Cliente; |
| Pré-Condição | * Estar ou não logado; |
| Pós- Condição | * Os serviços deverão ser visualizados; |
| Fluxo Principal | * Solicitar página de serviços; * Pesquisar serviço; * Processar e validar dados da pesquisa; * Apresentar serviço pesquisado ao utilizador. |
| Fluxo Alternativo | Erro encontrado no ponto 3: Serviço não encontrado;   * Informar o utilizador sobre o erro e voltar ao ponto 2. |

Tabela 3. 6 Descrição do Caso de Uso Visualizar Serviço

Na tabela 3.7 apresentamos a descrição do caso de uso Solicitar um Serviço.

|  |  |
| --- | --- |
| Identificador | [Solicitar Serviço] |
| Descrição | * Este Caso de Uso é responsável pelas marcações de serviços feitas pelos clientes. |
| Actores | * Cliente |
| Prioridades | * Essencial |
| Pré- Condições | * Estar no sistema |
| Fluxo Principal | * Solicitar pagina de todos serviços cadastrados no sistema; * Seleccionar o serviço pretendido; * Enviar solicitação; * Validar informação; * Guardar solicitação; * Informar ao cliente sobre a operação. |
| Fluxo Alternativo | Erro encontrado no ponto 4: Serviço inactivo;   * Informar ao utilizador sobre o erro e voltar ao ponto 2. |

Tabela 3. 7 Descrição do Caso de Uso Solicitar Serviço

Na tabela 3.8 apresentamos a descrição do caso de uso de como registrar um pagamento.

|  |  |
| --- | --- |
| UC:UC5: | Registrar Pagamento |
| Actores | * Responsável de Finanças, Administrador; |
| Pré- Condição | * Deverão estar logados; |
| Pós- Condição | * Um pagamento terá de ser registrado no sistema; |
| Fluxo Principal | * Solicitar pagina de registro de pagamentos; * Seleccionar os tipos de pagamentos; * Escolher a opção desejada pelo cliente; * Terminar e guardar a operação; * Informar ao cliente sobre a operação. |
| Fluxo Alternativo | Erro encontrado no ponto 3: Tipo de pagamento não encontrado;   * Informar ao utilizador sobre o erro e voltar ao ponto 2. |

Tabela 3. 8 Descrição do Caso de Uso Registrar Pagamento.

### **Matriz de Rastreabilidade**

A matriz de rastreabilidade é uma tabela na qual constam os requisitos funcionais e os casos de uso do sistema. Ela descreve de uma forma simples a relação entre os requisitos funcionais e os casos de uso.

A tabela 3.9 apresenta a matriz de rastreabilidade do nosso sistema.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | UC1 | UC2 | UC3 | UC4 |
| RF1 | X |  |  |  |
| RF2 |  |  |  | X |
| RF3 |  |  |  |  |
| RF4 |  |  |  |  |
| RF5 |  |  |  | X |
| RF6 |  |  | X |  |
| RF7 |  |  |  |  |
| RF8 |  |  |  |  |
| RF9 |  | X |  |  |
| RF10 |  | X |  |  |
| RF11 |  |  | X |  |
| RF12 |  |  |  |  |
| RF13 |  |  | X |  |
| RF14 |  | X |  |  |
| RF15 | X | X |  |  |
| RF16 |  | X |  |  |

Tabela 3. 9 Matriz de rastreabilidade

### **Diagrama de Actividades**

Em linguagem UML é um diagrama de actividade que representa os fluxos do trabalho passo-à-passo do negócio e operacionais dos componentes em um sistema. Ele mostra o fluxo de controlo geral. Para este sistema fez-se os seguintes diagramas de Actividades representados pelos principais casos de uso do diagrama de casos de uso.

A figura 3.2 é referente a um diagrama aonde representamos o fluxo de actividades necessários para a realização do caso de uso cadastrar cliente.

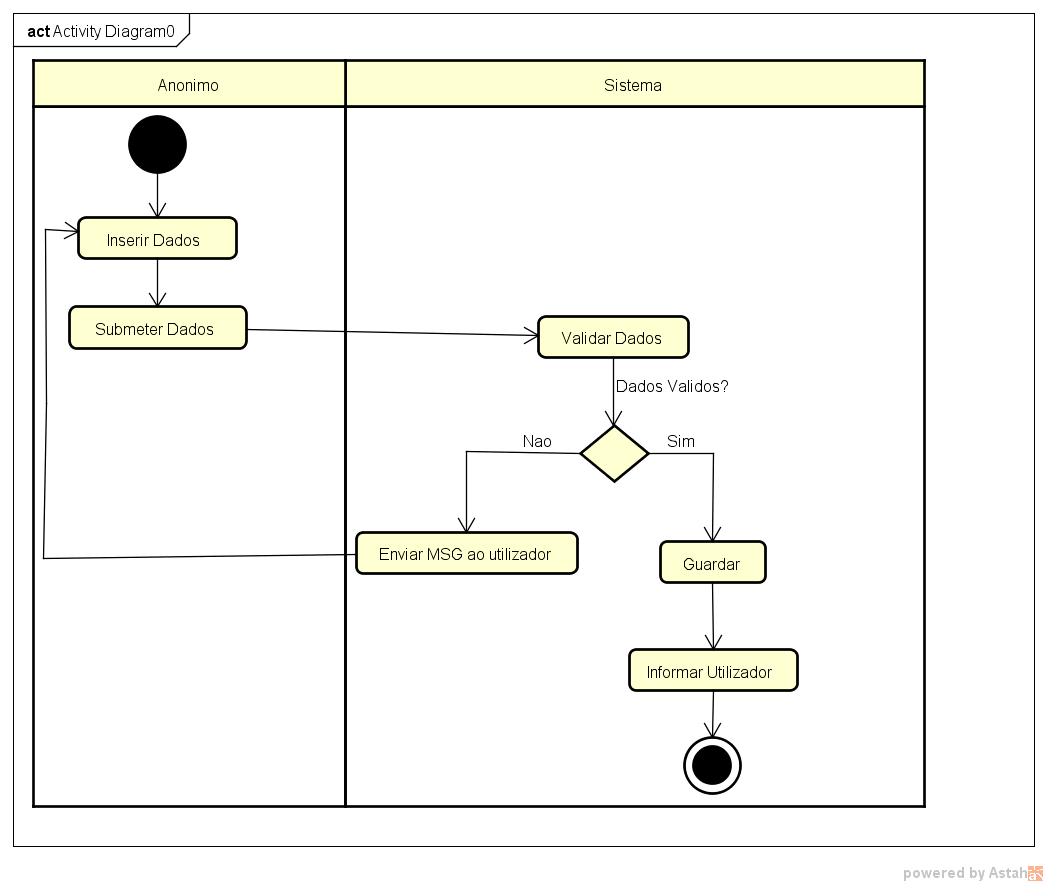


Figura 3. 2 Diagrama de Actividade Cadastrar Cliente.

A figura 3.3 é referente a um diagrama aonde representamos o fluxo de actividades necessários para a realização do caso de uso Visualizar um Serviço.

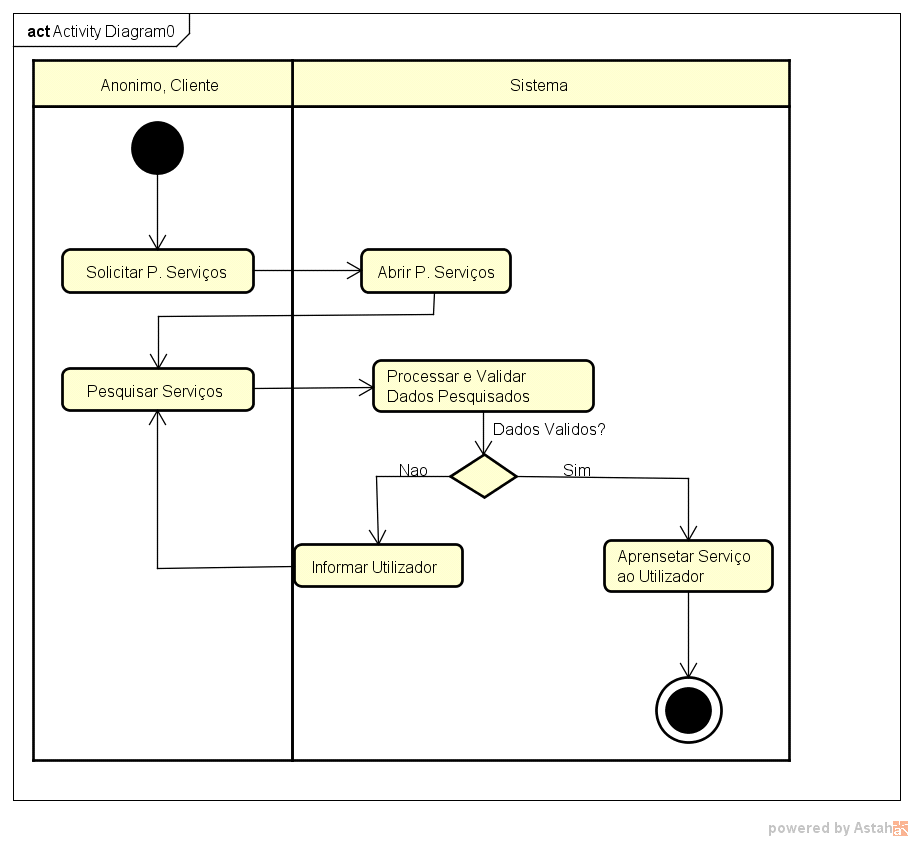


Figura 3. 3 Diagrama de Actividade Visualizar Serviço.

### **Diagrama de Sequência**

Os diagramas de sequência permitem representar o comportamento dinâmico do sistema, nomeadamente as interacções entre objectos. Os diagramas de sequência são uma das formas preferenciais de especificar os casos de uso. O diagrama consiste essencialmente na representação de sequência de chamada de métodos (troca de mensagens) entre os objectos das classes que suportam a aplicação.

Desta forma temos abaixo ilustrado alguns digramas de sequência que representam processos relevantes no nosso sistema.

Na figura 3.4 é representado a sequência de processos para que se realize o caso de uso Cadastrar Cliente.

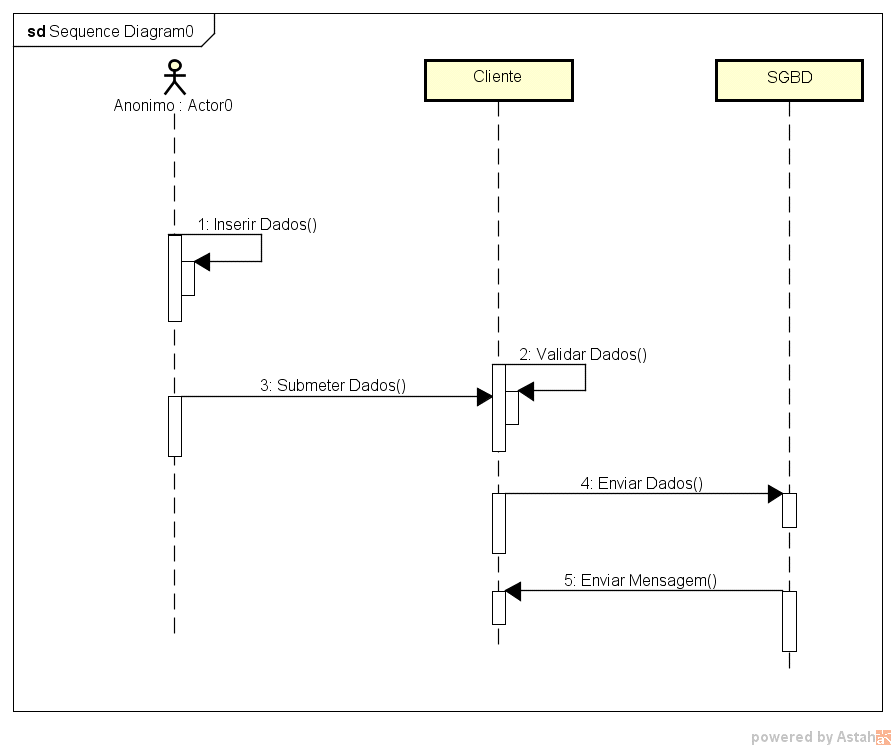


Figura 3. 4 Diagrama de Sequência Cadastrar Cliente.

Na figura 3.5 é representado a sequência de processos para que se realize o caso de uso Visualizar um Serviço.

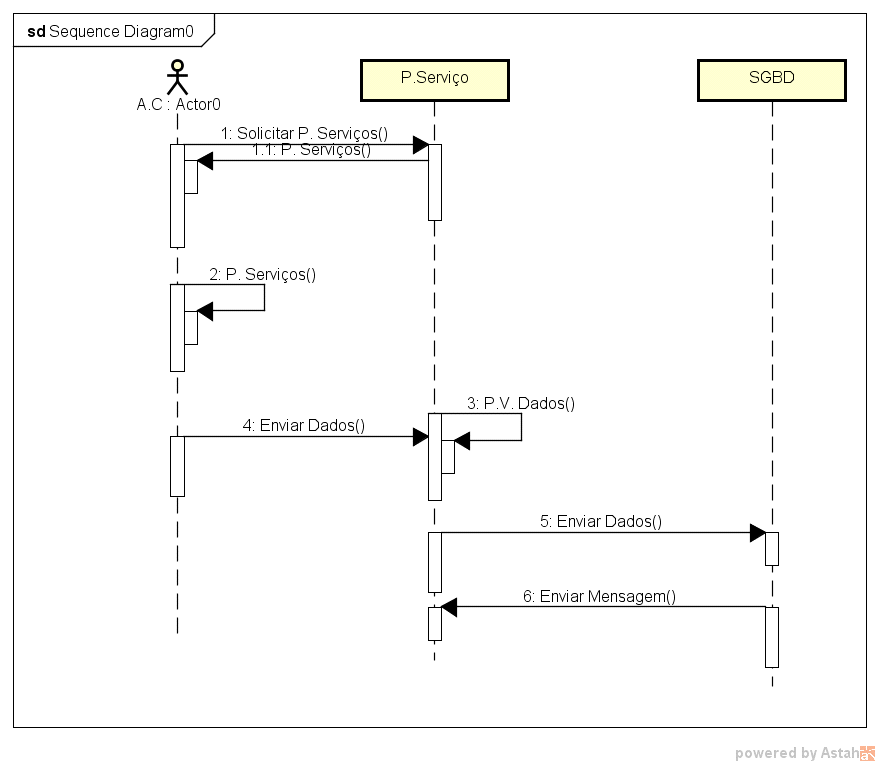


Figura 3. 5 Diagrama de Sequência Visualizar Serviço.

### **Diagrama de Classe**

Os diagramas de classe nos permitem denotar o conteúdo estático e os relacionamentos de classes. Em um diagrama de classe, também podemos mostrar se uma classe herda de outra ou se contém uma referência para outra. Em resumo, podemos retratar todas as dependências de código-fonte entre as classes.

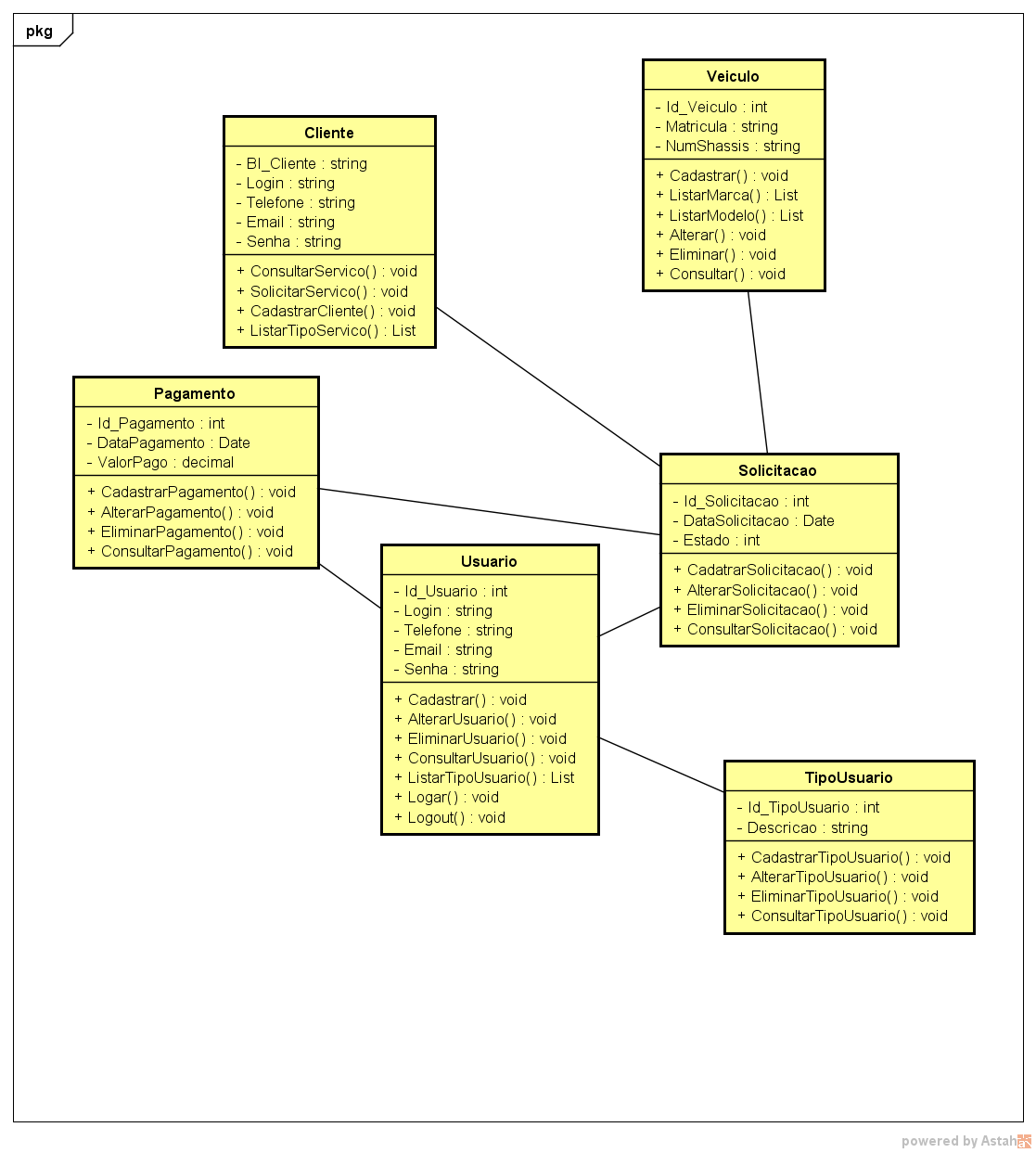


Figura 3. 6 Diagrama de Classe

### **Diagrama de Instalação**

A figura 3.7 apresenta o diagrama de como o sistema poderá ser instalado. A base de dados será alojada num servidor de base de dados com o SQL Server 2008 e a aplicação deverá estar instalado num servidor web IIS a comunicação entre o servidor de base de dados e do servidor web deverá ser feito através do protocolo TCP/IP pois os mesmos estão alojados na mesma rede. O cliente poderá ter acesso a aplicação através de browser com requisições http.

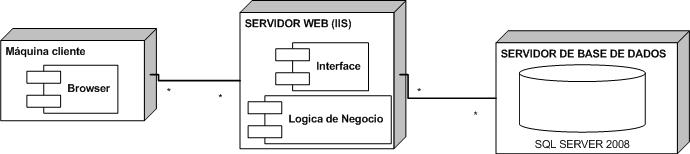


Figura 3. 7 Diagrama de instalação do sistema.

### **Modelo conceitual**

Denominamos entidade estes elementos. Atribuímos a cada entidade definidos atributos pertinentes ao sistema. Desta forma, podemos representar conceitualmente como entidades aqueles elementos no qual gostaríamos de armazenar dados que por sua vez, através do relacionamento representaremos o tipo de relação existente entre as entidades, logo a seguir na figura 3.8.

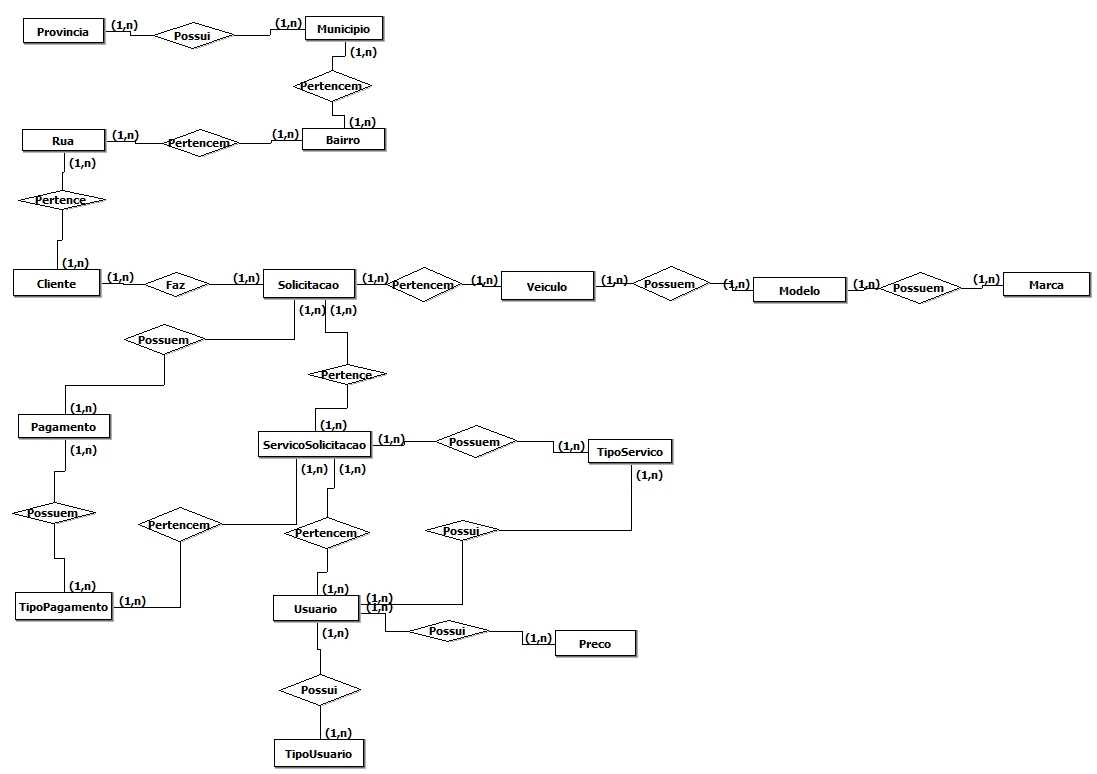


Figura 3. 8 Modelo Conceitual

### **Modelo lógico**

Na figura abaixo apresentamos o diagrama lógico que compreende uma descrição das estruturas que serão armazenadas no banco e que resulta numa representação gráfica dos dados de uma maneira lógica, inclusive nomeando os componentes e ações que exercem uns sobre os outros

A relação entre a tabela Género e a tabela Usuário: A relação entre a tabela Género e a tabela Usuário é de 1..n (lê-se um para muitos) sendo que a coluna Id\_genero da tabela Género, é uma chave estrangeira (ForeignKey - FK) na tabela Usuário. Partindo do pressuposto que um género pode ser atribuído a vários usuários.

A relação entre a tabela Cliente e tabela Serviço: A relação entre estas tabelas é de 1..n, sendo que a coluna Id\_Servico é chave estrangeira (ForeignKey- FK) na tabela Cliente. Pois um cliente pode solicitar vários serviços.

A relação entre a tabela Cliente e tabela Bairro: A relação elas é de 1..n, sendo que a coluna Id\_Bairro da tabela Bairro é chave estrangeira (ForeignKey – FK) na tabela Cliente. Pois um bairro pode pertencer à vários clientes.

A relação entre a tabela Serviço e tabela TipoServiço: Estas têm uma relação de 1..n, sendo que o Id\_TipoServiço da tabela TipoServico é chave estrangeira (ForeignKey – FK) na tabela Servico. Pois um serviço pode ter várias categorias.

A relação entre a tabela Preco e a tabela TipoServico: elas estão relacionadas de 1..n, sendo que o Id\_Preco da tabela Preco é chave estrangeira (ForeignKey) na tabela TipoServico. E que um preço pode ser pra vários tipos de serviços.

Na figura 3.9 veremos o diagrama de Entidade e Relacionamento para o nosso sistema:

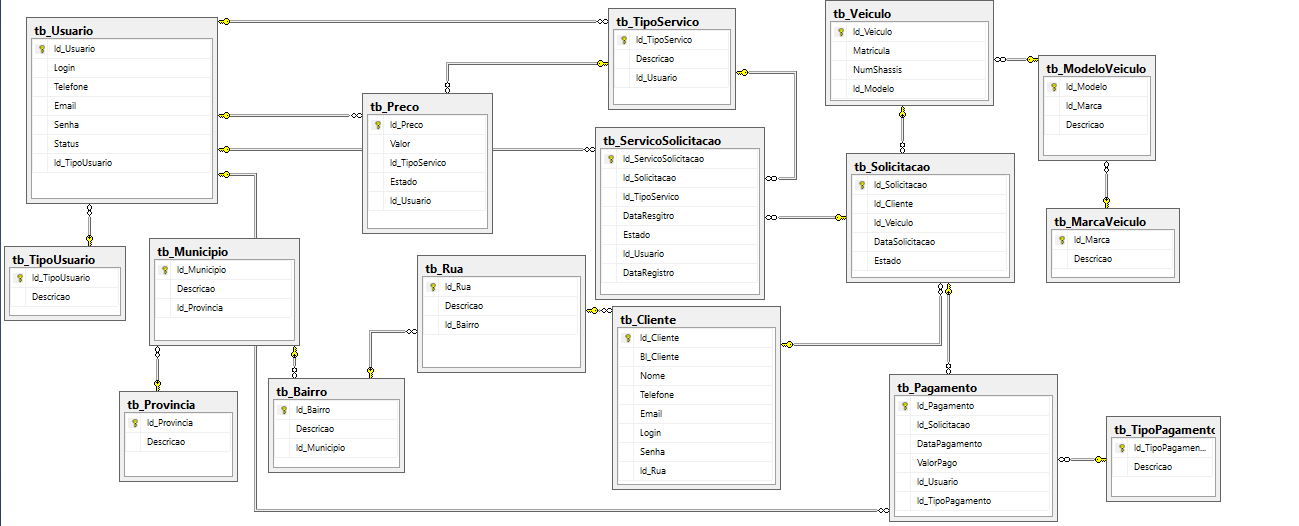


Figura 3. 9 Diagrama Entidade Relacional

## **Ferramentas utilizadas**

Para a modulação deste projecto usamos as seguintes ferramentas Microsoft como: ASP.NET, C#, SQL SERVER.

* **O Visual Studio** - é um conjunto completo de ferramentas para construir aplicativos desktop e aplicativos web empresariais desenvolvidos por equipas. Além de criar aplicativos individuais de alto desempenho, você pode usar as poderosas ferramentas de desenvolvimento baseados em componentes do Visual Studio e outras tecnologias para simplificar o trabalho em equipa no projecto, desenvolvimento e implementação de soluções empresariais.
* **My SQL Server 2014** - é um SGBD é um sistema de gerenciamento de base de dados, que utiliza a linguagem SQL (Linguagem de Consulta Estruturada, do inglês Structured Query Language) como interface. É actualmente uma das bases de dados mais populares criado pela Microsoft.
* **Asp.Net** - é a plataforma da Microsoft para o desenvolvimento de aplicações Web e é o sucessor da tecnologia ASP. Permite através de uma linguagem de programação integrada na .NET Framework criar páginas dinâmicas.
* **C#** - é uma linguagem de programação interpretada, multiparadigma, fortemente ‘tipada’, e possuindo paradigmas de programação imperativa, funcional, declarativa, orientada à objectos e genérica, C# foi desenvolvida pela Microsoft como parte da plataforma .Net.

# RESULTADOS

## **Descrição do projecto**

Como já mencionamos, este projecto consiste em desenvolver um sistema de gestão de serviços para a Empresa DARTIN.LDA, possui 4 módulos que são: Portanto o projecto permitirá que os clientes consultem, solicitem os serviços sem ter que sair da sua residência, a partir de um computador, com um browser, e que tenha acesso a internet.

Procuramos seguir todos requisitos que são exigidos pelo sistema de gestão. No nosso projecto procuramos integrar elementos programáticos de alto nível, no que tange ao designer procuramos seguir os padrões de usabilidade, de modo a tornar a experiência do usuário mais fácil.

O principal foco do sistema é permitir o feedback entre os clientes e os funcionários da Empresa. Categorizamos os usuários por perfis, de modo a fazer a separação de privilégio por omissão.

## **Interface do Sistema**

### **Página Inicial do nosso sistema**

A figura 18 mostra a página inicial do sistema, pagina esta que estará disponibilizada à todo visitante. Onde ele poderá saber tudo sobre a Empresa, ver os Serviços que nela são prestados, sua localização e possivelmente se cadastrar caso queira solicitar algum serviço que veremos no diagrama abaixo.

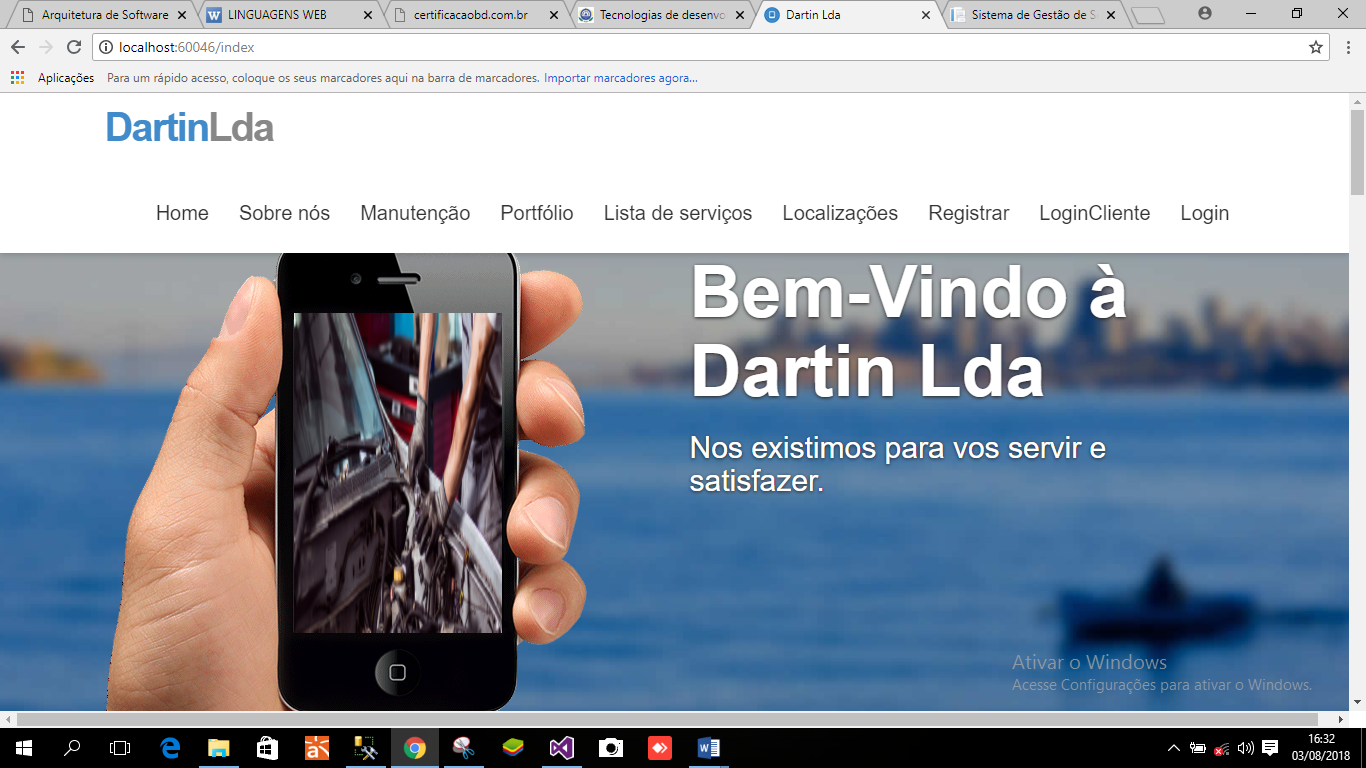


Figura 4. 1 Página inicial

### **Página de Cadastro de Solicitações**

A figura 4.2 mostra-nos como os Cadastros de Solicitações de Serviços são feitos, função essa que apenas um funcionário pode executar.



Figura 4. 2 Cadastro de solicitações de Serviços

### **Página de Cadastro de Usuários**

A figura 4.3 mostra como os procedimentos a seguir para que um usuário possa ser cadastrado no sistema usuários, função essa em que apenas o funcionário pode executar.

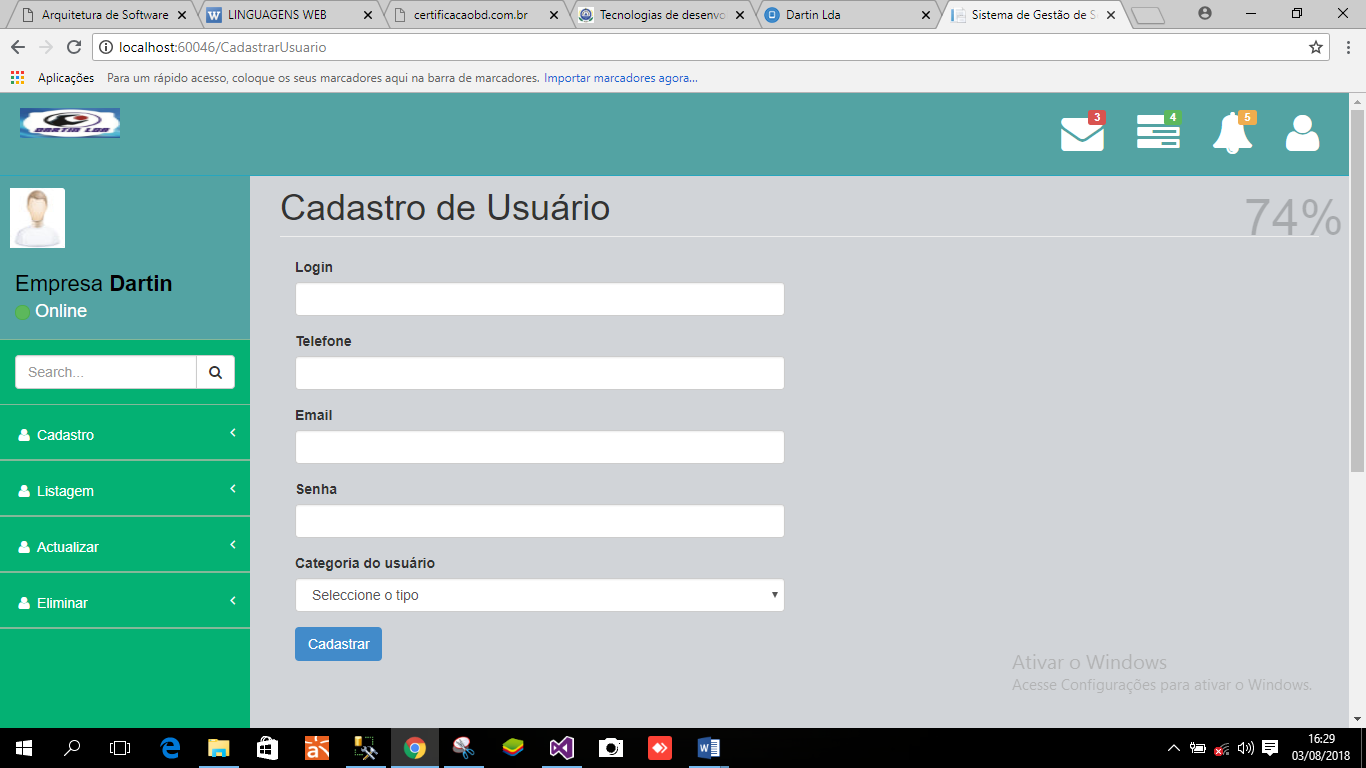


Figura 4. 3 Cadastro se Usuários

### **Página de Login**

A figura 4.4 mostra o login para acessar o sistema, que apenas utilizadores cadastrados poderão executar essa funcionalidade.

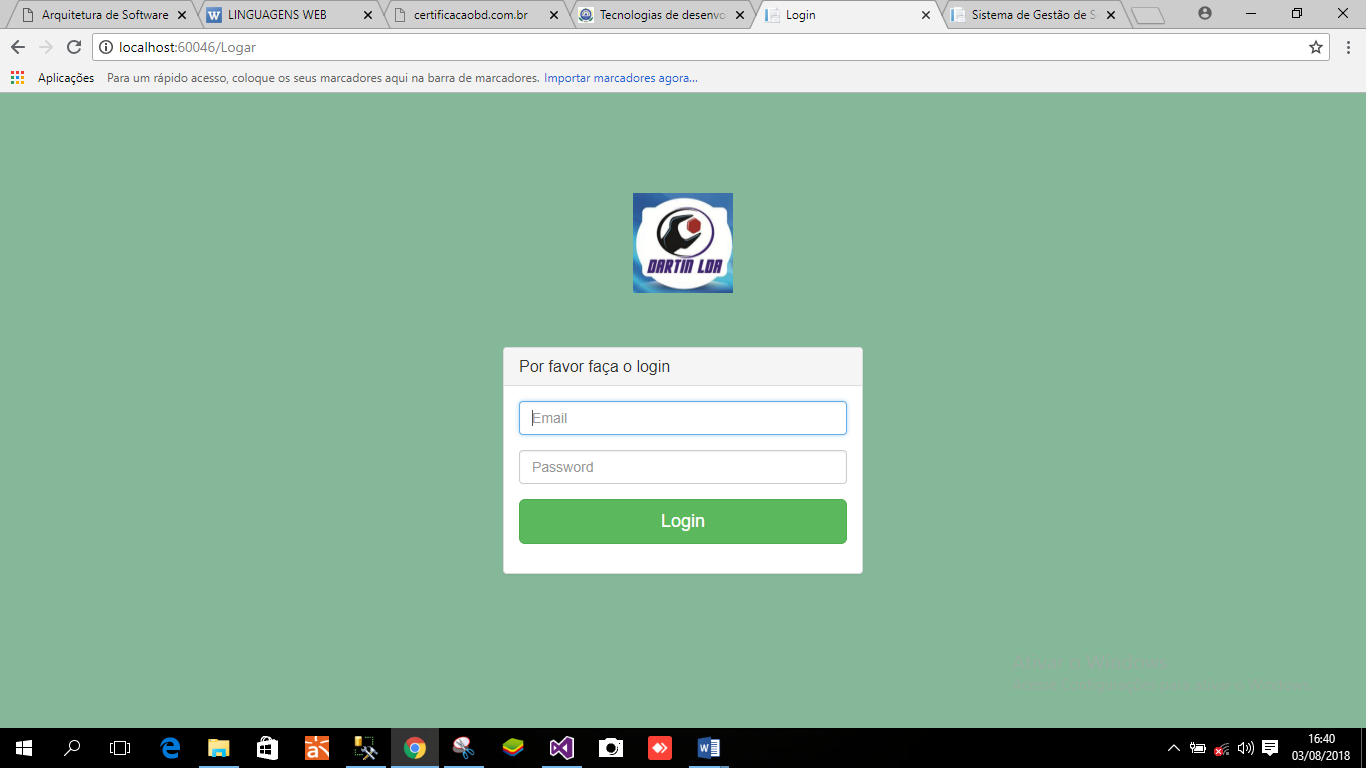


Figura 4. 4 Login

### **Página de Cadastro de Clientes**

A figura 4.5 mostra o cadastro de clientes, que é quase idêntico ao cadastro de usuários que já foi acima mostrado.

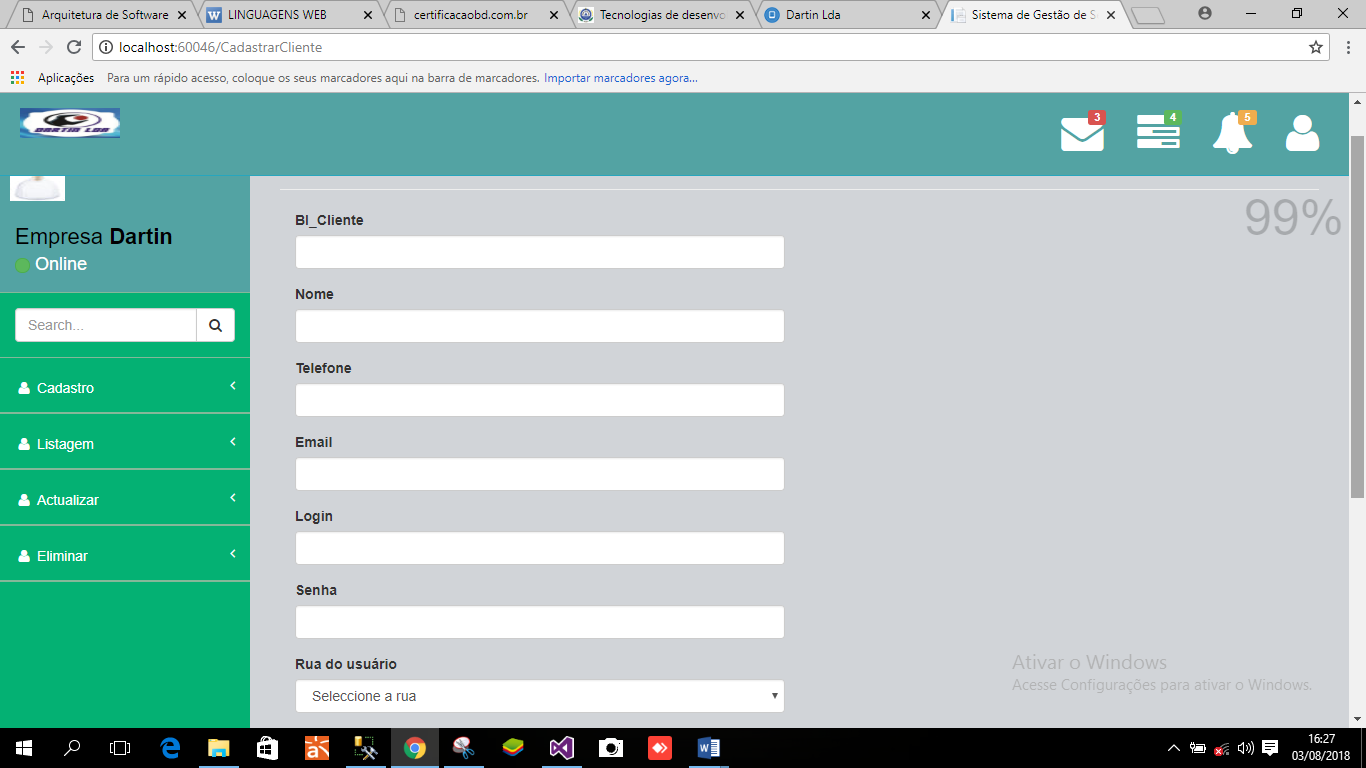


Figura 4. 5 Cadastro de Clientes

# CONCLUSÃO

O desenvolvimento do presente trabalho permitiu uma pesquisa de campo para obter dados mais consistentes sobre as etapas do processo.

Este projecto é resposta para alguns problemas que foram verificados dentro da Oficina DARTIN, quanto a organização total dela. Pois este vai permitir uma maior interacção entre os funcionários e clientes aumentando eficiência no atendimento e consequentemente poderá aumentar a satisfação dos clientes e aumentar o volume de atendimento.

Podemos em última instância dizer que a maior parte dos objectivos preconizados para este projecto forma alcançados, apesar de algumas adversidades registadas ao longo do mesmo.

## **Recomendações**

Recomendamos que em próximas versões sejam adicionadas mais funcionalidades ao sistema como: apresentar um calendário de disponibilidade, incluir um simulador de cálculos de orçamento para as reparações.

Por outro lado, seria benéfico e atendendo a evolução tecnológica que a parte dos clientes fosse desenvolvido em uma aplicação mobile.

# REFERÊNCIAS BIBLIOGRÁFICAS

* Abreu, Luís – ASP.NET 4.5 Curso Completo, “Editora FCA”, 2013;
* Battisti, Julio – ASP.NET, 2003;
* Damas, L. D. (2005). SQL Edição 13. FCA.
* LARMAN, Craig “UML y Patrones”, Prentice Hall Iberoamericana, 1999.
* PRESSMAN, R. S. (2003). Engenharia De Software Edição 2003. São Paulo (BR).
* MELO, Ana Cristina. **Desenvolvendo Aplicações com UML 2.0 – do conceitual àimplementação** – 2a. Rio de Janeiro: Brasport, 2004.
* Nunes, M., O´Neill, H. – Fundamental de UML, “Editora FCA”, 2003;
* http//www.automacaoerobotica.blogspot.com. acessado aos 10.06.2017..
* http//www.dsc.ufcg.edu.br. acessado aos 12.06.2017.
* http//www.oficinaintegrada.com. acessado aos 12.06.2017.
* http//www.omecanico.com.br/oficina-organizada. acessado aos 03.06.2017.
* <http://blog.takao.com.br/softwares-de-gestao-que-podem-ajudar-sua-oficina/> : acessado aos 21/11/2017;
* <http://professores.dcc.ufla.br/~bruno/aulas/livro-normal.pdf>: Acessado aos 28/12/2017.
* <http://walderson.com/site/wp-content/uploads/2015/07/DocumentodeRegrasdeNegocio-Exemplo.pdf>: acessado aos 28/12/2017;
* <https://fit.faccat.br/~guto/artigos/Artigo_Paradigmas_de_Programacao.pdf>: acessado aos 28/12/2017;
* <https://www.empresarialsoft.com.br/item/-programa-para-oficina-mec%E2nica-premium.html>: acessado aos 21/11/2017;
* msdn.microsoft.com/pt-br/libary/dd409377.aspx. acessado aos 15.06.2017.
* Programa de Gestão de Oficina Mecânica Integrada <<https://www.oficinaintegrada.com.br/software-gerencimento-oficina-mecanica/programa-gestao-oficina-mecanica-integrada/default.asp>:> acessado aos 09/Dez/2017;
* Programa de Gestão de Oficina Mecanica, disponível em <<https://www.oficinaintegrada.com.br/software-gerencimento-oficina-mecanica/programa-gestao-oficina-mecanica-integrada/blog.ASP?NOTICIA=Agendamento%20de%20servi%C3%A7os&ID_NOTICIA=58>> acessado aos 20/11/2017;
* Sistema de Gestão, disponível em ><https://www.guiaempreendedor.com/clico-responde-o-que-e-sistema-de-gestao/>> acessado aos 06/12/2017;
* Sistemas de Gestão de Serviços, Disponível em <https://www.erpflex.com.br/sistema-de-gestao-servicos> acessado aos 26/12/2017;

**ANEXOS**

CREATE DATABASE [db\_EmpresaDartin1]

CONTAINMENT = NONE

ON PRIMARY

( NAME = N'db\_EmpresaDartin1', FILENAME = N'C:\Program Files\Microsoft SQL Server\MSSQL12.MSSQLSERVER\MSSQL\DATA\db\_EmpresaDartin1.mdf' , SIZE = 4288KB , MAXSIZE = UNLIMITED, FILEGROWTH = 1024KB )

LOG ON

( NAME = N'db\_EmpresaDartin1\_log', FILENAME = N'C:\Program Files\Microsoft SQL Server\MSSQL12.MSSQLSERVER\MSSQL\DATA\db\_EmpresaDartin1\_log.ldf' , SIZE = 1856KB , MAXSIZE = 2048GB , FILEGROWTH = 10%)

GO

ALTER DATABASE [db\_EmpresaDartin1] SET COMPATIBILITY\_LEVEL = 120

GO

IF (1 = FULLTEXTSERVICEPROPERTY('IsFullTextInstalled'))

begin

EXEC [db\_EmpresaDartin1].[dbo].[sp\_fulltext\_database] @action = 'enable'

end

GO

ALTER DATABASE [db\_EmpresaDartin1] SET ANSI\_NULL\_DEFAULT OFF

GO

ALTER DATABASE [db\_EmpresaDartin1] SET ANSI\_NULLS OFF

GO

ALTER DATABASE [db\_EmpresaDartin1] SET ANSI\_PADDING OFF

GO

ALTER DATABASE [db\_EmpresaDartin1] SET ANSI\_WARNINGS OFF

GO

ALTER DATABASE [db\_EmpresaDartin1] SET ARITHABORT OFF

GO

ALTER DATABASE [db\_EmpresaDartin1] SET AUTO\_CLOSE OFF

GO

ALTER DATABASE [db\_EmpresaDartin1] SET AUTO\_SHRINK OFF

GO

ALTER DATABASE [db\_EmpresaDartin1] SET AUTO\_UPDATE\_STATISTICS ON

GO

ALTER DATABASE [db\_EmpresaDartin1] SET CURSOR\_CLOSE\_ON\_COMMIT OFF

GO

ALTER DATABASE [db\_EmpresaDartin1] SET CURSOR\_DEFAULT GLOBAL

GO

ALTER DATABASE [db\_EmpresaDartin1] SET CONCAT\_NULL\_YIELDS\_NULL OFF

GO

ALTER DATABASE [db\_EmpresaDartin1] SET NUMERIC\_ROUNDABORT OFF

GO

ALTER DATABASE [db\_EmpresaDartin1] SET QUOTED\_IDENTIFIER OFF

GO

ALTER DATABASE [db\_EmpresaDartin1] SET RECURSIVE\_TRIGGERS OFF

GO

ALTER DATABASE [db\_EmpresaDartin1] SET ENABLE\_BROKER

GO

ALTER DATABASE [db\_EmpresaDartin1] SET AUTO\_UPDATE\_STATISTICS\_ASYNC OFF

GO

ALTER DATABASE [db\_EmpresaDartin1] SET DATE\_CORRELATION\_OPTIMIZATION OFF

GO

ALTER DATABASE [db\_EmpresaDartin1] SET TRUSTWORTHY OFF

GO

ALTER DATABASE [db\_EmpresaDartin1] SET ALLOW\_SNAPSHOT\_ISOLATION OFF

GO

ALTER DATABASE [db\_EmpresaDartin1] SET PARAMETERIZATION SIMPLE

GO

ALTER DATABASE [db\_EmpresaDartin1] SET READ\_COMMITTED\_SNAPSHOT OFF

GO

ALTER DATABASE [db\_EmpresaDartin1] SET HONOR\_BROKER\_PRIORITY OFF

GO

ALTER DATABASE [db\_EmpresaDartin1] SET RECOVERY FULL

GO

ALTER DATABASE [db\_EmpresaDartin1] SET MULTI\_USER

GO

ALTER DATABASE [db\_EmpresaDartin1] SET PAGE\_VERIFY CHECKSUM

GO

ALTER DATABASE [db\_EmpresaDartin1] SET DB\_CHAINING OFF

GO

ALTER DATABASE [db\_EmpresaDartin1] SET FILESTREAM( NON\_TRANSACTED\_ACCESS = OFF )

GO

ALTER DATABASE [db\_EmpresaDartin1] SET TARGET\_RECOVERY\_TIME = 0 SECONDS

GO

ALTER DATABASE [db\_EmpresaDartin1] SET DELAYED\_DURABILITY = DISABLED

GO

EXEC sys.sp\_db\_vardecimal\_storage\_format N'db\_EmpresaDartin1', N'ON'

GO

USE [db\_EmpresaDartin1]

GO

/\*\*\*\*\*\* Object: Table [dbo].[tb\_Bairro] Script Date: 02/10/2018 07:10:15 \*\*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

SET ANSI\_PADDING ON

GO

CREATE TABLE [dbo].[tb\_Bairro](

[Id\_Bairro] [int] IDENTITY(1,1) NOT NULL,

[Descricao] [varchar](50) NOT NULL,

[Id\_Municipio] [int] NULL,

PRIMARY KEY CLUSTERED

(

[Id\_Bairro] ASC

)WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, IGNORE\_DUP\_KEY = OFF, ALLOW\_ROW\_LOCKS = ON, ALLOW\_PAGE\_LOCKS = ON) ON [PRIMARY]

) ON [PRIMARY]

GO

SET ANSI\_PADDING OFF

GO

/\*\*\*\*\*\* Object: Table [dbo].[tb\_Cliente] Script Date: 02/10/2018 07:10:15 \*\*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

SET ANSI\_PADDING ON

GO

CREATE TABLE [dbo].[tb\_Cliente](

[Id\_Cliente] [int] IDENTITY(1,1) NOT NULL,

[BI\_Cliente] [varchar](50) NOT NULL,

[Nome] [varchar](80) NOT NULL,

[Telefone] [varchar](15) NOT NULL,

[Email] [varchar](80) NOT NULL,

[Login] [varchar](80) NOT NULL,

[Senha] [varchar](80) NOT NULL,

[Id\_Rua] [int] NULL,

PRIMARY KEY CLUSTERED

(

[Id\_Cliente] ASC

)WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, IGNORE\_DUP\_KEY = OFF, ALLOW\_ROW\_LOCKS = ON, ALLOW\_PAGE\_LOCKS = ON) ON [PRIMARY]

) ON [PRIMARY]

GO

SET ANSI\_PADDING OFF

GO

/\*\*\*\*\*\* Object: Table [dbo].[tb\_MarcaVeiculo] Script Date: 02/10/2018 07:10:15 \*\*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

SET ANSI\_PADDING ON

GO

CREATE TABLE [dbo].[tb\_MarcaVeiculo](

[Id\_Marca] [int] IDENTITY(1,1) NOT NULL,

[Descricao] [varchar](50) NOT NULL,

PRIMARY KEY CLUSTERED

(

[Id\_Marca] ASC

)WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, IGNORE\_DUP\_KEY = OFF, ALLOW\_ROW\_LOCKS = ON, ALLOW\_PAGE\_LOCKS = ON) ON [PRIMARY]

) ON [PRIMARY]

GO

SET ANSI\_PADDING OFF

GO

/\*\*\*\*\*\* Object: Table [dbo].[tb\_ModeloVeiculo] Script Date: 02/10/2018 07:10:15 \*\*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

SET ANSI\_PADDING ON

GO

CREATE TABLE [dbo].[tb\_ModeloVeiculo](

[Id\_Modelo] [int] IDENTITY(1,1) NOT NULL,

[Id\_Marca] [int] NULL,

[Descricao] [varchar](50) NOT NULL,

PRIMARY KEY CLUSTERED

(

[Id\_Modelo] ASC

)WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, IGNORE\_DUP\_KEY = OFF, ALLOW\_ROW\_LOCKS = ON, ALLOW\_PAGE\_LOCKS = ON) ON [PRIMARY]

) ON [PRIMARY]

GO

SET ANSI\_PADDING OFF

GO

/\*\*\*\*\*\* Object: Table [dbo].[tb\_Municipio] Script Date: 02/10/2018 07:10:15 \*\*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

SET ANSI\_PADDING ON

GO

CREATE TABLE [dbo].[tb\_Municipio](

[Id\_Municipio] [int] IDENTITY(1,1) NOT NULL,

[Descricao] [varchar](50) NOT NULL,

[Id\_Provincia] [int] NULL,

PRIMARY KEY CLUSTERED

(

[Id\_Municipio] ASC

)WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, IGNORE\_DUP\_KEY = OFF, ALLOW\_ROW\_LOCKS = ON, ALLOW\_PAGE\_LOCKS = ON) ON [PRIMARY]

) ON [PRIMARY]

GO

SET ANSI\_PADDING OFF

GO

/\*\*\*\*\*\* Object: Table [dbo].[tb\_Pagamento] Script Date: 02/10/2018 07:10:15 \*\*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

CREATE TABLE [dbo].[tb\_Pagamento](

[Id\_Pagamento] [int] IDENTITY(1,1) NOT NULL,

[Id\_Solicitacao] [int] NULL,

[DataPagamento] [datetime] NULL,

[ValorPago] [decimal](18, 0) NULL,

[Id\_Usuario] [int] NULL,

[Id\_TipoPagamento] [tinyint] NULL,

PRIMARY KEY CLUSTERED

(

[Id\_Pagamento] ASC

)WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, IGNORE\_DUP\_KEY = OFF, ALLOW\_ROW\_LOCKS = ON, ALLOW\_PAGE\_LOCKS = ON) ON [PRIMARY]

) ON [PRIMARY]

GO

/\*\*\*\*\*\* Object: Table [dbo].[tb\_Preco] Script Date: 02/10/2018 07:10:15 \*\*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

CREATE TABLE [dbo].[tb\_Preco](

[Id\_Preco] [int] IDENTITY(1,1) NOT NULL,

[Valor] [decimal](18, 0) NOT NULL,

[Id\_TipoServico] [tinyint] NULL,

[Estado] [tinyint] NULL,

[Id\_Usuario] [int] NULL,

PRIMARY KEY CLUSTERED

(

[Id\_Preco] ASC

)WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, IGNORE\_DUP\_KEY = OFF, ALLOW\_ROW\_LOCKS = ON, ALLOW\_PAGE\_LOCKS = ON) ON [PRIMARY]

) ON [PRIMARY]

GO

/\*\*\*\*\*\* Object: Table [dbo].[tb\_Provincia] Script Date: 02/10/2018 07:10:15 \*\*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

SET ANSI\_PADDING ON

GO

CREATE TABLE [dbo].[tb\_Provincia](

[Id\_Provincia] [int] IDENTITY(1,1) NOT NULL,

[Descricao] [varchar](50) NOT NULL,

PRIMARY KEY CLUSTERED

(

[Id\_Provincia] ASC

)WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, IGNORE\_DUP\_KEY = OFF, ALLOW\_ROW\_LOCKS = ON, ALLOW\_PAGE\_LOCKS = ON) ON [PRIMARY]

) ON [PRIMARY]

GO

SET ANSI\_PADDING OFF

GO

/\*\*\*\*\*\* Object: Table [dbo].[tb\_Rua] Script Date: 02/10/2018 07:10:15 \*\*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

SET ANSI\_PADDING ON

GO

CREATE TABLE [dbo].[tb\_Rua](

[Id\_Rua] [int] IDENTITY(1,1) NOT NULL,

[Descricao] [varchar](50) NOT NULL,

[Id\_Bairro] [int] NULL,

PRIMARY KEY CLUSTERED

(

[Id\_Rua] ASC

)WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, IGNORE\_DUP\_KEY = OFF, ALLOW\_ROW\_LOCKS = ON, ALLOW\_PAGE\_LOCKS = ON) ON [PRIMARY]

) ON [PRIMARY]

GO

SET ANSI\_PADDING OFF

GO

/\*\*\*\*\*\* Object: Table [dbo].[tb\_ServicoSolicitacao] Script Date: 02/10/2018 07:10:15 \*\*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

CREATE TABLE [dbo].[tb\_ServicoSolicitacao](

[Id\_ServicoSolicitacao] [int] IDENTITY(1,1) NOT NULL,

[Id\_Solicitacao] [int] NULL,

[Id\_TipoServico] [tinyint] NULL,

[DataResgitro] [datetime] NOT NULL,

[Estado] [tinyint] NULL,

[Id\_Usuario] [int] NULL,

[DataRegistro] [datetime] NULL,

PRIMARY KEY CLUSTERED

(

[Id\_ServicoSolicitacao] ASC

)WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, IGNORE\_DUP\_KEY = OFF, ALLOW\_ROW\_LOCKS = ON, ALLOW\_PAGE\_LOCKS = ON) ON [PRIMARY]

) ON [PRIMARY]

GO

/\*\*\*\*\*\* Object: Table [dbo].[tb\_Solicitacao] Script Date: 02/10/2018 07:10:15 \*\*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

CREATE TABLE [dbo].[tb\_Solicitacao](

[Id\_Solicitacao] [int] IDENTITY(1,1) NOT NULL,

[Id\_Cliente] [int] NULL,

[Id\_Veiculo] [int] NULL,

[DataSolicitacao] [datetime] NULL DEFAULT (getdate()),

[Estado] [tinyint] NULL,

PRIMARY KEY CLUSTERED

(

[Id\_Solicitacao] ASC

)WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, IGNORE\_DUP\_KEY = OFF, ALLOW\_ROW\_LOCKS = ON, ALLOW\_PAGE\_LOCKS = ON) ON [PRIMARY]

) ON [PRIMARY]

GO

/\*\*\*\*\*\* Object: Table [dbo].[tb\_TipoPagamento] Script Date: 02/10/2018 07:10:15 \*\*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

SET ANSI\_PADDING ON

GO

CREATE TABLE [dbo].[tb\_TipoPagamento](

[Id\_TipoPagamento] [tinyint] IDENTITY(1,1) NOT NULL,

[Descricao] [varchar](50) NOT NULL,

PRIMARY KEY CLUSTERED

(

[Id\_TipoPagamento] ASC

)WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, IGNORE\_DUP\_KEY = OFF, ALLOW\_ROW\_LOCKS = ON, ALLOW\_PAGE\_LOCKS = ON) ON [PRIMARY]

) ON [PRIMARY]

GO

SET ANSI\_PADDING OFF

GO

/\*\*\*\*\*\* Object: Table [dbo].[tb\_TipoServico] Script Date: 02/10/2018 07:10:15 \*\*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

SET ANSI\_PADDING ON

GO

CREATE TABLE [dbo].[tb\_TipoServico](

[Id\_TipoServico] [tinyint] IDENTITY(1,1) NOT NULL,

[Descricao] [varchar](50) NOT NULL,

[Id\_Usuario] [int] NULL,

PRIMARY KEY CLUSTERED

(

[Id\_TipoServico] ASC

)WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, IGNORE\_DUP\_KEY = OFF, ALLOW\_ROW\_LOCKS = ON, ALLOW\_PAGE\_LOCKS = ON) ON [PRIMARY]

) ON [PRIMARY]

GO

SET ANSI\_PADDING OFF

GO

/\*\*\*\*\*\* Object: Table [dbo].[tb\_TipoUsuario] Script Date: 02/10/2018 07:10:15 \*\*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

SET ANSI\_PADDING ON

GO

CREATE TABLE [dbo].[tb\_TipoUsuario](

[Id\_TipoUsuario] [int] IDENTITY(1,1) NOT NULL,

[Descricao] [varchar](50) NOT NULL,

PRIMARY KEY CLUSTERED

(

[Id\_TipoUsuario] ASC

)WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, IGNORE\_DUP\_KEY = OFF, ALLOW\_ROW\_LOCKS = ON, ALLOW\_PAGE\_LOCKS = ON) ON [PRIMARY]

) ON [PRIMARY]

GO

SET ANSI\_PADDING OFF

GO

/\*\*\*\*\*\* Object: Table [dbo].[tb\_Usuario] Script Date: 02/10/2018 07:10:15 \*\*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

SET ANSI\_PADDING ON

GO

CREATE TABLE [dbo].[tb\_Usuario](

[Id\_Usuario] [int] IDENTITY(1,1) NOT NULL,

[Login] [varchar](50) NOT NULL,

[Telefone] [varchar](15) NOT NULL,

[Email] [varchar](80) NOT NULL,

[Senha] [varchar](80) NOT NULL,

[Status] [tinyint] NULL,

[Id\_TipoUsuario] [int] NULL,

PRIMARY KEY CLUSTERED

(

[Id\_Usuario] ASC

)WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, IGNORE\_DUP\_KEY = OFF, ALLOW\_ROW\_LOCKS = ON, ALLOW\_PAGE\_LOCKS = ON) ON [PRIMARY]

) ON [PRIMARY]

GO

SET ANSI\_PADDING OFF

GO

/\*\*\*\*\*\* Object: Table [dbo].[tb\_Veiculo] Script Date: 02/10/2018 07:10:15 \*\*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

SET ANSI\_PADDING ON

GO

CREATE TABLE [dbo].[tb\_Veiculo](

[Id\_Veiculo] [int] IDENTITY(1,1) NOT NULL,

[Matricula] [varchar](8) NOT NULL,

[NumShassis] [varchar](20) NOT NULL,

[Id\_Modelo] [int] NULL,

PRIMARY KEY CLUSTERED

(

[Id\_Veiculo] ASC

)WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, IGNORE\_DUP\_KEY = OFF, ALLOW\_ROW\_LOCKS = ON, ALLOW\_PAGE\_LOCKS = ON) ON [PRIMARY]

) ON [PRIMARY]

GO

SET ANSI\_PADDING OFF

GO

ALTER TABLE [dbo].[tb\_Bairro] WITH CHECK ADD CONSTRAINT [fk\_tb\_Bairro\_tb\_Municipio] FOREIGN KEY([Id\_Municipio])

REFERENCES [dbo].[tb\_Municipio] ([Id\_Municipio])

GO

ALTER TABLE [dbo].[tb\_Bairro] CHECK CONSTRAINT [fk\_tb\_Bairro\_tb\_Municipio]

GO

ALTER TABLE [dbo].[tb\_Cliente] WITH CHECK ADD CONSTRAINT [fk\_tb\_Cliente\_tb\_Rua] FOREIGN KEY([Id\_Rua])

REFERENCES [dbo].[tb\_Rua] ([Id\_Rua])

GO

ALTER TABLE [dbo].[tb\_Cliente] CHECK CONSTRAINT [fk\_tb\_Cliente\_tb\_Rua]

GO

ALTER TABLE [dbo].[tb\_ModeloVeiculo] WITH CHECK ADD CONSTRAINT [fk\_tb\_ModeloVeiculo\_tb\_MarcaVeiculo] FOREIGN KEY([Id\_Marca])

REFERENCES [dbo].[tb\_MarcaVeiculo] ([Id\_Marca])

GO

ALTER TABLE [dbo].[tb\_ModeloVeiculo] CHECK CONSTRAINT [fk\_tb\_ModeloVeiculo\_tb\_MarcaVeiculo]

GO

ALTER TABLE [dbo].[tb\_Municipio] WITH CHECK ADD CONSTRAINT [fk\_tb\_Municipio\_tb\_Provincia] FOREIGN KEY([Id\_Provincia])

REFERENCES [dbo].[tb\_Provincia] ([Id\_Provincia])

GO

ALTER TABLE [dbo].[tb\_Municipio] CHECK CONSTRAINT [fk\_tb\_Municipio\_tb\_Provincia]

GO

ALTER TABLE [dbo].[tb\_Pagamento] WITH CHECK ADD CONSTRAINT [fk\_tb\_Pagamento\_tb\_Solicitacao] FOREIGN KEY([Id\_Solicitacao])

REFERENCES [dbo].[tb\_Solicitacao] ([Id\_Solicitacao])

GO

ALTER TABLE [dbo].[tb\_Pagamento] CHECK CONSTRAINT [fk\_tb\_Pagamento\_tb\_Solicitacao]

GO

ALTER TABLE [dbo].[tb\_Pagamento] WITH CHECK ADD CONSTRAINT [fk\_tb\_Pagamento\_tb\_TipoPagamento] FOREIGN KEY([Id\_TipoPagamento])

REFERENCES [dbo].[tb\_TipoPagamento] ([Id\_TipoPagamento])

GO

ALTER TABLE [dbo].[tb\_Pagamento] CHECK CONSTRAINT [fk\_tb\_Pagamento\_tb\_TipoPagamento]

GO

ALTER TABLE [dbo].[tb\_Pagamento] WITH CHECK ADD CONSTRAINT [fk\_tb\_Pagamento\_tb\_Usuario] FOREIGN KEY([Id\_Usuario])

REFERENCES [dbo].[tb\_Usuario] ([Id\_Usuario])

GO

ALTER TABLE [dbo].[tb\_Pagamento] CHECK CONSTRAINT [fk\_tb\_Pagamento\_tb\_Usuario]

GO

ALTER TABLE [dbo].[tb\_Preco] WITH CHECK ADD CONSTRAINT [fk\_tb\_Preco\_tb\_TipoServico] FOREIGN KEY([Id\_TipoServico])

REFERENCES [dbo].[tb\_TipoServico] ([Id\_TipoServico])

GO

ALTER TABLE [dbo].[tb\_Preco] CHECK CONSTRAINT [fk\_tb\_Preco\_tb\_TipoServico]

GO

ALTER TABLE [dbo].[tb\_Preco] WITH CHECK ADD CONSTRAINT [fk\_tb\_Preco\_tb\_Usuario] FOREIGN KEY([Id\_Usuario])

REFERENCES [dbo].[tb\_Usuario] ([Id\_Usuario])

GO

ALTER TABLE [dbo].[tb\_Preco] CHECK CONSTRAINT [fk\_tb\_Preco\_tb\_Usuario]

GO

ALTER TABLE [dbo].[tb\_Rua] WITH CHECK ADD CONSTRAINT [fk\_tb\_Rua\_tb\_Bairro] FOREIGN KEY([Id\_Bairro])

REFERENCES [dbo].[tb\_Bairro] ([Id\_Bairro])

GO

ALTER TABLE [dbo].[tb\_Rua] CHECK CONSTRAINT [fk\_tb\_Rua\_tb\_Bairro]

GO

ALTER TABLE [dbo].[tb\_ServicoSolicitacao] WITH CHECK ADD CONSTRAINT [fk\_tb\_ServicoSolicitacao\_tb\_Solicitacao] FOREIGN KEY([Id\_Solicitacao])

REFERENCES [dbo].[tb\_Solicitacao] ([Id\_Solicitacao])

GO

ALTER TABLE [dbo].[tb\_ServicoSolicitacao] CHECK CONSTRAINT [fk\_tb\_ServicoSolicitacao\_tb\_Solicitacao]

GO

ALTER TABLE [dbo].[tb\_ServicoSolicitacao] WITH CHECK ADD CONSTRAINT [fk\_tb\_ServicoSolicitacao\_tb\_TipoServico] FOREIGN KEY([Id\_TipoServico])

REFERENCES [dbo].[tb\_TipoServico] ([Id\_TipoServico])

GO

ALTER TABLE [dbo].[tb\_ServicoSolicitacao] CHECK CONSTRAINT [fk\_tb\_ServicoSolicitacao\_tb\_TipoServico]

GO

ALTER TABLE [dbo].[tb\_ServicoSolicitacao] WITH CHECK ADD CONSTRAINT [fk\_tb\_ServicoSolicitacao\_tb\_Usuario] FOREIGN KEY([Id\_Usuario])

REFERENCES [dbo].[tb\_Usuario] ([Id\_Usuario])

GO

ALTER TABLE [dbo].[tb\_ServicoSolicitacao] CHECK CONSTRAINT [fk\_tb\_ServicoSolicitacao\_tb\_Usuario]

GO

ALTER TABLE [dbo].[tb\_Solicitacao] WITH CHECK ADD CONSTRAINT [fk\_tb\_Solicitacao\_tb\_Cliente] FOREIGN KEY([Id\_Cliente])

REFERENCES [dbo].[tb\_Cliente] ([Id\_Cliente])

GO

ALTER TABLE [dbo].[tb\_Solicitacao] CHECK CONSTRAINT [fk\_tb\_Solicitacao\_tb\_Cliente]

GO

ALTER TABLE [dbo].[tb\_Solicitacao] WITH CHECK ADD CONSTRAINT [fk\_tb\_Solicitacao\_tb\_Veiculo] FOREIGN KEY([Id\_Veiculo])

REFERENCES [dbo].[tb\_Veiculo] ([Id\_Veiculo])

GO

ALTER TABLE [dbo].[tb\_Solicitacao] CHECK CONSTRAINT [fk\_tb\_Solicitacao\_tb\_Veiculo]

GO

ALTER TABLE [dbo].[tb\_TipoServico] WITH CHECK ADD CONSTRAINT [fk\_tb\_TipoServico\_tb\_Usuario] FOREIGN KEY([Id\_Usuario])

REFERENCES [dbo].[tb\_Usuario] ([Id\_Usuario])

GO

ALTER TABLE [dbo].[tb\_TipoServico] CHECK CONSTRAINT [fk\_tb\_TipoServico\_tb\_Usuario]

GO

ALTER TABLE [dbo].[tb\_Usuario] WITH CHECK ADD CONSTRAINT [fk\_tb\_Usuario\_tb\_TipoUsario] FOREIGN KEY([Id\_TipoUsuario])

REFERENCES [dbo].[tb\_TipoUsuario] ([Id\_TipoUsuario])

GO

ALTER TABLE [dbo].[tb\_Usuario] CHECK CONSTRAINT [fk\_tb\_Usuario\_tb\_TipoUsario]

GO

ALTER TABLE [dbo].[tb\_Veiculo] WITH CHECK ADD CONSTRAINT [fk\_tb\_Veiculo\_tb\_Modelo] FOREIGN KEY([Id\_Modelo])

REFERENCES [dbo].[tb\_ModeloVeiculo] ([Id\_Modelo])

GO

ALTER TABLE [dbo].[tb\_Veiculo] CHECK CONSTRAINT [fk\_tb\_Veiculo\_tb\_Modelo]

GO

/\*\*\*\*\*\* Object: StoredProcedure [dbo].[Sp\_MarcaVeiculo] Script Date: 02/10/2018 07:10:15 \*\*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

--Gerado por RMSA(Romário Meireles Sebastião Ângelo)

Create proc [dbo].[Sp\_MarcaVeiculo]

(

@Id\_Marca int=NULL,

@Descricao varchar(500)=NULL,

@FCT Varchar(1000),

@Resultado Varchar(1000)=null output--Opcional

)

As

BEGIN

DECLARE @FC AS VARCHAR(500)

SET @FC=LTRIM(RTRIM(@FCT))

BEGIN TRY

IF @FC NOT IN('I','U','D','SA','SO')

BEGIN

RAISERROR('Escolha uma das operações!',50,1)with log

END

ELSE

BEGIN

IF @FC='I'

BEGIN

INSERT tb\_MarcaVeiculo(

Descricao

)

VALUES(

@Descricao

)

if @@error=0 set @Resultado='Registo inserido com sucesso'

END

IF @FC='U'

BEGIN

UPDATE tb\_MarcaVeiculo SET

Descricao=@Descricao

if @@error=0 set @Resultado='Registo alterado com sucesso'

END

IF @FC='D'

BEGIN

DELETE FROM tb\_MarcaVeiculo

WHERE Id\_Marca=@Id\_Marca

if @@error=0 set @Resultado='Registo eliminado com sucesso'

END

IF @FC='SA'

BEGIN

SELECT \* FROM tb\_MarcaVeiculo WITH (NOLOCK)

if @@rowcount=0 set @Resultado='Não existe registo para o criterio seleccionado'

END

IF @FC='SO'

BEGIN

SELECT \* FROM tb\_MarcaVeiculo WITH (NOLOCK)

WHERE Id\_Marca=@Id\_Marca

if @@rowcount=0 set @Resultado='Não existe registo para o criterio seleccionado'

END

END

END TRY

BEGIN CATCH

DECLARE @ErrorMessage NVARCHAR(4000)

DECLARE @ErrorSeverity INT

DECLARE @ErrorState INT

SELECT @ErrorMessage=ERROR\_MESSAGE(),

@ErrorSeverity=ERROR\_SEVERITY(),

@ErrorState=ERROR\_STATE()

RAISERROR(@ErrorMessage,@ErrorSeverity,@ErrorState)

END CATCH

RETURN

END

GO

/\*\*\*\*\*\* Object: StoredProcedure [dbo].[Sp\_ModeloVeiculo] Script Date: 02/10/2018 07:10:15 \*\*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

--Gerado por RMSA(Romário Meireles Sebastião Ângelo)

CREATE proc [dbo].[Sp\_ModeloVeiculo]

(

@Id\_Modelo int=NULL,

@Id\_Marca int=NULL,

@Descricao varchar(500)=NULL,

@FCT Varchar(1000),

@Resultado Varchar(1000)=null output--Opcional

)

As

BEGIN

DECLARE @FC AS VARCHAR(500)

SET @FC=LTRIM(RTRIM(@FCT))

BEGIN TRY

IF @FC NOT IN('I','U','D','SA','SO','SM')

BEGIN

RAISERROR('Escolha uma das operações!',50,1)with log

END

ELSE

BEGIN

IF @FC='I'

BEGIN

INSERT tb\_ModeloVeiculo(

Id\_Marca,

Descricao

)

VALUES(

@Id\_Marca,

@Descricao

)

if @@error=0 set @Resultado='Registo inserido com sucesso'

END

IF @FC='U'

BEGIN

UPDATE tb\_ModeloVeiculo SET

Id\_Marca=@Id\_Marca,

Descricao=@Descricao

if @@error=0 set @Resultado='Registo alterado com sucesso'

END

IF @FC='D'

BEGIN

DELETE FROM tb\_ModeloVeiculo

WHERE Id\_Modelo=@Id\_Modelo

if @@error=0 set @Resultado='Registo eliminado com sucesso'

END

IF @FC='SA'

BEGIN

SELECT \* FROM tb\_ModeloVeiculo WITH (NOLOCK)

if @@rowcount=0 set @Resultado='Não existe registo para o criterio seleccionado'

END

IF @FC='SO'

BEGIN

SELECT \* FROM tb\_ModeloVeiculo WITH (NOLOCK)

WHERE Id\_Modelo=@Id\_Modelo

if @@rowcount=0 set @Resultado='Não existe registo para o criterio seleccionado'

END

IF @FC='SM'

BEGIN

select\* from tb\_MarcaVeiculo

if @@rowcount=0 set @Resultado='Não existe registo para o criterio seleccionado'

END

END

END TRY

BEGIN CATCH

DECLARE @ErrorMessage NVARCHAR(4000)

DECLARE @ErrorSeverity INT

DECLARE @ErrorState INT

SELECT @ErrorMessage=ERROR\_MESSAGE(),

@ErrorSeverity=ERROR\_SEVERITY(),

@ErrorState=ERROR\_STATE()

RAISERROR(@ErrorMessage,@ErrorSeverity,@ErrorState)

END CATCH

RETURN

END

GO

/\*\*\*\*\*\* Object: StoredProcedure [dbo].[Sp\_Municipio] Script Date: 02/10/2018 07:10:15 \*\*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

--Gerado por RMSA(Romário Meireles Sebastião Ângelo)

Create proc [dbo].[Sp\_Municipio]

(

@Id\_Municipio int=NULL,

@Descricao varchar(500)=NULL,

@Id\_Provincia int=NULL,

@FCT Varchar(1000),

@Resultado Varchar(1000)=null output--Opcional

)

As

BEGIN

DECLARE @FC AS VARCHAR(500)

SET @FC=LTRIM(RTRIM(@FCT))

BEGIN TRY

IF @FC NOT IN('I','U','D','SA','SO')

BEGIN

RAISERROR('Escolha uma das operações!',50,1)with log

END

ELSE

BEGIN

IF @FC='I'

BEGIN

INSERT tb\_Municipio(

Descricao,

Id\_Provincia

)

VALUES(

@Descricao,

@Id\_Provincia

)

if @@error=0 set @Resultado='Registo inserido com sucesso'

END

IF @FC='U'

BEGIN

UPDATE tb\_Municipio SET

Descricao=@Descricao,

Id\_Provincia=@Id\_Provincia

if @@error=0 set @Resultado='Registo alterado com sucesso'

END

IF @FC='D'

BEGIN

DELETE FROM tb\_Municipio

WHERE Id\_Municipio=@Id\_Municipio

if @@error=0 set @Resultado='Registo eliminado com sucesso'

END

IF @FC='SA'

BEGIN

SELECT \* FROM tb\_Municipio WITH (NOLOCK)

if @@rowcount=0 set @Resultado='Não existe registo para o criterio seleccionado'

END

IF @FC='SO'

BEGIN

SELECT \* FROM tb\_Municipio WITH (NOLOCK)

WHERE Id\_Municipio=@Id\_Municipio

if @@rowcount=0 set @Resultado='Não existe registo para o criterio seleccionado'

END

END

END TRY

BEGIN CATCH

DECLARE @ErrorMessage NVARCHAR(4000)

DECLARE @ErrorSeverity INT

DECLARE @ErrorState INT

SELECT @ErrorMessage=ERROR\_MESSAGE(),

@ErrorSeverity=ERROR\_SEVERITY(),

@ErrorState=ERROR\_STATE()

RAISERROR(@ErrorMessage,@ErrorSeverity,@ErrorState)

END CATCH

RETURN

END

GO

/\*\*\*\*\*\* Object: StoredProcedure [dbo].[Sp\_Pagamento] Script Date: 02/10/2018 07:10:15 \*\*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

--Gerado por RMSA(Romário Meireles Sebastião Ângelo)

CREATE proc [dbo].[Sp\_Pagamento]

(

@Id\_Pagamento int=NULL,

@Id\_Solicitacao int=NULL,

@DataPagamento datetime=NULL,

@ValorPago decimal=NULL,

@Id\_Usuario int=NULL,

@Id\_TipoPagamento tinyint=NULL,

@FCT Varchar(1000),

@Resultado Varchar(1000)=null output--Opcional

)

As

BEGIN

DECLARE @FC AS VARCHAR(500)

SET @FC=LTRIM(RTRIM(@FCT))

BEGIN TRY

IF @FC NOT IN('I','U','D','SA','SO','SO','SS','SU','TP')

BEGIN

RAISERROR('Escolha uma das operações!',50,1)with log

END

ELSE

BEGIN

IF @FC='I'

BEGIN

INSERT tb\_Pagamento(

Id\_Solicitacao,

DataPagamento,

ValorPago,

Id\_Usuario,

Id\_TipoPagamento

)

VALUES(

@Id\_Solicitacao,

@DataPagamento,

@ValorPago,

@Id\_Usuario,

@Id\_TipoPagamento

)

if @@error=0 set @Resultado='Registo inserido com sucesso'

END

IF @FC='U'

BEGIN

UPDATE tb\_Pagamento SET

Id\_Solicitacao=@Id\_Solicitacao,

DataPagamento=@DataPagamento,

ValorPago=@ValorPago,

Id\_Usuario=@Id\_Usuario,

Id\_TipoPagamento=@Id\_TipoPagamento

if @@error=0 set @Resultado='Registo alterado com sucesso'

END

IF @FC='D'

BEGIN

DELETE FROM tb\_Pagamento

WHERE Id\_Pagamento=@Id\_Pagamento

if @@error=0 set @Resultado='Registo eliminado com sucesso'

END

IF @FC='SA'

BEGIN

SELECT \* FROM tb\_Pagamento WITH (NOLOCK)

if @@rowcount=0 set @Resultado='Não existe registo para o criterio seleccionado'

END

IF @FC='SO'

BEGIN

SELECT \* FROM tb\_Pagamento WITH (NOLOCK)

WHERE Id\_Pagamento=@Id\_Pagamento

if @@rowcount=0 set @Resultado='Não existe registo para o criterio seleccionado'

END

IF @FC='SS'

BEGIN

SELECT \* From tb\_Solicitacao WITH (NOLOCK)

if @@ROWCOUNT=0 set @Resultado='Não existe registo para o critério seleccionado'

END

IF @FC='SU'

BEGIN

SELECT \* From tb\_Usuario WITH (NOLOCK)

if @@ROWCOUNT=0 set @Resultado='Não existe registo para o critério seleccionado'

END

IF @FC='TP'

BEGIN

SELECT \* From tb\_TipoPagamento WITH (NOLOCK)

if @@ROWCOUNT=0 set @Resultado='Não existe registo para o critério seleccionado'

END

END

END TRY

BEGIN CATCH

DECLARE @ErrorMessage NVARCHAR(4000)

DECLARE @ErrorSeverity INT

DECLARE @ErrorState INT

SELECT @ErrorMessage=ERROR\_MESSAGE(),

@ErrorSeverity=ERROR\_SEVERITY(),

@ErrorState=ERROR\_STATE()

RAISERROR(@ErrorMessage,@ErrorSeverity,@ErrorState)

END CATCH

RETURN

END

GO

/\*\*\*\*\*\* Object: StoredProcedure [dbo].[Sp\_Preco] Script Date: 02/10/2018 07:10:15 \*\*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

--Gerado por RMSA(Romário Meireles Sebastião Ângelo)

CREATE proc [dbo].[Sp\_Preco]

(

@Id\_Preco int=NULL,

@Valor decimal=NULL,

@Id\_TipoServico tinyint=NULL,

@Estado tinyint=NULL,

@Id\_Usuario int=NULL,

@FCT Varchar(1000),

@Resultado Varchar(1000)=null output--Opcional

)

As

BEGIN

DECLARE @FC AS VARCHAR(500)

SET @FC=LTRIM(RTRIM(@FCT))

BEGIN TRY

IF @FC NOT IN('I','U','D','SA','SO','SU','ST')

BEGIN

RAISERROR('Escolha uma das operações!',50,1)with log

END

ELSE

BEGIN

IF @FC='I'

BEGIN

INSERT tb\_Preco(

Valor,

Id\_TipoServico,

Estado,

Id\_Usuario

)

VALUES(

@Valor,

@Id\_TipoServico,

@Estado,

@Id\_Usuario

)

if @@error=0 set @Resultado='Registo inserido com sucesso'

END

IF @FC='U'

BEGIN

UPDATE tb\_Preco SET

Valor=@Valor,

Id\_TipoServico=@Id\_TipoServico,

Estado=@Estado,

Id\_Usuario=@Id\_Usuario

if @@error=0 set @Resultado='Registo alterado com sucesso'

END

IF @FC='D'

BEGIN

DELETE FROM tb\_Preco

WHERE Id\_Preco=@Id\_Preco

if @@error=0 set @Resultado='Registo eliminado com sucesso'

END

IF @FC='SA'

BEGIN

SELECT \* FROM tb\_Preco WITH (NOLOCK)

if @@rowcount=0 set @Resultado='Não existe registo para o criterio seleccionado'

END

IF @FC='SO'

BEGIN

SELECT \* FROM tb\_Preco WITH (NOLOCK)

WHERE Id\_Preco=@Id\_Preco

if @@rowcount=0 set @Resultado='Não existe registo para o criterio seleccionado'

END

END

IF @FC='SU'

BEGIN

SELECT \* From tb\_Usuario WITH (NOLOCK)

if @@ROWCOUNT=0 set @Resultado='Não existe registo para o critério seleccionado'

IF @FC='ST'

BEGIN

SELECT \* From tb\_TipoServico WITH (NOLOCK)

if @@ROWCOUNT=0 set @Resultado='Não existe registo para o critério seleccionado'

END

END

END TRY

BEGIN CATCH

DECLARE @ErrorMessage NVARCHAR(4000)

DECLARE @ErrorSeverity INT

DECLARE @ErrorState INT

SELECT @ErrorMessage=ERROR\_MESSAGE(),

@ErrorSeverity=ERROR\_SEVERITY(),

@ErrorState=ERROR\_STATE()

RAISERROR(@ErrorMessage,@ErrorSeverity,@ErrorState)

END CATCH

RETURN

END

GO

/\*\*\*\*\*\* Object: StoredProcedure [dbo].[Sp\_ServicoSolicitacao] Script Date: 02/10/2018 07:10:15 \*\*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

--Gerado por RMSA(Romário Meireles Sebastião Ângelo)

CREATE proc [dbo].[Sp\_ServicoSolicitacao]

(

@Id\_ServicoSolicitacao int=NULL,

@Id\_Solicitacao int=NULL,

@Id\_TipoServico tinyint=NULL,

@DataResgitro datetime=NULL,

@Estado tinyint=NULL,

@Id\_Usuario int=NULL,

@DataRegistro datetime=NULL,

@FCT Varchar(1000),

@Resultado Varchar(1000)=null output--Opcional

)

As

BEGIN

DECLARE @FC AS VARCHAR(500)

SET @FC=LTRIM(RTRIM(@FCT))

BEGIN TRY

IF @FC NOT IN('I','U','D','SA','SO','SS','ST','SU','LSU')

BEGIN

RAISERROR('Escolha uma das operações!',50,1)with log

END

ELSE

BEGIN

IF @FC='I'

BEGIN

INSERT tb\_ServicoSolicitacao(

Id\_Solicitacao,

Id\_TipoServico,

DataResgitro,

Estado,

Id\_Usuario,

DataRegistro

)

VALUES(

@Id\_Solicitacao,

@Id\_TipoServico,

@DataResgitro,

@Estado,

@Id\_Usuario,

@DataRegistro

)

if @@error=0 set @Resultado='Registo inserido com sucesso'

END

IF @FC='U'

BEGIN

UPDATE tb\_ServicoSolicitacao SET

Id\_Solicitacao=@Id\_Solicitacao,

Id\_TipoServico=@Id\_TipoServico,

DataResgitro=@DataResgitro,

Estado=@Estado,

Id\_Usuario=@Id\_Usuario,

DataRegistro=@DataRegistro

if @@error=0 set @Resultado='Registo alterado com sucesso'

END

IF @FC='D'

BEGIN

DELETE FROM tb\_ServicoSolicitacao

WHERE Id\_ServicoSolicitacao=@Id\_ServicoSolicitacao

if @@error=0 set @Resultado='Registo eliminado com sucesso'

END

IF @FC='SA'

BEGIN

SELECT \* FROM tb\_ServicoSolicitacao WITH (NOLOCK)

if @@rowcount=0 set @Resultado='Não existe registo para o criterio seleccionado'

END

IF @FC='SO'

BEGIN

SELECT \* FROM tb\_ServicoSolicitacao WITH (NOLOCK)

WHERE Id\_ServicoSolicitacao=@Id\_ServicoSolicitacao

if @@rowcount=0 set @Resultado='Não existe registo para o criterio seleccionado'

END

IF @FC='SS'

BEGIN

select\* from tb\_Solicitacao

if @@rowcount=0 set @Resultado='Não existe registo para o criterio seleccionado'

END

IF @FC='ST'

BEGIN

select\* from tb\_TipoServico

if @@rowcount=0 set @Resultado='Não existe registo para o criterio seleccionado'

END

IF @FC='SU'

BEGIN

select\* from tb\_Usuario

if @@rowcount=0 set @Resultado='Não existe registo para o criterio seleccionado'

END

IF @FC='LSU'--lis...

BEGIN

SELECT \* FROM tb\_ServicoSolicitacao as t1

inner join tb\_Usuario as t2

on t1.Id\_Usuario=t2.Id\_Usuario

where t1.Id\_Usuario=@Id\_Usuario

if @@rowcount=0 set @Resultado='Não existe registo para o criterio seleccionado'

END

END

END TRY

BEGIN CATCH

DECLARE @ErrorMessage NVARCHAR(4000)

DECLARE @ErrorSeverity INT

DECLARE @ErrorState INT

SELECT @ErrorMessage=ERROR\_MESSAGE(),

@ErrorSeverity=ERROR\_SEVERITY(),

@ErrorState=ERROR\_STATE()

RAISERROR(@ErrorMessage,@ErrorSeverity,@ErrorState)

END CATCH

RETURN

END

GO

/\*\*\*\*\*\* Object: StoredProcedure [dbo].[Sp\_Solicitacao] Script Date: 02/10/2018 07:10:15 \*\*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

--Gerado por RMSA(Romário Meireles Sebastião Ângelo)

CREATE proc [dbo].[Sp\_Solicitacao]

(

@Id\_Solicitacao int=NULL,

@Id\_Cliente int=NULL,

@Id\_Veiculo int=NULL,

@DataSolicitacao datetime=NULL,

@Estado tinyint=NULL,

@FCT Varchar(1000),

@Resultado Varchar(1000)=null output--Opcional

)

As

BEGIN

DECLARE @FC AS VARCHAR(500)

SET @FC=LTRIM(RTRIM(@FCT))

BEGIN TRY

IF @FC NOT IN('I','U','D','SA','SO','SC','SV','SS','LSC')

BEGIN

RAISERROR('Escolha uma das operações!',50,1)with log

END

ELSE

BEGIN

IF @FC='I'

BEGIN

INSERT tb\_Solicitacao(

Id\_Cliente,

Id\_Veiculo,

DataSolicitacao,

Estado

)

VALUES(

@Id\_Cliente,

@Id\_Veiculo,

@DataSolicitacao,

@Estado

)

if @@error=0 set @Resultado='Registo inserido com sucesso'

END

IF @FC='U'

BEGIN

UPDATE tb\_Solicitacao SET

Id\_Cliente=@Id\_Cliente,

Id\_Veiculo=@Id\_Veiculo,

DataSolicitacao=@DataSolicitacao,

Estado=@Estado

if @@error=0 set @Resultado='Registo alterado com sucesso'

END

IF @FC='D'

BEGIN

DELETE FROM tb\_Solicitacao

WHERE Id\_Solicitacao=@Id\_Solicitacao

if @@error=0 set @Resultado='Registo eliminado com sucesso'

END

IF @FC='SA'

BEGIN

SELECT \* FROM tb\_Solicitacao WITH (NOLOCK)

if @@rowcount=0 set @Resultado='Não existe registo para o criterio seleccionado'

END

IF @FC='SO'

BEGIN

SELECT \* FROM tb\_Solicitacao WITH (NOLOCK)

WHERE Id\_Solicitacao=@Id\_Solicitacao

if @@rowcount=0 set @Resultado='Não existe registo para o criterio seleccionado'

END

IF @FC='SC'

BEGIN

select\* from tb\_Cliente

if @@rowcount=0 set @Resultado='Não existe registo para o criterio seleccionado'

END

IF @FC='SS'

BEGIN

select t1.Id\_Solicitacao,t2.Nome,t3.Matricula,t4.Descricao'Marca',t5.Descricao'Modelo',t1.DataSolicitacao,t1.Estado from tb\_Solicitacao as t1 inner join tb\_Cliente as t2 on t1.Id\_Cliente=t2.Id\_Cliente

inner join tb\_Veiculo as t3 on t1.Id\_Veiculo=t3.Id\_Veiculo

left join tb\_ModeloVeiculo as t4 on t3.Id\_Modelo=t4.Id\_Modelo

left join tb\_MarcaVeiculo as t5 on t4.Id\_Marca=t5.Id\_Marca

left join tb\_ServicoSolicitacao as t6 on t1.Id\_Solicitacao=t6.Id\_Solicitacao

if @@rowcount=0 set @Resultado='Não existe registo para o criterio seleccionado'

END

IF @FC='SV'

BEGIN

select\* from tb\_Veiculo

if @@rowcount=0 set @Resultado='Não existe registo para o criterio seleccionado'

END

IF @FC='LSC'--lis...

BEGIN

SELECT \* FROM tb\_Solicitacao as t1

inner join tb\_Cliente as t2

on t1.Id\_Cliente=t2.Id\_Cliente

where t1.Id\_Cliente=@Id\_Cliente

if @@rowcount=0 set @Resultado='Não existe registo para o criterio seleccionado'

END

END

END TRY

BEGIN CATCH

DECLARE @ErrorMessage NVARCHAR(4000)

DECLARE @ErrorSeverity INT

DECLARE @ErrorState INT

SELECT @ErrorMessage=ERROR\_MESSAGE(),

@ErrorSeverity=ERROR\_SEVERITY(),

@ErrorState=ERROR\_STATE()

RAISERROR(@ErrorMessage,@ErrorSeverity,@ErrorState)

END CATCH

RETURN

END

GO

/\*\*\*\*\*\* Object: StoredProcedure [dbo].[Sp\_TipoPagamento] Script Date: 02/10/2018 07:10:15 \*\*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

--Gerado por RMSA(Romário Meireles Sebastião Ângelo)

Create proc [dbo].[Sp\_TipoPagamento]

(

@Id\_TipoPagamento tinyint=NULL,

@Descricao varchar(500)=NULL,

@FCT Varchar(1000),

@Resultado Varchar(1000)=null output--Opcional

)

As

BEGIN

DECLARE @FC AS VARCHAR(500)

SET @FC=LTRIM(RTRIM(@FCT))

BEGIN TRY

IF @FC NOT IN('I','U','D','SA','SO')

BEGIN

RAISERROR('Escolha uma das operações!',50,1)with log

END

ELSE

BEGIN

IF @FC='I'

BEGIN

INSERT tb\_TipoPagamento(

Descricao

)

VALUES(

@Descricao

)

if @@error=0 set @Resultado='Registo inserido com sucesso'

END

IF @FC='U'

BEGIN

UPDATE tb\_TipoPagamento SET

Descricao=@Descricao

if @@error=0 set @Resultado='Registo alterado com sucesso'

END

IF @FC='D'

BEGIN

DELETE FROM tb\_TipoPagamento

WHERE Id\_TipoPagamento=@Id\_TipoPagamento

if @@error=0 set @Resultado='Registo eliminado com sucesso'

END

IF @FC='SA'

BEGIN

SELECT \* FROM tb\_TipoPagamento WITH (NOLOCK)

if @@rowcount=0 set @Resultado='Não existe registo para o criterio seleccionado'

END

IF @FC='SO'

BEGIN

SELECT \* FROM tb\_TipoPagamento WITH (NOLOCK)

WHERE Id\_TipoPagamento=@Id\_TipoPagamento

if @@rowcount=0 set @Resultado='Não existe registo para o criterio seleccionado'

END

END

END TRY

BEGIN CATCH

DECLARE @ErrorMessage NVARCHAR(4000)

DECLARE @ErrorSeverity INT

DECLARE @ErrorState INT

SELECT @ErrorMessage=ERROR\_MESSAGE(),

@ErrorSeverity=ERROR\_SEVERITY(),

@ErrorState=ERROR\_STATE()

RAISERROR(@ErrorMessage,@ErrorSeverity,@ErrorState)

END CATCH

RETURN

END

GO

/\*\*\*\*\*\* Object: StoredProcedure [dbo].[Sp\_TipoServico] Script Date: 02/10/2018 07:10:15 \*\*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

--Gerado por RMSA(Romário Meireles Sebastião Ângelo)

CREATE proc [dbo].[Sp\_TipoServico]

(

@Id\_TipoServico tinyint=NULL,

@Descricao varchar(500)=NULL,

@Id\_Usuario int=NULL,

@FCT Varchar(1000),

@Resultado Varchar(1000)=null output--Opcional

)

As

BEGIN

DECLARE @FC AS VARCHAR(500)

SET @FC=LTRIM(RTRIM(@FCT))

BEGIN TRY

IF @FC NOT IN('I','U','D','SA','SO','SA')

BEGIN

RAISERROR('Escolha uma das operações!',50,1)with log

END

ELSE

BEGIN

IF @FC='I'

BEGIN

INSERT tb\_TipoServico(

Descricao,

Id\_Usuario

)

VALUES(

@Descricao,

@Id\_Usuario

)

if @@error=0 set @Resultado='Registo inserido com sucesso'

END

IF @FC='U'

BEGIN

UPDATE tb\_TipoServico SET

Descricao=@Descricao,

Id\_Usuario=@Id\_Usuario

if @@error=0 set @Resultado='Registo alterado com sucesso'

END

IF @FC='D'

BEGIN

DELETE FROM tb\_TipoServico

WHERE Id\_TipoServico=@Id\_TipoServico

if @@error=0 set @Resultado='Registo eliminado com sucesso'

END

IF @FC='SA'

BEGIN

SELECT \* FROM tb\_TipoServico WITH (NOLOCK)

if @@rowcount=0 set @Resultado='Não existe registo para o criterio seleccionado'

END

IF @FC='SO'

BEGIN

SELECT \* FROM tb\_TipoServico WITH (NOLOCK)

WHERE Id\_TipoServico=@Id\_TipoServico

if @@rowcount=0 set @Resultado='Não existe registo para o criterio seleccionado'

END

IF @FC='SA'

BEGIN

select\* from tb\_Usuario

if @@rowcount=0 set @Resultado='Não existe registo para o criterio seleccionado'

END

END

END TRY

BEGIN CATCH

DECLARE @ErrorMessage NVARCHAR(4000)

DECLARE @ErrorSeverity INT

DECLARE @ErrorState INT

SELECT @ErrorMessage=ERROR\_MESSAGE(),

@ErrorSeverity=ERROR\_SEVERITY(),

@ErrorState=ERROR\_STATE()

RAISERROR(@ErrorMessage,@ErrorSeverity,@ErrorState)

END CATCH

RETURN

END

GO

/\*\*\*\*\*\* Object: StoredProcedure [dbo].[Sp\_TipoUsuario] Script Date: 02/10/2018 07:10:15 \*\*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

--Gerado por RMSA(Romário Meireles Sebastião Ângelo)

Create proc [dbo].[Sp\_TipoUsuario]

(

@Id\_TipoUsuario int=NULL,

@Descricao varchar(500)=NULL,

@FCT Varchar(1000),

@Resultado Varchar(1000)=null output--Opcional

)

As

BEGIN

DECLARE @FC AS VARCHAR(500)

SET @FC=LTRIM(RTRIM(@FCT))

BEGIN TRY

IF @FC NOT IN('I','U','D','SA','SO')

BEGIN

RAISERROR('Escolha uma das operações!',50,1)with log

END

ELSE

BEGIN

IF @FC='I'

BEGIN

INSERT tb\_TipoUsuario(

Descricao

)

VALUES(

@Descricao

)

if @@error=0 set @Resultado='Registo inserido com sucesso'

END

IF @FC='U'

BEGIN

UPDATE tb\_TipoUsuario SET

Descricao=@Descricao

if @@error=0 set @Resultado='Registo alterado com sucesso'

END

IF @FC='D'

BEGIN

DELETE FROM tb\_TipoUsuario

WHERE Id\_TipoUsuario=@Id\_TipoUsuario

if @@error=0 set @Resultado='Registo eliminado com sucesso'

END

IF @FC='SA'

BEGIN

SELECT \* FROM tb\_TipoUsuario WITH (NOLOCK)

if @@rowcount=0 set @Resultado='Não existe registo para o criterio seleccionado'

END

IF @FC='SO'

BEGIN

SELECT \* FROM tb\_TipoUsuario WITH (NOLOCK)

WHERE Id\_TipoUsuario=@Id\_TipoUsuario

if @@rowcount=0 set @Resultado='Não existe registo para o criterio seleccionado'

END

END

END TRY

BEGIN CATCH

DECLARE @ErrorMessage NVARCHAR(4000)

DECLARE @ErrorSeverity INT

DECLARE @ErrorState INT

SELECT @ErrorMessage=ERROR\_MESSAGE(),

@ErrorSeverity=ERROR\_SEVERITY(),

@ErrorState=ERROR\_STATE()

RAISERROR(@ErrorMessage,@ErrorSeverity,@ErrorState)

END CATCH

RETURN

END

GO

/\*\*\*\*\*\* Object: StoredProcedure [dbo].[Sp\_Usuario] Script Date: 02/10/2018 07:10:15 \*\*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

--Gerado por RMSA(Romário Meireles Sebastião Ângelo)

CREATE proc [dbo].[Sp\_Usuario]

(

@Id\_Usuario int=NULL,

@Login varchar(500)=NULL,

@Telefone varchar(500)=NULL,

@Email varchar(500)=NULL,

@Senha varchar(500)=NULL,

@Status tinyint=NULL,

@Id\_TipoUsuario int=NULL,

@FCT Varchar(1000),

@Resultado Varchar(1000)=null output--Opcional

)

As

BEGIN

DECLARE @FC AS VARCHAR(500)

SET @FC=LTRIM(RTRIM(@FCT))

BEGIN TRY

IF @FC NOT IN('I','U','D','SA','SO','ST','LG')

BEGIN

RAISERROR('Escolha uma das operações!',50,1)with log

END

ELSE

BEGIN

IF @FC='I'

BEGIN

INSERT tb\_Usuario(

Login,

Telefone,

Email,

Senha,

Status,

Id\_TipoUsuario

)

VALUES(

@Login,

@Telefone,

@Email,

@Senha,

@Status,

@Id\_TipoUsuario

)

if @@error=0 set @Resultado='Registo inserido com sucesso'

END

IF @FC='U'

BEGIN

UPDATE tb\_Usuario SET

Login=@Login,

Telefone=@Telefone,

Email=@Email,

Senha=@Senha,

Status=@Status,

Id\_TipoUsuario=@Id\_TipoUsuario

if @@error=0 set @Resultado='Registo alterado com sucesso'

END

IF @FC='D'

BEGIN

DELETE FROM tb\_Usuario

WHERE Id\_Usuario=@Id\_Usuario

if @@error=0 set @Resultado='Registo eliminado com sucesso'

END

IF @FC='SA'

BEGIN

SELECT \* FROM tb\_Usuario WITH (NOLOCK)

if @@rowcount=0 set @Resultado='Não existe registo para o criterio seleccionado'

END

IF @FC='SO'

BEGIN

SELECT \* FROM tb\_Usuario WITH (NOLOCK)

WHERE Id\_Usuario=@Id\_Usuario

if @@rowcount=0 set @Resultado='Não existe registo para o criterio seleccionado'

END

IF @FC='ST'

BEGIN

SELECT \* From tb\_TipoUsuario WITH (NOLOCK)

if @@ROWCOUNT=0 set @Resultado='Não existe registo para o critério seleccionado'

END

IF @FC='LG'

BEGIN

SELECT \* From tb\_Usuario WITH (NOLOCK)

WHERE Login=@Login and Senha=@Senha

if @@ROWCOUNT=0 set @Resultado='Não existe registo para o critério seleccionado'

END

END

END TRY

BEGIN CATCH

DECLARE @ErrorMessage NVARCHAR(4000)

DECLARE @ErrorSeverity INT

DECLARE @ErrorState INT

SELECT @ErrorMessage=ERROR\_MESSAGE(),

@ErrorSeverity=ERROR\_SEVERITY(),

@ErrorState=ERROR\_STATE()

RAISERROR(@ErrorMessage,@ErrorSeverity,@ErrorState)

END CATCH

RETURN

END

GO

/\*\*\*\*\*\* Object: StoredProcedure [dbo].[Sp\_Veiculo] Script Date: 02/10/2018 07:10:15 \*\*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

CREATE proc [dbo].[Sp\_Veiculo]

(

@Id\_Veiculo int=null,

@Matricula varchar(8)=null,

@NumShassis varchar(20)=null,

@Id\_Modelo int=null,

@FCT Varchar(1000),

@Resultado Varchar(1000)=null output--Opcional

)

As

BEGIN

DECLARE @FC AS VARCHAR(500)

SET @FC=LTRIM(RTRIM(@FCT))

BEGIN TRY

IF @FC NOT IN('I','U','D','SA','SO','SV')

BEGIN

RAISERROR('Escolha uma das operações!',50,1)with log

END

ELSE

BEGIN

IF @FC='I'

BEGIN

INSERT tb\_Veiculo(

Matricula,

NumShassis,

Id\_Modelo

)

VALUES(

@Matricula,

@NumShassis,

@Id\_Modelo

)

if @@error=0 set @Resultado='Registo inserido com sucesso'

END

IF @FC='U'

BEGIN

UPDATE tb\_Veiculo SET

Matricula=@Matricula,

NumShassis=@NumShassis,

Id\_Modelo=@Id\_Modelo

if @@error=0 set @Resultado='Registo alterado com sucesso'

END

IF @FC='D'

BEGIN

DELETE FROM tb\_Veiculo

WHERE Id\_Veiculo=@Id\_Veiculo

if @@error=0 set @Resultado='Registo eliminado com sucesso'

END

IF @FC='SA'

BEGIN

SELECT \* FROM tb\_Veiculo WITH (NOLOCK)

if @@rowcount=0 set @Resultado='Não existe registo para o criterio seleccionado'

END

IF @FC='SO'

BEGIN

SELECT \* FROM tb\_Veiculo WITH (NOLOCK)

WHERE Id\_Veiculo=@Id\_Veiculo

if @@rowcount=0 set @Resultado='Não existe registo para o criterio seleccionado'

END

IF @FC='SV'

BEGIN

select\* from tb\_ModeloVeiculo

if @@rowcount=0 set @Resultado='Não existe registo para o criterio seleccionado'

END

END

END TRY

BEGIN CATCH

DECLARE @ErrorMessage NVARCHAR(4000)

DECLARE @ErrorSeverity INT

DECLARE @ErrorState INT

SELECT @ErrorMessage=ERROR\_MESSAGE(),

@ErrorSeverity=ERROR\_SEVERITY(),

@ErrorState=ERROR\_STATE()

RAISERROR(@ErrorMessage,@ErrorSeverity,@ErrorState)

END CATCH

RETURN

END

GO

/\*\*\*\*\*\* Object: StoredProcedure [dbo].[Sptb\_Cliente] Script Date: 02/10/2018 07:10:15 \*\*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

--Gerado por RMSA(Romário Meireles Sebastião Ângelo)

CREATE proc [dbo].[Sptb\_Cliente]

(

@Id\_Cliente int=NULL,

@BI\_Cliente varchar(500)=NULL,

@Nome varchar(500)=NULL,

@Telefone varchar(500)=NULL,

@Email varchar(500)=NULL,

@Login varchar(500)=NULL,

@Senha varchar(500)=NULL,

@Id\_Rua int=NULL,

@FCT Varchar(1000),

@Resultado Varchar(1000)=null output--Opcional

)

As

BEGIN

DECLARE @FC AS VARCHAR(500)

SET @FC=LTRIM(RTRIM(@FCT))

BEGIN TRY

IF @FC NOT IN('I','U','D','SA','SO','SR','LGC')

BEGIN

RAISERROR('Escolha uma das operações!',50,1)with log

END

ELSE

BEGIN

IF @FC='I'

BEGIN

INSERT tb\_Cliente(

BI\_Cliente,

Nome,

Telefone,

Email,

Login,

Senha,

Id\_Rua

)

VALUES(

@BI\_Cliente,

@Nome,

@Telefone,

@Email,

@Login,

@Senha,

@Id\_Rua

)

if @@error=0 set @Resultado='Registo inserido com sucesso'

END

IF @FC='U'

BEGIN

UPDATE tb\_Cliente SET

BI\_Cliente=@BI\_Cliente,

Nome=@Nome,

Telefone=@Telefone,

Email=@Email,

Login=@Login,

Senha=@Senha,

Id\_Rua=@Id\_Rua

if @@error=0 set @Resultado='Registo alterado com sucesso'

END

IF @FC='D'

BEGIN

DELETE FROM tb\_Cliente

WHERE Id\_Cliente=@Id\_Cliente

if @@error=0 set @Resultado='Registo eliminado com sucesso'

END

IF @FC='SA'

BEGIN

SELECT \* FROM tb\_Cliente WITH (NOLOCK)

if @@rowcount=0 set @Resultado='Não existe registo para o criterio seleccionado'

END

IF @FC='SO'

BEGIN

SELECT \* FROM tb\_Cliente WITH (NOLOCK)

WHERE Id\_Cliente=@Id\_Cliente

if @@rowcount=0 set @Resultado='Não existe registo para o criterio seleccionado'

END

IF @FC='SR'

BEGIN

select\* from tb\_Rua

if @@rowcount=0 set @Resultado='Não existe registo para o criterio seleccionado'

END

IF @FC='LGC'

BEGIN

SELECT \* From tb\_Cliente WITH (NOLOCK)

WHERE Login=@Login and Senha=@Senha

if @@ROWCOUNT=0 set @Resultado='Não existe registo para o critério seleccionado'

END

END

END TRY

BEGIN CATCH

DECLARE @ErrorMessage NVARCHAR(4000)

DECLARE @ErrorSeverity INT

DECLARE @ErrorState INT

SELECT @ErrorMessage=ERROR\_MESSAGE(),

@ErrorSeverity=ERROR\_SEVERITY(),

@ErrorState=ERROR\_STATE()

RAISERROR(@ErrorMessage,@ErrorSeverity,@ErrorState)

END CATCH

RETURN

END

GO

USE [master]

GO

ALTER DATABASE [db\_EmpresaDartin1] SET READ\_WRITE

GO



