

FACULDADE SENAI FATESG
PÓS GRADUAÇÃO EM PROJETO E
DESENVOLVIMENTO DE SOFTWARE PARA
MOBILE

FABRICIO NOGUEIRA DOS SANTOS

Orçamento WEB

**Desenvolvimento de uma infra-estrutura integrada para
criação e controle de orçamentos e ordens de serviço para
centros automotivos de pequeno e médio porte e
profissionais liberais**

Goiânia
2017

FABRICIO NOGUEIRA DOS SANTOS

Orçamento WEB

Desenvolvimento de uma infra-estrutura integrada para criação e controle de orçamentos e ordens de serviço para centros automotivos de pequeno e médio porte e profissionais liberais

Trabalho de Conclusão de Curso apresentado a PÓS GRADUAÇÃO EM PROJETO E DESENVOLVIMENTO DE SOFTWARE PARA MOBILE da FACULDADE SENAI FATESG, como requisito parcial para obtenção do título de Especialista em Projeto e desenvolvimento de software para mobile.

Orientador: Prof. Msc. Edjalma Queiroz da Silva

Goiânia
2017

FABRICIO NOGUEIRA DOS SANTOS

Orçamento WEB

Desenvolvimento de uma infra-estrutura integrada para criação e controle de orçamentos e ordens de serviço para centros automotivos de pequeno e médio porte e profissionais liberais

Trabalho de Conclusão apresentado à Coordenação do Curso de PÓS GRADUAÇÃO EM PROJETO E DESENVOLVIMENTO DE SOFTWARE PARA MOBILE da FACULDADE SENAI FATESG como requisito parcial para obtenção do título de Especialista em Projeto e desenvolvimento de software para mobile, aprovada em 17 de Maio de 2017, pela Banca Examinadora constituída pelos professores:

Prof. Msc. Edjalma Queiroz da Silva
FACULDADE SENAI FATESG
Presidente da Banca

Prof. à definir
FACULDADE SENAI FATESG

Prof. à definir
FACULDADE SENAI FATESG

Todos os direitos reservados. É proibida a reprodução total ou parcial do trabalho sem autorização da universidade, do autor e do orientador(a).

Fabricio Nogueira dos Santos

Todos os direitos desse trabalho está reservado ao autor.

Dedico este trabalho primeiramente a Deus que me deu força e saúde para me manter firme em mais essa etapa de minha vida. A Nathalia, minha esposa, que sempre me apoiou mesmo em momentos no qual tive que abdicar do tempo junto a ela devido aos estudos. A minha filha, Lis, que mesmo ainda estando dentro da barriga da mãe, já me fortalece e me motiva a ser uma pessoa e profissional melhor. A todos os professores que já passaram em minha vida, que pacientemente me ensinaram, orientaram e apontaram o caminho certo a seguir. Eles podem ter passado, seus ensinamentos não. Muito obrigado a todos!

Agradecimentos

Algumas pessoas foram essenciais para a realização desse trabalho. Colaborando direta ou indiretamente, muita gente teve um papel fundamental para que os objetivos fossem concretizados. Com o trabalho pronto, chegou a hora de agradecer. Ao orientador Edjalma Queiroz da Silva, pela oportunidade, pelos ensinamentos fornecidos. A Faculdade de Tecnologia e Desenvolvimento Gerencial Senai, pela excelente estrutura, suporte e ambiente oferecidos aos alunos, à secretaria, o coordenador do curso e a todos os professores, principalmente aos que tive a oportunidade de conhecer e aprender um pouco mais com eles. Aos colegas de classe que contribuíram diretamente e indiretamente. E à minha família, que sempre acreditou em meu potencial.

A todos, o meu singelo muito obrigado!

Resumo

Santos, F. N. **Orçamento WEB** – Desenvolvimento de uma infra-estrutura integrada para criação e controle de orçamentos e ordens de serviço para centros automotivos de pequeno e médio porte e profissionais liberais. Goiânia, 2017. 56p. Trabalho de Conclusão de Curso. PÓS GRADUAÇÃO EM PROJETO E DESENVOLVIMENTO DE SOFTWARE PARA MOBILE, FACULDADE SENAI FATESG.

O Orçamento WEB é uma suite, ou seja, um conjunto de sistemas, multi-plataforma, integrados, que através da arquitetura *RestFul* se comunicam através dos protocolos HTTP e assim formam o sistema por completo. Trata-se de um sistema para criação de orçamentos e ordens de serviço (O.S) on-line de forma simples, rápida e direta. Para esse projeto, o sistema é composto por uma *API*, que disponibiliza os serviços, regras de negócio e acesso ao dados, desenvolvida em *JavaScript* com NodeJS, banco de dados PostgreSQL. Um cliente Web, desenvolvido em Html5, Css3 e *JavaScript* utilizando o *framework* Aurelia em sua versão 1.0.0 para ser o console administrativo do sistema, em suma, será a interface de comunicação com a *API* para administração, criação e manipulação dos orçamentos e ordens de serviço. Um cliente móvel desenvolvido em Android nativo que será disponibilizado para que o cliente (humano), que nesse caso, uma pessoa que fará um serviço em seu veículo em um centro automotivo que irá utilizar o sistema. E através do aplicativo, poderá acompanhar as etapas e status da ordem de serviço. Terá a opção de gerar relatórios das manutenções já realizadas em seus veículos, receber notificações de aviso quando um ordem de serviço ativa for encerrada. O sistema não abrangerá a parte administrativa / financeiro do negócio, apenas contará com cadastro de peças, serviços, clientes, veículos, orçamentos que podem ser convertidos ou não em uma ordem de serviço para um veículo de um cliente (humano). O foco do trabalho será a integração entre as diferentes plataformas, arquitetura dos sistemas nos diferentes ambientes e a relação entre eles de forma homogênea e transparente.

Palavras-chave

Mobile, Web, REST, Restful, Desenvolvimento.

Abstract

Santos, F. N. – Desenvolvimento de uma infra-estrutura integrada para criação e controle de orçamentos e ordens de serviço para centros automotivos de pequeno e médio porte e profissionais liberais. Goiânia, 2017. 56p. Trabalho de Conclusão de Curso. PÓS GRADUAÇÃO EM PROJETO E DESENVOLVIMENTO DE SOFTWARE PARA MOBILE, FACULDADE SENAI FATESG.

The Orçamento WEB is a suite, that is, a set of multi-platform, integrated systems that through the RestFul architecture communicate through HTTP protocols and thus form the system completely. It is a system for creating budgets and work orders (W.O) online simply, quickly and directly. For this project, the system consists of an API, which provides services, business rules and data access, developed in JavaScript with NodeJS with PostgreSQL database. A web client, developed in Html5, Css3 and JavaScript, using Aurelia framework 1.0.0 to be the administrative console of the system, in short, will be the communication interface with the API for administration, creation and manipulation of budgets and work orders. A mobile client developed in native Android that will be made available to the customer (human), in that case, a person who will do a service on your vehicle in an automotive center that will use the system. And through the application, you can follow the steps and status of the work order. You will have the option to generate maintenance reports already performed on your vehicles, receive notice notifications when an active work order is closed. The system will not cover the administrative/financial part of the business, it will only count parts, services, customers, vehicles, budgets that can be converted into a service order for a (human) customer's vehicle. The focus of the work will be the integration between the different platforms, systems architecture in the different environments and the relation between them in a homogeneous and transparent way.

Keywords

Mobile, Web, Rest, RestFul, Development

Sumário

Lista de Figuras	11
Lista de Tabelas	12
1 Introdução	14
1.1 Finalidade	14
1.2 Objetivo Geral	15
2 Escopo do Projeto	16
2.1 Situação Atual	16
2.1.1 Motivação	16
2.1.2 Problema	16
2.1.3 Justificativa	17
2.2 Objetivos	18
2.2.1 Objetivos Gerais	18
2.2.2 Objetivos Específicos	18
3 Fundamentação Teórica	19
3.1 Android	19
3.2 Doze-fatores	20
3.3 Versionamento Semântico 2.0.0	20
3.4 Web service RestFul	21
3.4.1 Cliente-Servidor	22
3.4.2 Stateless	22
4 Requisitos de software	24
4.1 Requisitos Serviço API	24
4.1.1 RF01:API - Requisições Geral	24
4.1.2 RF02:API - Autenticação	25
4.1.3 RF03:API - Peça	25
4.1.4 RF04:API - Serviço	25
4.1.5 RF05:API - Cliente	25
4.1.6 RF03:API - Veículo	25
4.1.7 RF03:API - Orçamento	26
4.1.8 RF03:API - Ordem de serviço	26
4.1.9 RF03:API - Relatorios	26
4.2 Requisitos Cliente WEB	27
4.2.1 RF01:WEB - Autenticação	27
4.2.2 RF02:WEB - Peça	27

4.2.3	RF03:WEB - Serviço	28
4.2.4	RF04:WEB - Cliente	29
4.2.5	RF05:WEB - Veículo	29
4.2.6	RF06:WEB - Orçamento	30
4.2.7	RF07:WEB - Ordem de Serviço	32
4.2.8	RF08:WEB - Itens Orçamentos e Ordens de Serviço	34
4.2.9	RF09:WEB - Aviso	34
4.2.10	RF10:WEB - Dashboard - Pannel de indicadores	35
4.3	Requisitos Cliente Mobile	38
4.3.1	RF01:MOBILE - Autenticação	38
4.3.2	RF02:MOBILE - Cache de consultas	38
4.3.3	RF02:MOBILE - Inicio	38
4.3.4	RF03:MOBILE - Relatório	38
4.3.5	RF04:MOBILE - Mensagens	39
4.4	Mapa Mental	39
4.5	Design do Software	40
5	Especificações técnicas	41
5.1	Diagramas da UML	41
5.1.1	Diagrama de Classe	42
5.1.2	Diagrama de Caso de Uso	44
5.2	Banco De Dados	45
5.3	Prototipação	46
5.4	Código	48
5.5	Frameworks e Ferramentas	49
5.5.1	Práticas adotadas em todo o projeto	49
	Versionamento de código com Git	49
	Estratégia de branches para git e release de software	49
	Versionamento semântico	49
5.5.2	Ferramentas Api	50
	Visual Studio code	50
	Docker	50
5.5.3	Frameworks Api	50
	NodeJs	50
	Nodal	50
5.5.4	Ferramentas Web	50
5.5.5	Frameworks Web	51
	Aurelia	51
	Typescript	51
5.5.6	Ferramentas Mobile	51
	Android Studio	51
5.5.7	Frameworks Mobile	51
	Firebase	51
	Retrofit	51
	Glide	51
	RxAndroid	51
	EventBus	52

	Hellocharts	52
	Realm	52
5.5.8	Documentação	52
	Astah Community	52
6	Considerações Finais	53
6.1	Conclusão	53
6.1.1	Contribuição	53
6.2	Trabalhos Futuros	53
	Referências Bibliográficas	55

Lista de Figuras

2.1	Listagem de programas amostragem	17
3.1	Versões Android	19
3.2	Cliente-Servidor	22
3.3	Cliente-Stateless-Server	23
4.1	Mapa Mental do projeto	39
5.1	Representação dos diagramas da UML	41
5.2	Diagrama de Classe	43
5.3	Caso de Uso	44
5.4	PostgreSQL	45
5.5	Protótipo - Tela login	46
5.6	Protótipo - Formulário de peças	47
5.7	Protótipo - Formulário cliente	47

Lista de Tabelas

4.1	Lista de Requisitos Não Funcionais API	27
4.2	Lista de Requisitos Não Funcionais WEB	37

Lista de Códigos

5.1 Gerenciador das requisições

48

Introdução

A principal finalidade do trabalho será a arquitetura de uma plataforma *Restful* integrada que seja altamente escalável, modular e ao mesmo tempo simplificada, aberta a agregação e desenvolvimento de novos módulos com um mínimo esforço necessário. E para isso, o presente trabalho terá como foco o desenvolvimento de uma solução para centros automotivos. Uma ferramenta simples onde será possível gerar orçamentos e convertê-los em ordens de serviço. Serão desenvolvidos três atores, uma *API* (*Application Programming Interface*) ou *WebService*, que funcionará de forma independente e será responsável por integrar e disponibilizar serviços para seus clientes. Um cliente Web responsável por controlar e cadastrar dados na *API*, funcionará como a View, pensando em um sistema desenvolvido no modelo *MVC* (*Model View Controller*) e um cliente móvel, desenvolvido em Android que será disponibilizado para os clientes (humano) do centro automotivo que possibilitará a consulta de seus dados, visualizar seus orçamentos e Ordens de serviço, receber informativos enviados pelo centro automotivo, ser informado quando uma Ordem de serviço for finalizada. O cliente web deverá contar com um controle de autenticação, tanto quanto o cliente móvel, isso quer dizer que, a *API* deverá disponibilizar um serviço de geração de *Hash* de segurança para identificação de seus clientes. Isso se deve ao fato de que os dados gerados serão sensíveis e pessoais.

1.1 Finalidade

O presente trabalho tem por finalidade aplicar os conhecimentos adquiridos ao longo da pós-graduação em um sistema que contemple o desenvolvimento e integração de sistemas utilizando a arquitetura *RestFul* através do protocolo *HTTP*. Desenvolver uma *API* (*WebService*), cliente web e cliente móvel (Android).

1.2 Objetivo Geral

A pesquisa tem como objetivo geral: a integração entre os sistemas através de um serviço web *RestFul*.

Escopo do Projeto

Este documento define a modelagem do sistema prevendo uma visão da documentação do sistema Orçamento WEB que poderá ser utilizado pela equipe de desenvolvimento para documentar os requisitos, modelos, tecnologias e arquitetura utilizadas no projeto do sistema.

2.1 Situação Atual

Atualmente, através de uma pesquisa empírica sobre o mercado de softwares para centros automotivos. Foi identificado que em sua grande maioria, as soluções estão disponíveis apenas para a plataforma Windows e praticamente todos os que foram tomados na amostragem, não possuíam uma forma de integração com outras plataformas, nem mesmo com dispositivos móveis. A Figura 2.1 apresenta a pesquisa realizada no popular site para downloads Baixaki .Pesquisa realizada em 10 de Abril de 2017.

2.1.1 Motivação

O Orçamento WEB será um sistema destinado a pequenas empresas e profissionais liberais, terá como objetivo, ser a forma mais simples de gerenciar clientes, orçamentos e ordens de serviço em seus negócios. Com o conhecimento adquirido ao longo do curso, o projeto foi criado para colocar em prática tudo o que foi aprendido. Uma das principais motivações foi a de criar uma ferramenta simples e de fácil utilização para seus usuários, visando sempre as boas práticas de usabilidade e experiência do usuário final. Desenvolver em um ambiente totalmente Web utilizando tecnologias e práticas modernas para desenvolvimento de sistemas integrados e independente de plataformas.

2.1.2 Problema

Há no mercado muitas ferramentas para gerenciamento de centros automotivos, porém, em sua grande maioria Desktop, ou seja, não são multi-plataforma, rodam apenas
























ordens de serviço						
Mostrando de 1 a 23 de 23 programas encontrados						
	Relevância	Nome	Data	Nota	Licença	Downloads
		OS - Ordens de Serviço 1.9.0 Controle de ordens de serviço	01/12/2009	★★★★★ 0 votos	Gratuito para testar 517 KB	Total 22.356
		Tougg 1.0 Controle suas ordens de serviço online facilitando o cadastro de clientes e organizando sua empresa	20/12/2013	★★★★★ 1 voto	Gratuito para testar	Total 7.470 1
		Tech OS Basic 1.00.1 Sistema completo de Ordens de Serviço.	04/12/2009	★★★★★ 1 voto	Gratuito para testar 1.102 KB	Total 5.894
		IntegraOS 2.0 Software para controle e administração de ordens de serviço	04/12/2009	★★★★★ 0 votos	Gratuito 17,30 MB	Total 11.955
		Aplicações Comerciais - Ordem de Serviço Software de ordens de serviço para empresas de manutenções em geral	19/02/2014	★★★★★ 0 votos	Gratuito para testar 10,30 MB	Total 5.559
		Henning Comércio/Serviços for Windows SILVER 1.3.3-17082007 Programa para controlar as vendas e ordens de serviço de sua empresa.	01/11/2007	★★★★★ 0 votos	Gratuito 1,11 MB	Total 93.475
		Laundry 4.0 Sistema para automação de lavanderia.	04/12/2009	★★★★★ 0 votos	Gratuito para testar 20,30 MB	Total 2.192
		Ordem de Serviço Software online para gestão de ordens de serviço, clientes, vendas e muito mais	20/02/2015	★★★★★ 0 votos	Gratuito para testar	Total 708 2
		Número Verde Gerencie suas Ordens de Serviço, Orçamentos e Clientes facilmente!	19/03/2016	★★★★★ 2 votos	Gratuito	Total 2.791 3
		OrdemWeb Sistema para ordens de serviço completamente web e com integração para smartphones	14/12/2016	★★★★★ 0 votos	Gratuito para testar	Total 130 4
		ISOftComp - Ordem de Serviço 1.0.0.2 Controle as ordens de serviço, seus produtos e ainda receba notificações via sms	14/02/2014	★★★★★ 10 votos	Gratuito para testar 25,40 MB	Total 792
		MaintEasy 1.0 Ferramenta para organizar suas Ordens de serviço e gerenciar as mesmas de forma eficiente.	14/02/2014	★★★★★ 0 votos	Gratuito para testar 22,00 MB	Total 168
		Hesk 2.1 FREE Uma opção gratuita e completa para inserir campos de ordens de serviço em seu site.	23/12/2009	★★★★★ 0 votos	Gratuito 257 KB	Total 8.130
		Manutenção Fácil Network 1.0 Sistema de gerenciamento de serviços ideal para empresas de manutenção de computadores.	04/12/2009	★★★★★ 1 voto	Gratuito para testar 7,36 MB	Total 3.022
		SmartAssist 1.3.4.72 Gerencie sua loja de assistência técnica com controle detalhado de ordens de serviço	18/03/2015	★★★★★ 0 votos	Gratuito para testar 14,08 MB	Total 64
		WorkOrder XP 1.0 Gerencie ordens de serviços.	01/12/2009	★★★★★ 0 votos	Gratuito para testar 15,38 MB	Total 2.369
		OS ConTrol 1.6.5 Programa para o gerenciamento da produção e emissão de ordens de serviços em empresas de assistência técnica.	09/02/2014	★★★★★ 0 votos	Gratuito para testar 11,17 MB	Total 12.741
		REPTecno O.S. Basic 2.0.0.264.8 Gerenciamento de assistência técnica e oficinas.	19/03/2009	★★★★★ 0 votos	Gratuito para testar 4,55 MB	Total 4.025
		Tech OS FULL 1.00.00 Os mais completo software para quem realiza manutenções de Equipamentos em geral.	10/03/2009	★★★★★ 0 votos	Gratuito para testar 6,41 MB	Total 2.897
		Conecta autoplus 6.4 Controle sua oficina com emissão de ordens de serviços e orçamentos.	10/02/2009	★★★★★ 0 votos	Gratuito para testar 4,54 MB	Total 3.932
		Sistema Motoboy Entregas Rápidas 3.07 Gerencie e administre de forma eficaz a sua empresa de entregas rápidas	04/02/2014	★★★★★ 0 votos	Gratuito para testar 5,36 MB	Total 2.159
		SystemGlass 1.10.1.607 Software especialmente desenvolvido para controle de vidraçarias com agendamentos, orçamentos e muito mais.	29/12/2009	★★★★★ 3 votos	Gratuito para testar 10,70 MB	Total 13.892
		Agora OS Sistema de gestão empresarial on-line destinado a Empresas e Profissionais Autônomos	23/02/2014	★★★★★ 1 voto	Gratuito para testar	Total 746 5

Figura 2.1: Listagem de programas amostragem

instaladas em um computador com S.O. Windows. Além de que, a grande maioria abordam várias etapas e processos para se chegar a um resultado, que no caso do sistema apresentado, é o de gerar Orçamentos e ordens de serviço sem depender de um processo burocrático para se chegar em um resultado final.

2.1.3 Justificativa

Essa primeira versão tem como foco centros automotivos de pequeno e médio porte e profissionais liberais. Por se tratar de uma solução em que não é necessário cadastro de informações cruciais para controle de estoque de itens, geração de documentos fiscais etc., apenas estreitar o relacionamento do cetro automotivo com seus clientes pelo fato de que, o cliente terá a possibilidade de instalar em seu celular o aplicativo, inicialmente apenas para Android, um ferramente que faz parte do sistema

que faz parte do sistema como um todo e assim, possibilitar ao cliente acompanhar suas ordens de serviço realizadas, orçamentos ativos e receber notificações de promoções e ofertas exclusivas oferecidas pelo centro automotivo.

2.2 Objetivos

Um sistema simples, com objetivos e processos simples e bem definidos. Integrado, independente de plataforma e que facilite a comunicação entre os centros automotivos e seus clientes. Possibilitar o cliente do centro automotivo ter em suas mãos todos os trabalhos realizados e peças adquiridas para seus veículos.

2.2.1 Objetivos Gerais

O sistema tem como finalidade inovar no gerenciamento dos orçamentos e ordens de serviço dos centros automotivos, estreitar os laços entre ele e seus clientes. E além da simplicidade, elevar a experiência do usuário, tanto do cliente Web (Centro automotivo) quanto do mobile (Cliente), ao um nível de satisfação para ótimo a excelente.

2.2.2 Objetivos Específicos

Desenvolver um sistema para um ambiente integrado e multi-plataforma focado na experiência de seus usuários.

Fundamentação Teórica

3.1 Android

Desenvolvido especialmente para dispositivos móveis, como aparelhos celulares e tablets, e agora também suportando dispositivos "vestíveis" (*wearables*), como relógios, o Android é uma plataforma composta por um sistema operacional, *middlewares* e um conjunto de aplicativos principais, como os contatos, o navegador de internet e o telefone propriamente dito. Além disso, existe o Android SDK, que é um conjunto de ferramentas e *APIs* para o desenvolvimento de aplicativos para a plataforma, utilizando a linguagem Java [MONTEIRO 2014].

Version	Codename	API	Distribution
2.2	Froyo	8	0.1%
2.3.3 - 2.3.7	Gingerbread	10	1.7%
4.0.3 - 4.0.4	Ice Cream Sandwich	15	1.6%
4.1.x	Jelly Bean	16	6.0%
4.2.x		17	8.3%
4.3		18	2.4%
4.4	KitKat	19	29.2%
5.0	Lollipop	21	14.1%
5.1		22	21.4%
6.0	Marshmallow	23	15.2%

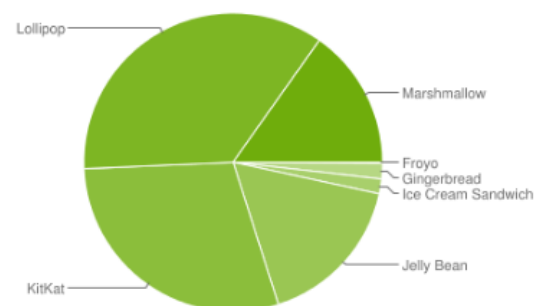


Figura 3.1: Versões Android

A Figura 3.1 representam os dados coletados durante um período de 7 dias encerrado em 1º de agosto de 2016. Todas as versões com menos de 0,1 por cento de distribuição não foram exibidas.

Baseado no Linux, o sistema operacional Android teve seu desenvolvimento iniciado em 2003 pela empresa Android Inc. Em 2005, esta foi adquirida pelo Google, que hoje lidera o desenvolvimento do Android [MONTEIRO 2014].

3.2 Doze-fatores

Na era moderna, software é comumente entregue como um serviço: denominados web apps, ou software-como-serviço. A aplicação doze-fatores é uma metodologia para construir softwares-como-serviço que:

1. Usam formatos declarativos para automatizar a configuração inicial, minimizar tempo e custo para novos desenvolvedores participarem do projeto;
2. Tem um contrato claro com o sistema operacional que o suporta, oferecendo portabilidade máxima entre ambientes que o executem;
3. São adequados para implantação em modernas plataformas em nuvem, evitando a necessidade por servidores e administração do sistema;
4. Minimizam a divergência entre desenvolvimento e produção, permitindo a implantação contínua para máxima agilidade;
5. E podem escalar sem significativas mudanças em ferramentas, arquiteturas, ou práticas de desenvolvimento.

A metodologia doze-fatores pode ser aplicada a aplicações escritas em qualquer linguagem de programação, e que utilizem qualquer combinação de serviços de suportes (banco de dados, filas, cache de memória, etc) [Wiggins].

3.3 Versionamento Semântico 2.0.0

No mundo de gerenciamento de software existe algo terrível conhecido como inferno das dependências (*"dependency hell"*). Quanto mais o sistema cresce, e mais pacotes são adicionados a ele, maior será a possibilidade de, um dia, você encontrar-se neste poço de desespero.

Em sistemas com muitas dependências, lançar novos pacotes de versões pode se tornar rapidamente um pesadelo. Se as especificações das dependências são muito amarradas você corre o risco de um bloqueio de versão (A falta de capacidade de atualizar um pacote sem ter de liberar novas versões de cada pacote dependente). Se as dependências são vagamente especificadas, você irá inevitavelmente ser mordido pela 'promiscuidade da versão' (assumindo compatibilidade com futuras versões mais do que é razoável). O inferno das dependências é onde você está quando um bloqueio de versão

e/ou promiscuidade de versão te impede de seguir em frente com seu projeto de maneira fácil e segura.

Como uma solução para este problema utilizo um conjunto simples de regras e requisitos que ditam como os números das versões são atribuídos e incrementados.

Dado um número de versão *MAJOR.MINOR.PATCH*, incremente a:

1. Versão Maior(*MAJOR*) quando fizer mudanças incompatíveis na *API*;
2. Versão Menor(*MINOR*) quando adicionar funcionalidades mantendo compatibilidade;
3. Versão de Correção(*PATCH*) quando corrigir falhas mantendo compatibilidade.

Rótulos adicionais para pré-lançamento(pre-release) e metadados de construção(build) estão disponíveis como extensão ao formato *MAJOR.MINOR.PATCH* [Versionamento-semantico] / [Preston-Werner].

3.4 Web service RestFul

Web service é uma tecnologia de acesso a dados que permite a troca de informações estruturada a partir de mensagens XML e SOAP. Esta tecnologia permite que diferentes empresas, mesmo utilizando tecnologias e plataformas distintas, conectem-se de maneira padrão e executem procedimentos remotos através da utilização do protocolo padrão da internet - HTTP.

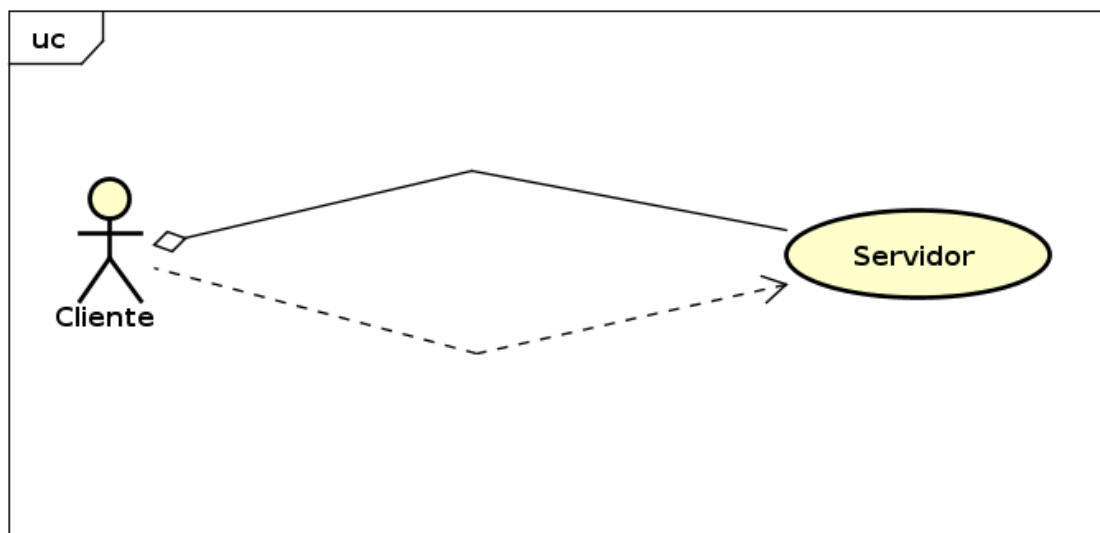
O uso do *Web service* é muito interessante, pois permite acesso a rotinas de várias empresas e com informações de tempo real visto que estas podem mudar o tempo todo.

Os *web services* podem ser implementados de muitas maneiras diferentes, cada uma com vantagens e desvantagens próprias da solução técnica adotada. Em sistemas desenvolvidos se utilizando a linguagem Java existem duas soluções, que na documentação da Oracle são denominadas como *Big Web Services* e *RESTful Web Services*. Tanto os *web services Big* quanto os *RESTful* fazem parte da *API Java para XML* (Java API for XML - JAX), que foi introduzida ao Java SE na versão 5.

Os *web services RESTful* Transferência de Estado Representacional *Representational State Transfer* são mais adequados para a utilização em cenários mais básicos, e também são melhor adaptados ao uso do protocolo HTTP do que os serviços *SOAP*. Os serviços *RESTful* são mais leves, o que significa que podem ser desenvolvidos com menor esforço, tornando-os mais fáceis de serem adotados como parte da implementação de um sistema. São mais indicados para a integração de sistemas através da internet, conferindo às soluções qualidades como escalabilidade, simplicidade e baixo acoplamento entre as partes.

3.4.1 Cliente-Servidor

Clientes e servidores são separados por uma interface por razões de interesses, isto significa que o cliente não implica em como os dados são armazenados pois é uma responsabilidade interna do servidor, assim como os servidores não se importam com a interface do usuário.



powered by Astah

Figura 3.2: *Cliente-Servidor*

A Figura 3.3 representa a associação entre o cliente e o servidor. Por ela podemos observar que o cliente é agregado ao servidor, ou seja, o servidor não é um dependente do cliente, que por sua vez, é totalmente dependente do servidor, qualquer alteração no serviço pode interferir no funcionamento dos clientes.

3.4.2 Stateless

É um importante conceito em relação aos elementos do estilo desta arquitetura. Representa uma restrição as interações entre o cliente e o servidor, a comunicação será *client-stateless-server (CSS)*, isto significa que para cada requisição será enviada toda informação necessária para o entendimento da requisição e não poderá re-utilizar nenhum contexto armazenado no servidor.

Toda requisição deve ser auto-suficiente e você deve manter o estado da sessão no cliente. Com este conceito você tem alguns benefícios, tais eles como:

Visibilidade O pacote de requisição de dados contém todas as informações necessárias para responder a solicitação, diminuindo o trabalho do servidor.

Confiabilidade É melhorada porque facilita a tarefa de recuperação de falhas.

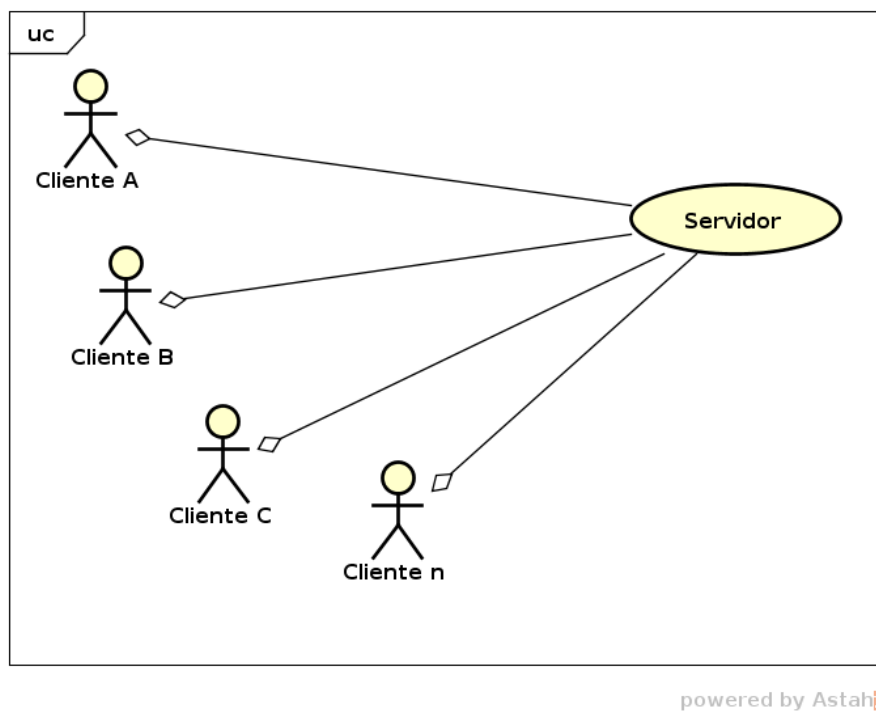


Figura 3.3: *Cliente-Stateless-Server*

Escalabilidade Não há necessidade de manter o estado das solicitações, permitindo que o servidor se livre dos recursos alocados rapidamente e ainda simplificando a implementação [ICS] / [Fielding].

Requisitos de software

Os requisitos de software descrevem explicitamente as funcionalidades e serviços do sistema. O requisitos funcionais são responsáveis por conjunto de entradas, seu comportamento e sua saída, ou seja, envolve cálculos, lógicas de trabalho, manipulação e processamento de dados, entre outros. As subseções asseguir descrevem os requisitos funcionais da sistema Orçamento WEB .

4.1 Requisitos Servico API

A sigla *API* refere-se ao termo em inglês "*Application Programming Interface*" que significa em tradução para o português "Interface de Programação de Aplicativos". Será responsável por fornecer recursos para seus clientes mediante o fornecimento de um hash de autenticação. É responsável por realizar o acesso a base de dados e conter as regras de negócio do sistema. Deve prover a integração entre os sistemas.

4.1.1 RF01:API - Requisições Geral

O serviço não deve armazenar o estado da requisição, toda requisição deve ser entendida como isolada, ou seja, deve se comportar como um serviço burro em que toda e qualquer ação, todas as informações necessárias devem ser enviados pelo cliente solicitante. Isso é conhecido como uma API Stateless.

1. As respostas aos clientes, devem seguir no formato JSON;
2. Todas as requisições devem ser realizadas somente pelos protocolos HTTP:
 - (a) GET - Recupera todos os registros;
 - (b) GET - Enviando um código identificador, recupera apenas o recurso informado;
 - (c) POST - Realiza o cadastro dos itens;
 - (d) PUT - Enviando um código identificador, deve atualizar o recurso informado;
 - (e) DELETE - Enviando um código identificador, deve excluir ou inativar o recurso.

4.1.2 RF02:API - Autenticação

Os recursos somente podem ser acessados por usuários autenticados. Cada requisição deve conter um hash de acesso no cabeçalho que foi disponibilizado para o cliente no momento em que ele realizou a autenticação. Esse hash deve ser válido apenas por 24 horas contadas à partir de sua criação.

4.1.3 RF03:API - Peça

1. Listar todas as peças;
2. Listar uma peça por identificador;
3. Cadastrar uma peça;
4. Editar uma peça;
5. Excluir, exclusão lógica, uma peça;

4.1.4 RF04:API - Serviço

1. Listar todos os serviços;
2. Listar um serviço por identificador;
3. Cadastrar um serviço;
4. Editar um serviço;
5. Excluir, exclusão lógica, um serviço;

4.1.5 RF05:API - Cliente

1. Listar todos os clientes;
2. Listar uma cliente por identificador;
3. Cadastrar um cliente;
4. Editar um cliente;
5. Excluir, exclusão lógica, um cliente;

4.1.6 RF03:API - Veículo

1. Listar todos os veículos;
2. Listar um veículo por identificador;
3. Cadastrar um veículo;
4. Editar um veículo;
5. Excluir, exclusão lógica, um veículo;

4.1.7 RF03:API - Orçamento

1. Listar todas os orçamentos;
2. Listar um orçamento por identificador;
3. Cadastrar um orçamento;
4. Consolidar um orçamento em uma ordem de serviço;
5. Editar um orçamento. Apenas orçamentos com prazo de validade maior que a data atual podem ser editados;
6. Cancelar orçamentos que possuem prazo de validade menor que a data atual;
7. Excluir, exclusão lógica, um orçamento;

4.1.8 RF03:API - Ordem de serviço

1. Listar todas as ordens de serviço;
2. Listar uma ordem de serviço por identificador;
3. Cadastrar uma ordem de serviço;
4. Editar uma ordem de serviço;
5. Excluir, exclusão lógica, uma ordem de serviço;

4.1.9 RF03:API - Relatorios

1. Listar a última promoção válida cadastrada e enviada pelo sistema;
2. Listar o totalizador de orçamentos que:
 - (a) Vencem hoje;
 - (b) Vencem nessa semana;
 - (c) Não forma consolidados.
3. Listar o top dez de:
 - (a) Peças mais vendidas;
 - (b) Serviços mais realizados;
4. Totalizador de ordens de serviço.

Contudo, um sistema deve também atender a requisitos que não são necessariamente regras do negócio do Software. Assim, de nada adianta ter um sistema seguro, interativo e confiável se ele consome muitos recursos do computador e demora para executar a suas rotinas internas. Um sistema lento é alvo de crítica dos usuários, mesmo que seja funcional. A *performance* do software pode ser melhorada utilizando técnicas de programação orientada a objetos, gerenciamento de memória, *threads* e otimização de código.

Dessa forma, foram definidos os seguintes Requisitos Não-Funcionais do sistema proposto, que podem ser observados na Tabela 4.1.

Tabela 4.1: *Lista de Requisitos Não Funcionais API*

Cod. Requisito	Nome	Descrição
RNF01:API	Disponibilidade	O sistema deverá ter alta disponibilidade, por exemplo 99% do tempo.
RNF02:API	Processamento	O sistema deverá processar N requisições por um determinado tempo.
RNF03:API	Confiabilidade	O sistema não deverá apresentar aos usuários quaisquer dados de cunho privativo.

4.2 Requisitos Cliente WEB

A versão Web da aplicação será a interface de integração para o centro automotivo. Ela deverá de forma simples, encapsular as regras de negócio e apresentá-las a seus usuários.

4.2.1 RF01:WEB - Autenticação

O sistema deve possuir uma tela para a autenticação de seus usuários.

1. Somente usuários autorizados devem acessar o sistema;
2. Um usuário deve possuir uma senha para realizar o acesso;
3. Pessoas não autenticadas não podem ter acesso ao sistema.
4. Campos obrigatórios:
 - (a) Código numérico auto incremental
 - (b) Usuário;
 - (c) Senha.

4.2.2 RF02:WEB - Peça

Centros automotivos podem ou não realizar a venda de peças que farão parte dos serviços prestados.

1. As peças persistidas deverão ser apresentadas em uma listagem para o usuário;

2. A listagem deve possuir um campo para busca de uma peça.
3. Pela listagem deve ser possível:
 - (a) Acessar o formulário para cadastro de uma nova peça;
 - (b) Atualizar a listagem com os dados mais recentes;
 - (c) Filtrar por peças ativas ou inativas;
 - (d) Acessar o formulário para edição;
 - (e) Possibilitar a inativação através do clique em um botão. O usuário deverá confirmar essa ação;
 - (f) A exclusão não deve ser física, apenas lógica, ou seja, a funcionalidade excluir deve inativar o item não excluí-lo do banco de dados.
4. Campos obrigatórios:
 - (a) Código numérico auto incremental
 - (b) Referência;
 - (c) Descrição.
5. Campos opcionais:
 - (a) Modelo;
 - (b) Valor sugerido para a venda;
 - (c) Observações.

4.2.3 RF03:WEB - Serviço

Centros automotivos podem ou não realizar a prestação de um serviço.

1. Os serviços persistidos deverão ser apresentados em uma listagem para o usuário;
2. A listagem deve possuir um campo para busca de um serviço;
3. Pela listagem deve ser possível:
 - (a) Acessar o formulário para cadastro de um novo serviço;
 - (b) Atualizar a listagem com os dados mais recentes;
 - (c) Filtrar por serviços ativos ou inativos;
 - (d) Acessar o formulário para edição;
 - (e) Possibilitar a inativação através do clique em um botão. O usuário deverá confirmar essa ação.
 - (f) A exclusão não deve ser física, apenas lógica, ou seja, a funcionalidade excluir deve inativar o item não excluí-lo do banco de dados.
4. Campos obrigatórios:
 - (a) Código numérico auto incremental
 - (b) Referência;

(c) Descrição.

5. Campos opcionais:

- (a) Valor sugerido para a venda;
- (b) Observações.

4.2.4 RF04:WEB - Cliente

Centros automotivos podem ou não realizar o cadastro de seus clientes. Um cliente pode possuir um ou mais veículos, pode possuir um ou mais telefones e e-mail.

1. No formulário, o usuário poderá optar por cadastrar os veículos, um ou mais, do cliente com as informações básicas obrigatórias do castro de veículo 4.2.5;
2. Os clientes persistidos deverão ser apresentados em uma listagem para o usuário;
3. A listagem deve possuir um campo para busca de um cliente;
4. Pela listagem deve ser possível:
 - (a) Visualizar um ícone que representará quais clientes possuem o aplicativo;
 - (b) Acessar o formulário para cadastro de um novo cliente;
 - (c) Atualizar a listagem com os dados mais recentes;
 - (d) Filtrar por clientes ativos ou inativos;
 - (e) Acessar o formulário para edição;
 - (f) Possibilitar a inativação através do clique em um botão. O usuário deverá confirmar essa ação.
 - (g) A exclusão não deve ser física, apenas lógica, ou seja, a funcionalidade excluir deve inativar o item não excluí-lo do banco de dados.

5. Campos obrigatórios:

- (a) Código numérico auto incremental
- (b) Nome;
- (c) Documento.

6. Campos opcionais:

- (a) Lista de telefones onde podem ser indicados os principais;
- (b) Lista de e-mails onde podem ser indicados os principais;
- (c) Observações.

4.2.5 RF05:WEB - Veículo

Centros automotivos devem realizar o cadastro de um veículo que devem pertencer a um cliente.

1. Um veículo pertence exclusivamente a um cliente, porém, ao longo do tempo, poderá ser alterado para outro cliente;
2. No formulário, um campo para localização do cliente deve estar disponível;
3. Somente cliente com o status ativo pode ser listado no campo de busca de clientes;
4. O histórico de manutenção do veículo não deverá ser perdido quando ocorrer a mudança de cliente;
5. Os veículos persistidos deverão ser apresentados em uma listagem para o usuário;
6. A listagem deve possuir um campo para busca de um veículo;
7. Pela listagem deve ser possível:
 - (a) Acessar o formulário para cadastro de um novo veículo;
 - (b) Atualizar a listagem com os dados mais recentes;
 - (c) Filtrar por veículos ativos ou inativos;
 - (d) Acessar o formulário para edição;
 - (e) Possibilitar a inativação através do clique em um botão. O usuário deverá confirmar essa ação.
 - (f) A exclusão não deve ser física, apenas lógica, ou seja, a funcionalidade excluir deve inativar o item não excluí-lo do banco de dados.
8. Campos obrigatórios:
 - (a) Código numérico auto incremental
 - (b) Cliente;
 - (c) Placa;
 - (d) Modelo.
9. Campos opcionais:
 - (a) Fabricante;
 - (b) Ano/modelo;
 - (c) Cor;
 - (d) Observações.

4.2.6 RF06:WEB - Orçamento

O orçamento servirá para a avaliação ou cálculo especulativo para o cliente sobre os custos das peças e/ou serviços a serem prestados no centro automotivo.

1. Um Orçamento pode ou não ser convertido em ordem de serviço;
2. O formulário de cadastro deve possuir um campo para localizar o veículo que será relacionado;
3. Somente veículos ativos devem ser listados no campo de busca;

4. Devem possuir 15 dias de validade, que deverá ser contabilizada à partir da data de seu cadastro;
5. Após o vencimento, fim da data de validade, o orçamento não poderá ser reaberto e deverá ser automaticamente cancelado;
6. Itens Peças e/ou Serviços ver: 4.2.8
7. Impressão em arquivo pdf do espelho do orçamento contendo data de validade, informações do cliente, peças, serviços e totalizadores;
8. Resumo do orçamento que deverá listar o valor total, desconto e subtotal das peças e serviços;
9. Opção para aplicar desconto geral no orçamento que deverá ser calculado em reais ou porcentagem;
10. Apresentar o valor total para o usuário, somatório dos itens vezes suas quantidades menos o desconto;
11. Ao se alterar um item, seja peça ou serviço, o valor deve ser calculado e apresentado instantaneamente ao cliente;
12. O orçamento pode ser consolidado em uma ordem de serviço;
13. Os orçamentos persistidos deverão ser apresentados em uma listagem para o usuário;
14. A listagem deve possuir os campos de filtro por nome do cliente, placa do veículo e código do orçamento;
15. Pela listagem deve ser possível:
 - (a) Acessar o formulário para cadastro de um novo orçamento;
 - (b) Atualizar a listagem com os dados mais recentes;
 - (c) A listagem deve apresentar os orçamentos por ordem de validade e diferenciar a cor de acordo com a proximidade:
 - i. Verde para mais de sete dias;
 - ii. Amarelo entre três e sete dias;
 - iii. Vermelho entre zero e três dias;
 - iv. Cinza vencidos, apresentá-los na listagem de inativos;
 - v. Azul consolidados, apresentá-los na listagem de consolidados.
 - (d) Filtrar por orçamentos ativos, consolidados ou cancelados;
 - (e) Acessar o formulário para edição;
 - (f) Possibilitar o cancelamento através do clique em um botão. O usuário deverá confirmar essa ação.
 - (g) A cancelamento não deve ser física, apenas lógica, ou seja, quando cancelado, para fins de histórico o orçamento deve ser inativado, não excluído do banco de dados;
 - (h) Quando cancelado, um orçamento não pode ser reaberto as únicas opções

devem ser a de visualização e impressão.

16. Campos obrigatórios:

- (a) Código numérico auto incremental
- (b) Veículo;

17. Campos opcionais:

- (a) Km atual do veículo;
- (b) Observações.
- (c) Itens:
 - i. Relação de peças;
 - ii. Relação de serviços.

4.2.7 RF07:WEB - Ordem de Serviço

A ordem de serviço é um documento que tem a função de definir as peças e serviços, suas quantidades e respectivos valores a respeito do que precisa ser efetuado no veículo do cliente do centro automotivo. Serve como um contrato entre o cliente e o centro automotivo. Permite ao cliente comprovar todo o histórico de manutenção de seus veículos.

1. Deve possuir uma data prevista para finalização;
2. O formulário de cadastro deve possuir um campo para localizar o veículo que será relacionado;
3. Somente veículos ativos devem ser listados no campo de busca;
4. Status que identificará seu estágio atual:
 - (a) Em andamento indica que está em andamento dentro do prazo de validade;
 - (b) Atrasado indica que está em andamento, porém, a data prevista para a finalização foi ultrapassada;
 - (c) Renegociado indica que está em andamento, porém a data prevista para a finalização foi ultrapassada mas houve uma renegociação com o cliente prevendo uma nova data;
 - (d) Retrabalho indica que foi finalizada porém teve que ser reaberta para uma nova intervenção;
 - (e) Cancelado indica o cancelamento;
 - (f) Finalizado o trabalho foi finalizado.
5. Um histórico dos estágios da ordem de serviço deve ser apresentado no formulário;
6. Itens Peças e/ou Serviços 4.2.8
7. Impressão em arquivo pdf do espelho do orçamento contendo data de validade, informações do cliente, peças, serviços e totalizadores;

8. Resumo da ordem de serviço que deverá listar o valor total, desconto e subtotal das peças e serviços;
9. Opção para aplicar desconto geral no orçamento que deverá ser calculado em reais ou porcentagem;
10. Apresentar o valor total para o usuário, somatório dos itens vezes suas quantidades menos o desconto;
11. Ao se alterar um item, seja peça ou serviço, o valor deve ser calculado e apresentado instantaneamente ao cliente;
12. As ordens de serviços persistidas deverão ser apresentados em uma listagem para o usuário;
13. A listagem deve possuir os campos de filtro por nome do cliente, placa do veículo e código da O.S.;
14. Pela listagem deve ser possível:
 - (a) Acessar o formulário para cadastro de uma nova O.S.;
 - (b) Atualizar a listagem com os dados mais recentes;
 - (c) A listagem deve apresentar as O.S. por ordem de previsão e diferenciar a cor de acordo com a proximidade:
 - i. Verde para dentro do prazo;
 - ii. Amarelo próximo do prazo de validade;
 - iii. Vermelho após o prazo de validade;
 - iv. Cinza cancelados, apresentá-los na listagem de cancelado;
 - v. Azul finalizados, apresentá-los na listam de finalizados.
 - (d) Na listagem, quando a ordem de serviço for gerada através de um orçamento, o código do orçamento deve ser apresentado;
 - (e) Filtrar por ativos, finalizados ou cancelados;
 - (f) Acessar o formulário para edição;
 - (g) Possibilitar o cancelamento através do clique em um botão. O usuário deverá confirmar essa ação.
 - (h) A cancelamento não deve ser física, apenas lógica, ou seja, quando cancelado, para fins de histórico o orçamento deve ser inativado, não excluído do banco de dados;
 - (i) Quando cancelado, não pode ser reaberto as únicas opções devem ser a de visualização e impressão;
 - (j) Quando finalizado, uma opção para reabertura da O.S. deve estar disponível;
15. Campos obrigatórios:
 - (a) Código numérico auto incremental;
 - (b) Veículo;

(c) Data entrada, deve ser preenchida automaticamente.

16. Campos opcionais:

- (a) Km atual do veículo;
- (b) Data e hora prevista;
- (c) Data e hora da saída;
- (d) Observações.
- (e) Itens:
 - i. Relação de peças;
 - ii. Relação de serviços.

4.2.8 RF08:WEB - Itens Orçamentos e Ordens de Serviço

Os itens não são obrigatórios, um orçamento ou ordem de serviço pode possuir peças e serviços, somente peças, somente serviços ou nenhum dos dois.

1. Através da referência, um item pode ser localizado para ser adicionado à relação dos itens;
2. Somente itens com o status ativo pode ser localizado pela busca;
3. O valor sugerido pode ser alterado;
4. A quantidade pode ser selecionada, não há restrições quanto a quantidade;
5. Pode ser definido um desconto diretamente no item que pode ser em reais ou em porcentagem;
6. Após ser acrescentado à listagem dos itens relacionados, o mesmo pode ser removido.
7. Campos obrigatórios:
 - (a) Referência.

4.2.9 RF09:WEB - Aviso

Aviso é um nome dado à funcionalidade que permitira que o centro automotivo envie notificações aos clientes que possuem o aplicativo instalado em seu smartphone informado sobre novidade ou promoções vigentes. Essa pode ser uma forma de estreitar o contato e fidelizar o cliente.

1. Os avisos persistidos deverão ser apresentados em uma listagem para o usuário;
2. A listagem deve possuir um campo para busca de um determinado aviso;
3. Pela listagem deve ser possível:
 - (a) Acessar o formulário para cadastro de um novo aviso;
 - (b) Atualizar a listagem com os dados mais recentes;

- (c) Acessar o formulário para edição;
- (d) Avisos enviados, ao invés de editar, a opção visualizar deve ser apresentada;
- (e) Excluir um aviso, exclusão física;
- (f) Copiar um determinado aviso, ou seja, gerar um novo à partir das informações do que foi copiado;
- (g) Devem possuir os status:
 - i. Enviado, quando o aviso tiver sido enviado aos clientes. Nessa opção, a data de envio deve ser informada;
 - ii. Rascunho, quando um aviso foi criado, porém não enviado ainda;
 - iii. Falha no envio, quando ao tentar enviar, por algum motivo ocorra uma falha e a ação não seja concluída com sucesso.

4. Campos obrigatórios:

- (a) Código numérico auto incremental
- (b) Título;
- (c) Resumo.

5. Campos opcionais:

- (a) Código promocional;
- (b) Data início;
- (c) Data fim;
- (d) Mensagem;
- (e) Observações, esse campo não é enviado nas mensagens.
- (f) Imagem*

6. *Sobre a imagem:

- (a) Será persistida no banco de dados codificada em um hash em base64;
- (b) Poderá ser nos formatos .jpeg ou .png;
- (c) O tamanho máximo do arquivo não deve ultrapassar 1024kb;
- (d) No formulário ao finalizar a criação de um aviso, o usuário poderá gravar como rascunho ou realizar o envio imediato.

4.2.10 RF10:WEB - Dashboard - Painel de indicadores

O painel de indicadores apresentará ao cliente, relatórios básicos

1. Promoção em vigor:

- (a) Apresentar a promoção que possui a data de validade mais próxima da data atual;
- (b) Campos:

- i. Título;
 - ii. Código da promoção;
 - iii. Data início;
 - iv. Data fim;
 - v. Thumb da imagem.
- 2. Quantitativo de orçamentos que:
 - (a) Vencem hoje, ou seja, possuem a data de validade final igual a data atual;
 - (b) Vecem essa semana, ou seja, possuem a data valiidade à partir da data atual menos sete dias.
 - (c) Totalizador dos orçamentos ativos que não foram consolidados em um ordens de serviço.
- 3. Top dez peças mais vendidas e serviços mais realizados pelo centro automotivo:
 - (a) Apresenta o campo referência do item e o quantitativo.
- 4. Totalizador Ordens de serviço:
 - (a) Grafico no formato donut informado a quantidade total de ordens de serviços cadastradas no sistema, dividido por status:
 - i. Azul para abertas;
 - ii. Vermelho para atrasadas;
 - iii. Cinza para canceladas;
 - iv. Verde para finalizadas.

Requisitos Não-Funcionais do cliente Web podem ser observados na Tabela 4.2.

Tabela 4.2: *Lista de Requisitos Não Funcionais WEB*

Cod. Requisito	Nome	Descrição
RNF01:WEB	Codigo	O sistema deverá ser desenvolvido em JavaScript, Html5 e Css3.
RNF02:WEB	Confiabilidade	O sistema não deverá apresentar aos usuários quaisquer dados de cunho privativo.
RNF03:WEB	Visual	A itenteface deve seguir a linguagem visual definidas no guia Material design da Google.
RNF04:WEB	Navegadores	<p>Deve atender aos seguintes navegadores:</p> <ol style="list-style-type: none">1. Gloogle chrome2. Firefox3. Opera4. Safari <p>O navegador Internet explorer não é requisito funcional e não funcional.</p>

4.3 Requisitos Cliente Mobile

A interface mobile deve ser desenvolvida em Android nativo e atender as versões: mínima SDK 19, máxima SDK 25. O aplicativo deve ser disponibilizado para instalações nos dispositivos através da loja de aplicativos do Google, Google play.

4.3.1 RF01:MOBILE - Autenticação

1. Para ter acesso aos dados, em seu primeiro acesso, o aplicativo deverá apresentar um formulário em que o usuário deverá informar seu código de usuário, fornecido pelo centro automotivo, e seu número de documento e caso seja um usuário válido, deverá ser redirecionado para a primeira tela do aplicativo;
2. A autenticação deve ser somente requerida no primeiro acesso do usuário;
3. Caso não seja um usuário válido, ou tenha digitado alguma informação errada, uma mensagem deve ser apresentada contendo informações do erro ocorrido.

4.3.2 RF02:MOBILE - Cache de consultas

1. Para otimizar e economizar o acesso aos dados na API, o cliente mobile deve manter em cache as informações do usuário;
2. Somente no primeiro acesso todos os dados devem ser consultados, nos demais, para atualizar as informações, a ação deve partir do usuário utilizando os métodos de atualização disponíveis no aplicativo.

4.3.3 RF02:MOBILE - Início

1. Apresentar para o cliente um gráfico contendo o somatório das ordens de serviço realizadas no centro automotivo;
2. Apresentar os últimos orçamentos em aberto, ou seja, com data de validade maior que a data atual apenas para a conferência do usuário;
3. Uma notificação, apresentada na caixa de notificações do sistema, deve informar ao usuário quando uma ordem de serviço for finalizada. Essa ação partirá do cliente Web 4.2.7

4.3.4 RF03:MOBILE - Relatório

1. O relatório deverá apresentar uma listagem com todas as ordens de serviço realizadas pelo usuário;
2. Nessa listagem devem conter apenas ordens de serviço com o status finalizado;

3. Deve possibilitar através de um clique em um item, acessar de forma completa a ordem de serviço em questão, ou seja, visualizar informações mais detalhadas como peças e serviços;
4. Possibilitar que o usuário gere um arquivo em formato pdf com as informações da ordem de serviço em questão.

4.3.5 RF04:MOBILE - Mensagens

1. A última mensagem vigente deve ser apresentada;
2. Em caso de uma nova mensagem, uma notificação deve ser apresentada na caixa de notificações do sistema contendo os dados básico dessa notificação como título e resumo e ao clicar no item, o usuário deverá ser redirecionado para a aba de notificações do aplicativo;
3. A ação de notificação de mensagem partirá do cliente Web 4.2.9.

4.4 Mapa Mental

Um mapa mental é um diagrama que se elabora para representar ideias, tarefas ou outros conceitos que se contram relacionados com uma palavra-chave ou uma ideia central, e cujas informações relacionadas em si são irradiadas.

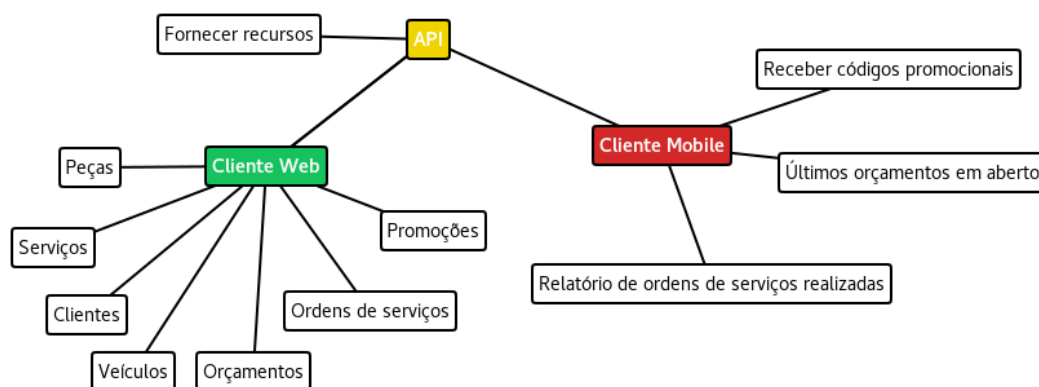


Figura 4.1: Mapa Mental do projeto

A sua principal função é geração, visualização e classificação taxonômica de ideias, pelo que serve de ajuda para o estudo, a organização de informações, a tomada de decisões e a escrita. Num mapa mental, os elementos são incluídos de forma intuitiva de acordo com a importância dos conceitos, embora se organizem nos grupos, nos ramos ou nas áreas. Segundo especialistas, este tipo de representação gráfica auxilia a memória. Em um mapa mental, é recomendado utilizar um mínimo de palavras e iniciar a tarefa sempre no centro da folha, onde se coloca portanto a ideia central. Na Figura 4.1 é possível visualizar o mapa mental proposto para o projeto.

4.5 Design do Software

O cliente Web pode ser acessado via internet pelo navegador que pode ser acessado de qualquer lugar do mundo via *WWW (World Wide Web)*. A escolha da arquitetura orientada Web permite que varias maquinas acessem um ambiente centralizado.

Nessa estrutura, o navegador web e responsavel pela interpretação da linguagem *HTML(Hyper Text Markup Language)* que envia requisições dos dados quanto ao lado do provedor de dados. Esse protocolo é chamado de *HTTP(Hyper Text Transfer Protocol)*.

No lado do provedor de dados *API*, o serviço, está toda a complexidade do sistema em si onde são determinadas as questões como regras de negocio, requisitos como segurança e armazenamento de banco de dados.

Uma grande vantagem do sistema para web é sua facil portabilidade, podendo migrar de ambiente pra outro sem a necessidade de muita configuração pós mudança.

Especificações técnicas

As especificações técnicas (ET) descrevem, de forma precisa, completa e ordenada, os recursos, processos e procedimentos adotados na construção dos sistemas do projeto.

5.1 Diagramas da UML

UML - Unified Modeling Language é uma linguagem que define uma série de artefatos que nos ajuda na tarefa de modelar e documentar os sistemas orientados a objetos que desenvolvemos. A maioria dos problemas encontrados em sistemas orientados a objetos tem sua origem na construção do modelo, no desenho do sistema.

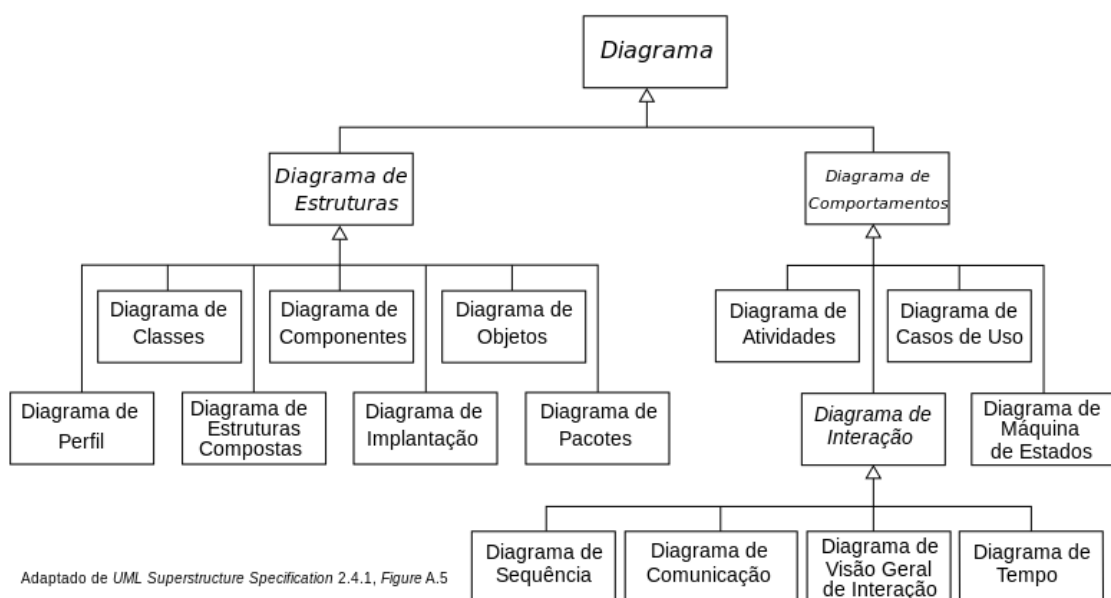


Figura 5.1: Representação dos diagramas da UML

A *UML* não é uma metodologia de desenvolvimento, o que significa que ela não diz para você o que fazer primeiro e em seguida ou como projetar seu sistema, mas ela lhe auxilia a visualizar seu desenho e a comunicação entre objetos [D’Souza 1998].

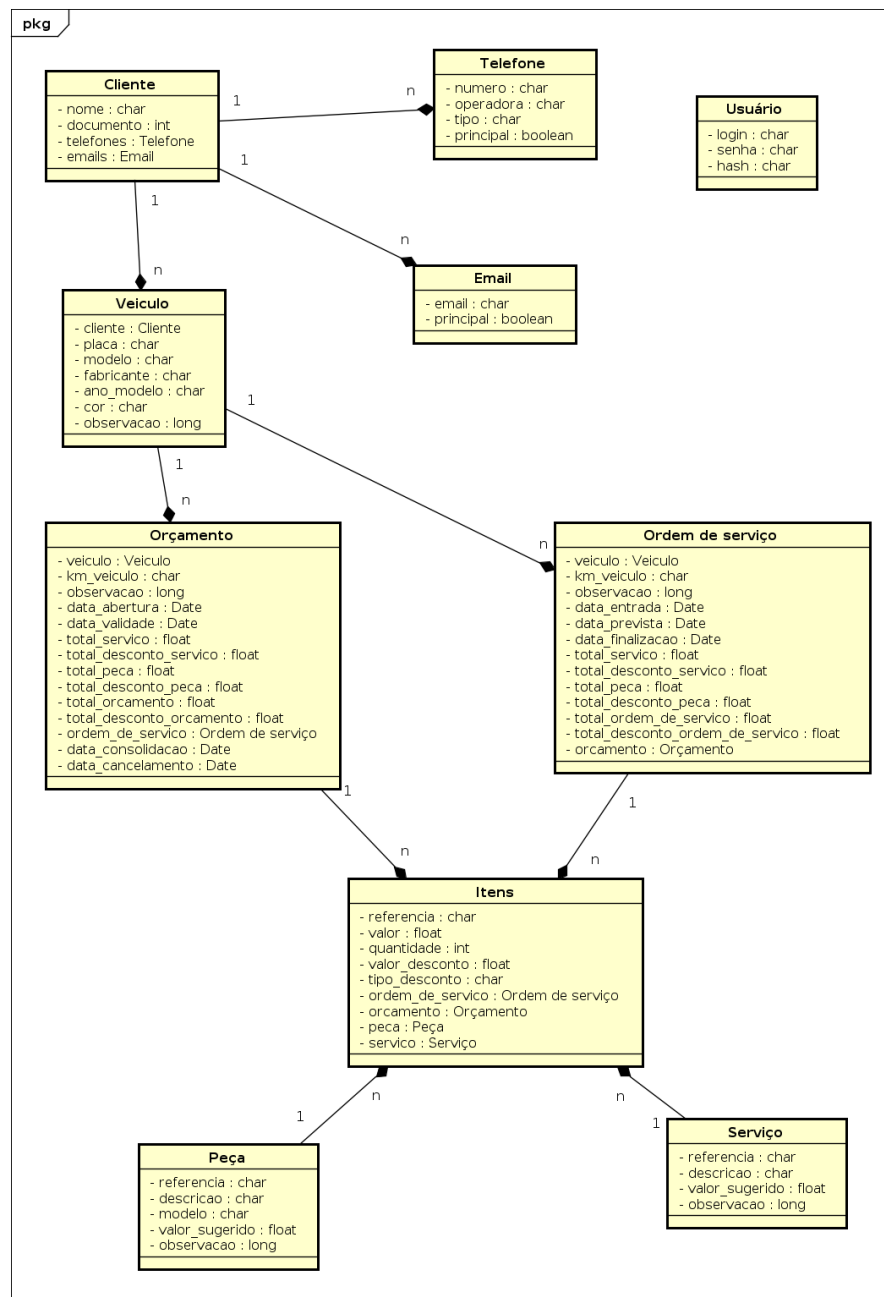
A *UML* é uma especificação da *Object Management Group - OMG (OMG, 1997 - 2011)*. É uma linguagem gráfica de modelagem para visualizar, especificar, construir e documentar os artefatos de sistemas de objetos distribuídos (UML, 2011).

A *UML* possui treze modelos gráficos que estão divididos em duas categorias, os diagramas de aplicações estáticas que representam a estrutura e os diagramas de comportamentos, no entanto dentro desta última categoria, existe uma subcategoria que compõe os diagramas de interação [Silva, Freire e Callahan 2011]. A categoria de diagramas de Estrutura inclui diagrama de classe, diagrama de objeto, diagrama de componentes, diagrama de estrutura composta, diagrama de pacote e diagrama de utilização [Silva, Freire e Callahan 2011]. Os diagramas de Comportamento são: diagrama de caso de uso, diagrama de máquina de estados e diagrama de atividades. Em sua subcategoria Interação estão inclusos os diagramas de sequência, comunicação, visão geral de interação e por último, porém não menos importante o de temporização [Silva, Freire e Callahan 2011].

Pela Figura 5.1 podemos visualizar uma representação da organização dos modelos da *UML*.

5.1.1 Diagrama de Classe

É uma modelagem muito útil para o desenvolvimento de sistemas, pois define todas as classes que o sistema necessita possuir e é a base para a construção dos diagramas de comunicação, sequência e estados. O diagrama de classes representa a estrutura do sistema, recorrendo ao conceito de classe e suas relações. O modelo de classes resulta de um processo de abstração onde são identificados os objectos relevantes do sistema em estudo. A Figura 5.2 representa a estrutura de classes da *API* do projeto.



powered by Astah

Figura 5.2: Diagrama de Classe

Cada classe é descrita através do seu nome, identificação de todos os seus atributos e identificação de todas as operações que traduzem o seu comportamento. O diagrama de classes mostra como cada classe se relaciona com as outras, tendo como objetivo, a satisfação dos requisitos funcionais definidos para o sistema em estudo. Qualquer classe e relação devem ter um nome elucidativo e claro para que o diagrama seja facilmente entendido. Depois de se terem identificado as classes e os seus atributos, há que identificar as relações que existem entre as diferentes classes, de forma a satisfazer os requisitos funcionais do sistema. É possível que na atividade de análise nem todos os

atributos tenham seus tipos definidos. Neste caso, estes elementos poderão ser adicionados na atividade de projeto, na medida do necessário. Além disso, os tipos abstratos de dados definidos nos papéis de associações poderão ser substituídos por tipos concretos [Bezerra 2007]. O diagrama de de classes lista todos os conceitos do domínio que serão implementados no sistema e as relações entre os conceitos. Ele é muito importante pois define a estrutura do sistema a desenvolver. Elaborar de forma criteriosa diagramas de classes é um fator de sucesso de projetos de software por que, além do fato de ser um momento propenso à inserção de defeitos no software, são neles em que são transformados os problemas do usuário em uma solução computacional, servindo como uma ponte entre requisitos e codificação. Se esta ponte for mal projetada, o software também será [Silveira 2011].

5.1.2 Diagrama de Caso de Uso

O diagrama de casos de uso é um diagrama da *UML* cujo objetivo é representar um requisito do sistema que será automatizado. Considere como requisito uma necessidade do sistema.

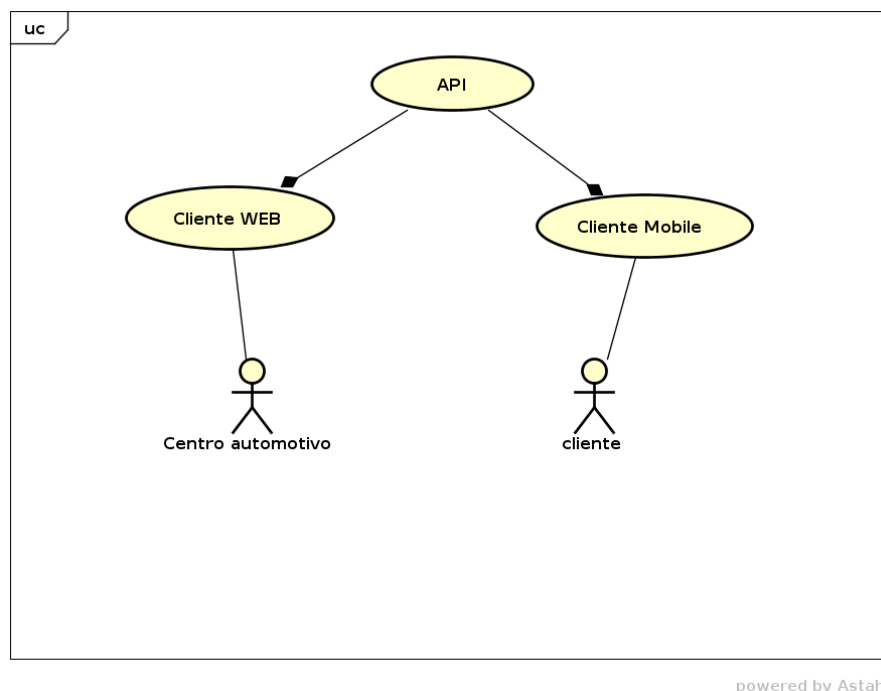


Figura 5.3: *Caso de Uso*

Na Figura 5.3 é mostrado o diagrama de caso de uso do sistema que contém o centro automotivo e seus clientes como atores de casos de usos. Os atores representam os usuários de casos de uso desempenham quando interagem com sistema e os círculos suas possíveis interações.

5.2 Banco De Dados

Na realidade econômica em que vivemos a disputa entre as empresas para conquistar uma fatia do mercado ou, pelo menos, manter os clientes que possui, as informações tornaram-se elementos essenciais para o bom funcionamento dos negócios. E se uma empresa pretende expandir seus negócios, é fundamental que as informações estejam disponíveis sempre que forem necessárias. Para isto, é imprescindível a utilização de um Sistema de Gerenciamento de Banco de Dados (SGBD).

Atualmente existem diversos softwares gerenciadores de banco de dados disponíveis no mercado. É possível classificá-los quanto a sua distribuição que pode ser livre ou proprietária. Dentre os proprietários estão, entre outros, o Oracle® e o SQL Server®. Dentre os livres o PostgreSQL é o que mais se destaca por possuir recursos que o equipara aos bancos de dados proprietários. Este fator tem contribuído para que seu uso aumente significativamente.

Banco de dados é um conjunto de dados persistentes, com o intuito de armazenar informações de uma determinada organização. Esses dados são mantidos por um software chamado de Sistema Gerenciador de Banco de Dados (SGBD), onde os usuários desse sistema podem realizar busca, exclusão, inserção e alteração nesses arquivos de banco de dados [Date 1985].

Alguns autores definem que dados e informações tem o mesmo significado, por outro lado, outros definem dados como os valores fisicamente armazenados no banco de dados e informações como o significado gerado a partir de um determinado dado [Date 1985].

Neste projeto utilizamos o PostgreSQL por ser um banco de dados livre, avançado e seguro. Hoje o PostgreSQL é um poderoso software para gerenciamento de banco de dados, agregando algumas funções dos SGBDs mais avançados, como por exemplo, os softwares proprietários *Oracle* ou *SQL Server*. É conhecido pela sua robustez e extrema segurança. Abaixo símbolo do PostgreSQL



Figura 5.4: *PostgreSQL*

O PostgreSQL oferece o mais baixo custo, reduzindo de forma significativa seus custos de administração, suporte e licenciamento e, ao mesmo tempo, fornecendo alta performance, confiabilidade e escalabilidade.

É uma solução perfeita e viável para as necessidades de pequenas e médias empresas, sendo uma alternativa aos tradicionais Bancos de dados.

5.3 Prototipação

A prototipação é uma etapa fundamental na construção de software. Nesta etapa, é prudente que todos os envolvidos no projeto também estejam envolvidos em sua criação.

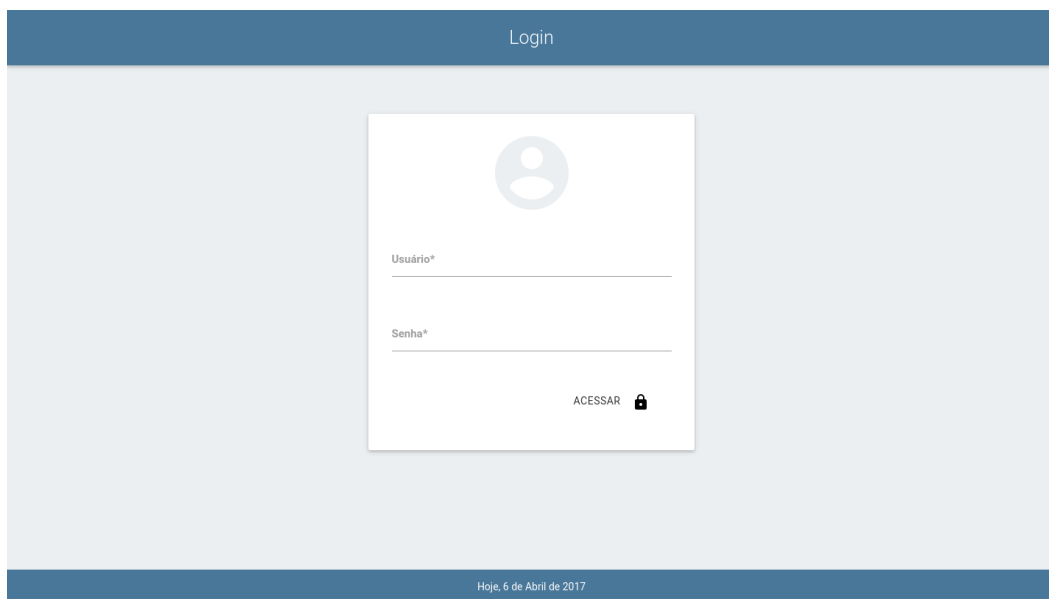


Figura 5.5: *Protótipo - Tela login*

Dessa forma, as chances de acertar e criar um produto de qualidade são muito maiores. Apenas é preciso ter em mente que o protótipo é uma pequena parte de algo maior e que ele deve durar apenas o tempo necessário para que aja um direcionamento do sistema. No fim das contas, quem trará faturamento para a sua empresa é o software, não o protótipo. Mesmo com as métricas bem definidas e com os resultados na mão, chega uma hora em que é preciso abandonar o protótipo e se dedicar apenas ao software final. Nesses casos, quando o protótipo não possui uma arquitetura que possa evoluir, ele é chamado de descartável. É possível também criar um protótipo do tipo evolutivo, onde o software vai nascendo à medida em que o protótipo evolui. Protótipos podem melhorar a qualidade de requisitos e especificações fornecidas aos desenvolvedores. Após a versão final do software, qualquer tipo de mudança poderá aumentar exponencialmente o custo do software, além de estourar os prazos definidos no início.

Nos últimos dez anos, tornou-se comum uma filosofia de prototipagem conhecida como “always beta”, ou sempre beta. Popular em empresas de internet, significa que o software será permanentemente um protótipo e que ele irá sendo melhorado de acordo com as necessidades do usuário.

Figura 5.6: Protótipo - Formulário de peças

Um software normalmente é reproduzido em cópias originais e seu código-fonte é único, porém tem complexidades suficientes para necessitar de protótipos antes de seu total desenvolvimento. Os protótipos de software são representações que descrevem todas as características funcionais do software, ou seja, aquelas que serão utilizadas no dia-a-dia, pelo usuário.[Rodrigues 2008]

Figura 5.7: Protótipo - Formulário cliente

O usuário do sistema não tem, nem haveria de ter, muito conhecimento técnico para entender a documentação da fase de análise e projeto, por isso, os protótipos de software são ferramentas úteis para uma comunicação clara com o usuário. Ao

protótipar estaremos permitindo que o usuário conheça o sistema antes mesmo de sua real codificação. [Rodrigues 2008]

As Figuras 5.5, 5.6 e 5.7 representam alguns protótipos do sistema.

5.4 Código

```

1  package site.fabricionogueira.androidclient.http;
2
3  import com.google.firebase.iid.FirebaseInstanceId;
4
5  import okhttp3.OkHttpClient;
6  import okhttp3.Request;
7  import okhttp3.logging.HttpLoggingInterceptor;
8  import retrofit2.Retrofit;
9  import retrofit2.converter.gson.GsonConverterFactory;
10 import site.fabricionogueira.androidclient.BuildConfig;
11
12 /**
13  * Gerenciador para as requisições ao serviço.
14  *
15  * @author Fabricio Nogueira <nogsantos@gmail.com>
16  * @since 19/03/2017
17  */
18 public class Service {
19
20     /**
21      * Cria um cabeçalho para as requisições
22      *
23      * @return OkHttpClient Cabeçalho construido
24      */
25     private static OkHttpClient header() {
26         OkHttpClient.Builder httpClient = new OkHttpClient.Builder();
27         /*
28          * Log
29          */
30         if (BuildConfig.LOG) {
31             HttpLoggingInterceptor logging = new HttpLoggingInterceptor();
32             logging.setLevel(HttpLoggingInterceptor.Level.BODY);
33             httpClient.addInterceptor(logging);
34         }
35         httpClient.addInterceptor(chain -> {
36             Request request = chain.request().newBuilder()
37                 .addHeader("Accept", "application/json; charset=utf-8")
38                 .addHeader("Authorization", FirebaseInstanceId.getInstance().getToken())
39                 .addHeader("App-authorization", BuildConfig.shared_name)
40                 .build();
41             return chain.proceed(request);
42         });
43         return httpClient.build();
44     }
45
46     /**
47      * Realiza a requisição
48      *
49      * @return Retrofit
50      */
51     public static Retrofit request() {
52         return new Retrofit.Builder()
53             .baseUrl(BuildConfig.service_address)
54             .addConverterFactory(GsonConverterFactory.create())
55             .addCallAdapterFactory(RxErrorHandlingCallAdapterFactory.create())
56             .client(header())
57             .build();
58     }
59
60 }

```

Código 5.1: Gerenciador das requisições

O Código 5.1 representa o modo de como as requisições devem ser realizadas pelo cliente mobile. A biblioteca Retrofit é responsável por gerenciá-las.

5.5 Frameworks e Ferramentas

Framework em computação pode ser definido como uma abstração na qual o software que fornece funcionalidade genérica pode ser alterado seletivamente por código adicional. Em sistemas de computador, uma estrutura é muitas vezes uma estrutura em camadas indicando que tipo de programas podem ou devem ser construídos e como eles se inter-relacionam. Algumas estruturas de sistema de computador também incluem programas reais, especificam interfaces de programação ou oferecem ferramentas de programação para usar os *frameworks*. Um *framework* pode ser para um conjunto de funções dentro de um sistema e como elas se inter-relacionam; As camadas de um sistema operacional; As camadas de um subsistema de aplicação; Como a comunicação deve ser padronizada em algum nível de uma rede; e assim por diante. Um quadro é geralmente mais abrangente do que um protocolo e mais prescritivo do que uma estrutura.

Ferramentas, para este projeto, podem ser definidas como programas de computador que os desenvolvedores de software usam para criar, depurar, manter ou suportar outros programas e aplicativos. O termo geralmente se refere a programas relativamente simples, que podem ser combinados juntos para realizar uma tarefa, tanto quanto se pode usar várias ferramentas manuais para corrigir um objeto físico. A capacidade de usar uma variedade de ferramentas de forma produtiva é uma marca de um bom engenheiro de software.

5.5.1 Práticas adotadas em todo o projeto

Versionamento de código com Git

O Git é um sistema de controle de versão distribuído gratuito e de código aberto projetado para lidar com projetos pequenos a muito grandes, com velocidade e eficiência [Software-Freedom-Conservancy].

Estratégia de branches para git e release de software

O git-flow é um conjunto de extensões para o git que provê operações de alto-nível para repositórios usando o modelo de branches do Vincent Driessen [Vincent].

Versionamento semântico

Um conjunto simples de regras e requisitos que ditam como os números das versões são atribuídos e incrementados [Versionamento-semantico].

5.5.2 Ferramentas Api

Visual Studio code

Visual Studio Code é um editor de código-fonte desenvolvido pela Microsoft para Windows, Linux e macOS. Ele inclui suporte para depuração, controle Git incorporado, realce de sintaxe, conclusão de código inteligente, trechos e refatoração de código [Microsoft].

Docker

Docker é a plataforma de contêiner de software líder mundial. Os desenvolvedores usam o Docker para eliminar problemas de "problemas com a minha máquina" ao colaborar em código com colegas de trabalho. Os operadores usam Docker para executar e gerenciar aplicativos lado a lado em contêineres isolados para obter uma melhor densidade de computação. As empresas usam o Docker para construir gasodutos de entrega de software ágeis para enviar novas funcionalidades de forma mais rápida, segura e confiável para os aplicativos Linux e Windows Server [Docker].

5.5.3 Frameworks Api

O código do *framework* utilizado na construção da *API* pode ser acessado em meu repositório no Github.

NodeJs

Node foi projetado para criar aplicativos de rede escaláveis [Joyent].

Nodal

Nodal é um servidor web para Node.js, otimizado para a criação de serviços *API* de forma rápida e eficiente. Porém, nesse para esse projeto, um *fork* do repositório foi criado e várias customizações realizadas [Keith], [Fabricio].

5.5.4 Ferramentas Web

Visual Studio Code 5.5.2.

5.5.5 Frameworks Web

Aurelia

Aurelia é um *framework* JavaScript que aproveita convenções simples e modernas, facilitando o desenvolvimento de soluções robustas e escaláveis em JavaScript [Rob].

Typescript

O TypeScript compila um código JavaScript simples e limpo que é executado em qualquer navegador, em Node.js ou em qualquer mecanismo que suporte JavaScript [Microsoft].

5.5.6 Ferramentas Mobile

Android Studio

O Android Studio oferece as ferramentas mais rápidas para a criação de aplicativos em todos os tipos de dispositivos Android [Google].

5.5.7 Frameworks Mobile

Firebase

Firebase é uma plataforma móvel que ajuda você a desenvolver rapidamente aplicativos de alta qualidade, aumentar sua base de usuários e ganhar mais dinheiro. Firebase é composto de recursos complementares que você pode misturar e combinar para atender às suas necessidades [Google].

Retrofit

Um cliente HTTP seguro para Android e Java [Square].

Glide

Uma biblioteca de armazenamento e de carregamento de imagens para Android centrada na rolagem suave [Bumptech].

RxAndroid

Tornar a escrita componentes reativos em aplicativos Android fácil e sem complicações [ReactiveX].

EventBus

O EventBus permite a comunicação central para classes desacopladas com apenas algumas linhas de código simplificando o código, removendo dependências e acelerando o desenvolvimento de aplicativos [GreenRobot].

Hellocharts

Biblioteca de gráficos para Android [Leszek].

Realm

O Realm Java permite que você escreva de forma eficiente a camada de modelo de seu aplicativo de forma segura, persistente e rápida [Realm].

5.5.8 Documentação

Astah Community

Astah Community é um software para modelagem UML. É desenvolvido na plataforma Java, o que garante sua portabilidade para qualquer plataforma que possui uma máquina virtual Java. Foi utilizado para a modelagem *UML* por ser uma ferramenta gratuita, e atender totalmente as necessidades do projeto ao desenvolver os diagramas *UML*.

Considerações Finais

6.1 Conclusão

A motivação para o desenvolvimento do projeto, além de servir como prática do assuntos estudados e abordados no curso de pós-graduação. Gera a possibilidade de se criar um produto que pode ser lançado no mercado com foco nos centros automotivos de pequeno e médio porte e profissionais liberais com um custo mais acessível para esse nicho. Visando sempre em primeiro lugar a melhor experiência para o usuário ao utilizar o sistema, a simplicidade nos fluxos do sistema, telas limpas e com poucas informações fecham a fase de estudos teóricos e colocam em prática tudo o que foi aprendido.

6.1.1 Contribuição

Neste trabalho foi possível desenvolver um sistema de controle de orçamentos e ordens de serviço para centros automotivos de pequenos e médio porte e profissionais liberais.

- Foi estudado o referencial teórico para o desenvolvimento de um software;
- Recolhido os requisitos funcionais e não funcionais;
- Foi realizado o projeto de Banco de Dados;
- Implementado um sistema de software denominado Orçamento WEB ;
- Foi criado o mapa mental do sistema
- Um app foi publicado na google play, que apoia o uso do cliente web pelo centro automotivo.

6.2 Trabalhos Futuros

Para trabalhos futuros, na versão 2.0.0 ficam:

1. Geração de relatórios;

2. Melhorias na usabilidade do cliente web e cliente mobile de acordo com o retorno dos usuários;
3. Implementação de grupos de acesso com diferentes perfis;
4. Integração com software de controle de estoque de produtos;
5. Integração com software de controle financeiro;
6. Melhorias na segurança do projeto habilitando ssl para as plataformas;

Referências Bibliográficas

- [Bezerra 2007]BEZERRA, E. *Princípios de análise e projeto de sistemas com UML*. [S.l.]: CAMPUS - RJ, 2007. ISBN 9788535216967.
- [Bumpteck]BUMPTECH. *Glide*. www.github.com/bumpteck/glide.
- [Date 1985]DATE, C. *Introdução a sistemas de bancos de dados*. [S.l.]: Campus, 1985. ISBN 9788535212730.
- [Docker]DOCKER. *Docker*. www.docker.com.
- [D'Souza 1998]D'SOUZA, D. *Interface Specification, Refinement, and Design with UML/Catalysis*. [S.l.: s.n.], 1998. 12-18 p.
- [Fabricio]FABRICIO, N. S. *Nodal Fork*. www.github.com/nogsantos/nodal.
- [Fielding]FIELDING, R. T. *Roy Thomas Fielding, Ph.D.* roy.gbiv.com/vita.html.
- [Google]GOOGLE. *Android Studio*. www.developer.android.com.
- [Google]GOOGLE. *Firebase*. www.firebase.google.com.
- [GreenRobot]GREENROBOT. *EventBus*. www.greenrobot.org/eventbus/.
- [ICS]ICS. *Representational State Transfer (REST)*. www.ics.uci.edu.
- [Joyent]JOYENT. *NodeJs*. www.nodejs.org/en/about/.
- [Keith]KEITH, H. *Nodal*. www.nodaljs.com.
- [Leszek]LESZEK, W. *HelloCharts*. www.github.com/lecho/hellocharts-android.
- [Microsoft]MICROSOFT. *TypeScript*. www.typescriptlang.org.
- [Microsoft]MICROSOFT. *Visual Studio Code*. www.code.visualstudio.com.
- [MONTEIRO 2014]MONTEIRO, J. *GOOGLE ANDROID - CRIE APLICAÇÕES PARA CELULARES E: TABLETS*. CASA DO CODIGO, 2014. ISBN 9788566250022. Disponível em: <<https://books.google.com.br/books?id=QFUdnQEACAAJ>>.

- [Preston-Werner]PRESTON-WERNER. *Tom Preston-Werner*. tom.preston-werner.com.
- [ReactiveX]REACTIVEX. *RxAndroid*. www.github.com/ReactiveX/RxAndroid.
- [Realm]REALM. *Realm*. www.realm.io/docs/java/latest/.
- [Rob]ROB, E. *Aurelia*. www.aurelia.io.
- [Rodrigues 2008]RODRIGUES, E. *Curso de Engenharia de Software*. Universo dos Livros Editora, 2008. ISBN 9788578730109. Disponível em: <<https://books.google.com.br/books?id=ZJznA9UrtVAC>>.
- [Silva, Freire e Callahan 2011]SILVA, C. T.; FREIRE, J.; CALLAHAN, S. P. *Provenance for Visualizations: Reproducibility and Beyond*. [S.l.: s.n.], 2011.
- [Silveira 2011]SILVEIRA, S. L. . P. S. . G. *Introdução à Arquitetura e Design de Software - Uma Visão Sobre a Plataforma Java*. São Paulo: ELSEVIER - CAMPUS, 2011.
- [Software-Freedom-Conservancy]SOFTWARE-FREEDOM-CONSERVANCY. *Git*. www.git-scm.com.
- [Square]SQUARE. *Retrofit*. www.square.github.io.
- [Versionamento-semantico]VERSIONAMENTO-SEMANTICO. *Versionamento Semântico 2.0.0*. www.semver.org.
- [Vincent]VINCENT, D. *Git branching model*. www.nvie.com.
- [Wiggins]WIGGINS, A. *The twelve-factor App*. 12factor.net.