UNIVERSIDADE DE CAXIAS DO SUL
ÁREA DE CIÊNCIAS EXATAS E ENGENHARIAS

GABRIEL ALVES BISOL

DESENVOLVIMENTO DE UMA SOLUÇÃO PARA TROCA DE DADOS ENTRE UM ERP E UM APLICATIVO MÓVEL

CAXIAS DO SUL
2019

UNIVERSIDADE DE CAXIAS DO SUL
ÁREA DE CIÊNCIA EXATAS E ENGENHARIA

GABRIEL ALVES BISOL

DESENVOLVIMENTO DE UMA SOLUÇÃO PARA TROCA DE DADOS ENTRE UM ERP E UM APLICATIVO MÓVEL

Trabalho de Conclusão de Curso para obtenção do Grau de Bacharel em Sistemas de Informação da Universidade de Caxias do Sul.

Orientadora: Prof.ª Dr.ª Helena Graziottin Ribeiro

CAXIAS DO SUL
2019

GABRIEL ALVES BISOL

DESENVOLVIMENTO DE UMA SOLUÇÃO PARA TROCA DE DADOS ENTRE UM ERP E UM APLICATIVO MÓVEL

Trabalho de Conclusão de Curso para obtenção do Grau de Bacharel em Sistemas de Informação da Universidade de Caxias do Sul.

Orientadora: Prof.ª Dr.ª Helena Graziottin Ribeiro

Aprovado em: 29/11/2019.

BANCA EXAMINADORA:

_____

Prof.ª Dr.ª Helena Graziottin Ribeiro – Orientadora

UCS – Universidade de Caxias do Sul

_____

Prof.ª Msc. Iraci Cristina Silveira de Carli

UCS – Universidade de Caxias do Sul

_____

Prof. Msc. Alexandre Erasmo Krohn Nascimento

UCS – Universidade de Caxias do Sul

Dedico este trabalho a todos que me acompanharam durante o decorrer da minha graduação, mas em especial a minha família que sempre esteve apta a me ajudar e dar conselhos.

AGRADECIMENTO

Nesta página quero agradecer a todos que me ajudaram, e auxiliaram neste trabalho de conclusão.

Agradeço a meus pais, Flávio e Rosenei, e minha namorada Brenda pelo incentivo e apoio que me foi dado em todos os momentos. Sem vocês, não teria alcançado o fim desta etapa tão importante da minha vida.

À minha orientadora, Professora Helena Graziottin Ribeiro, pelas orientações que me foram dadas, pois sem elas não seria possível concluir este projeto.

RESUMO

Nos negócios é necessário que as organizações gerem diferenciais competitivos para possuir destaque no mercado. Com as grandes proporções que os dispositivos móveis ocupam em nossa sociedade, eles passaram a ser uma ferramenta de suma importância dentro e fora das organizações. Através dos *smartphones* ou *tablets* é possível enviar e receber dados a qualquer momento, agilizando os processos e as tomadas de decisão nas organizações. A empresa Sartor Assessoria e Comércio de Informática gostaria de ter um diferencial em seu ERP, o MS2 Sistema de Gestão Clínico, e assim otimizar e agilizar o processo de coleta dos dados dos exames de refração. Atualmente o MS2 já inclui um local para armazenar os dados obtidos através do exame de refração. Para melhorar o processo de coleta dos dados do exame de refração esse trabalho apresenta a proposta de um aplicativo móvel desenvolvido para o sistema operacional *Android* e de um *web service* implementado em Java que disponibilizará serviços de troca de dados para realizar a integração entre o aplicativo móvel e o sistema MS2.

Palavras-chave: Aplicativo móvel. Web Service. Android. ERP.

ABSTRACT

In business, organizations need to generate competitive differentials to include market prominence. With the large proportions that mobile devices occupy in our society, they are a very important tool within a forum. Through smartphones or tablets, you can send and receive data at any time, streamlining processes and decision making in organizations. The company Sartor Assessoria e Comercio de Informatica wishes to have a differential in its ERP, the MS2 Clinical Management System and, thus, optimize and streamline the process of collecting refractive exam data. Currently, MS2 already includes a place to store the data used through the refractive exam. To improve the process of data collection from the refractive exam, this paper presents a mobile application proposal developed for the Android operating system and a web service implemented in Java that provides data exchange services to integrate the mobile application. and the MS2 system.

Keywords: Mobile application. Web service. *Android*. ERP.

LISTA DE FIGURAS

LISTA DE QUADROS

LISTA DE ABREVIATURAS E SIGLAS

ADT	*Android Development Tools*
API	*Application Programming Interface*
APK	*Android Package File*
CSS	*Cascade Style Sheet*
EDI	*Enterprise Data Interchange*
ERP	*Enterprise Resources Planning*
HMR	*Hot Module Replacement*
HTML	*Hyper Text Markup Language*
HTTP	*Hyper Text Transport Protocol*
IDC	*International Data Corporation*
JSON	*JavaScript Object Notation*
MOM	*Middleware* Orientado à Mensagem
MRP	*Material Requirements Planning*
REST	*Representational State Transfer*
RPC	*Remote Procedure Call*
SDK	*Software Development Kit*
SI	Sistema de Informação
SO	Sistema Operacional
SOAP	*Simple Object Access Protocol*
SQL	*Structured Query Language*
TCP	*Transmission Control Protocol*
TDD	Desenvolvimento Orientado a Testes
TI	Tecnologia da Informação
UDDI	*Universal Description, Discovery and Integration*
URI	*Uniform Resource Identifier*
URL	*Uniform Resource Locator*
WSDL	*Web Service Definition Language*
XML	*Extensible Markup Language*

SUMÁRIO

1 INTRODUÇÃO

O trabalho aborda o desenvolvimento de um protótipo de um aplicativo móvel que suportará a coleta dos dados de um exame clínico de refração que se comunicará com o *Enterprise Resources Planning* (ERP) MS2 através de um *web service*. Dois motivos despertam para a relevância deste tema. Um dos fatores o qual aguçou a atenção para o assunto foi que Tecnologia da Informação (TI) passou a estar cada vez mais presente em nosso cotidiano juntamente com os sistemas informatizados. Assim se faz necessário uma busca constante de soluções cada vez mais rápidas e eficazes para os usuários. Eis que surge o segundo fator importante considerado nesta pesquisa pelo acadêmico: o crescente uso dos aplicativos móveis. Os aplicativos atualmente no mercado são de fácil e acesso, e estão presentes quase que diariamente na vida da população mundial, sendo utilizados para os mais variados fins.

A evolução da infraestrutura da TI (LAUDON, 2010) e a maior facilidade no acesso à Internet proporciona aos usuários acesso aos mais diversos sistemas de informação, como os ERPs.

O contexto desta pesquisa se aplica dentro da empresa Sartor Assessoria e Comércio de Informática LTDa que atualmente já trabalha com o sistema ERP. Com este *software* a empresa abrange uma diversificada gama de clientes. Dentre eles estão: indústrias, comércios, serviços, setor agrícola e clínicas médicas.

Ao observar o dia-a-dia da empresa, constatou-se a necessidade de um determinado cliente do ramo de clínicas oftalmológicas por um sistema que lhe permitisse mais mobilidade interna. Hoje a empresa Sartor disponibiliza a este cliente e aos demais do ramo um sistema denominado “MS2 Sistema de Gestão Clínico”, que possui inúmeras funcionalidades de inserção de dados. Desde as mais básicas, tais como: cadastro de paciente e organização de agendas diárias/semanais dos médicos, até as inserções mais complexas como: prontuário eletrônico e resultado dos exames de refração. Porém as inserções dos dados do exame de refração requerem a utilização do sistema instalado em uma sala distante de onde é realizado o exame, sem a mobilidade necessária que o cliente requer. Visto que o equipamento para realização do exame de refração se localiza em uma parte isolada da clínica, pelo fato de não poder sofrer interferências eletrônicas, pois esta interferência pode ocasionar uma possível alteração no resultado final do exame.

Verificou-se assim uma necessidade de tornar esse processo mais fácil e ágil, já que após a realização do exame é preciso retirar o cupom impresso com resultado da máquina, ir até o computador e lançar os dados no sistema.

Assim considerou-se estender a tecnologia relacionada aos ERPs (CAIÇARA, 2015) já utilizada pela empresa Sartor a uma solução prática: um aplicativo móvel.

Pensou-se em uma solução através de um aplicativo móvel, pois são sistemas leves, portáteis, de fácil uso, acessíveis pela *web* entre outras características (El-Kassas et al., 2015) apresentando-se assim como uma solução para o atual problema de mobilidade encontrado no cliente da empresa Sartor Assessoria e Comércio de Informática LTDa.

Uma forma bastante utilizada para conectar dispositivos móveis aos sistemas de informação são os *web services* (SADAGI, 2013), aos quais possibilitam a comunicação entre as aplicações através da internet, esta comunicação é realizada utilizando os protocolos padrão da internet (www.w3.og).

1.1 PROBLEMA DE PESQUISA

Sendo assim, diante desses fatores anteriormente apresentados, a presente pesquisa diz respeito ao desenvolvimento de um protótipo de um aplicativo móvel que suportará a coleta dos dados de um exame clínico de refração e que se comunicará com o ERP MS2 através de um *web service*, tendo como ponto de partida o seguinte problema: O ERP MS2 permite a integração com um protótipo de um aplicativo móvel, capaz de suportar a coleta de dados de um exame clínico de refração, através de um *web service*?

1.2 OBJETIVOS

Para que o problema mencionado acima fosse investigado, foi necessário elaborar o seguinte objetivo geral: Desenvolver um protótipo de um aplicativo móvel que suportará a coleta dos dados de um exame clínico de refração que se comunicará com o ERP MS2 através de um *web service*, seguido dos seguintes objetivos específicos:

a) elaborar um *web service* capaz de viabilizar a comunicação com o protótipo do aplicativo móvel, através dos serviços necessários;
b) planear um protótipo de aplicativo móvel, apropriado para a coleta de dados dos exames clínicos de refração, capaz de comunicar-se com o ERP através do *web service* desenvolvido;

1.3 METODOLOGIA DO TRABALHO

Sendo assim, nesta pesquisa a metodologia adotada pelo pesquisador para responder suas inquietações se fundamenta em uma pesquisa qualitativa. A pesquisa qualitativa não tem pretensão de generalização do assunto, ao contrário, ela lida com assuntos subjetivos e particulares, ou seja, um cunho localizador. Assim como Fonseca (2002, p. 2) defende: "a pesquisa qualitativa preocupa-se, portanto, com aspectos da realidade que não podem ser quantificados, centrando-se na compreensão e explicação da dinâmica das relações sociais".

Portanto a abordagem desta pesquisa se corrobora à pesquisa qualitativa, pois seu problema de pesquisa se volta a compreensão e explicação de um possível encontro entre um software de uma única empresa com o sistema de *web service*. Além disto, pode se dizer de que esta pesquisa é qualitativa de cunho bibliográfico, pois contempla o estudo de material bibliográfico para formação do conhecimento: quais as tecnologias disponíveis para desenvolvimento de aplicativos móveis, e quais as interfaces possíveis do ERP que armazena os dados. É bibliográfica pois, ainda segundo Fonseca (2002, p. 32):

> A pesquisa bibliográfica é feita a partir do levantamento de referências teóricas já analisadas, e publicadas por meios escritos e eletrônicos, como livros, artigos científicos, páginas de web sites. Qualquer trabalho científico inicia-se com uma pesquisa bibliográfica, que permite ao pesquisador conhecer o que já se estudou sobre o assunto [...].

Na primeira etapa deste trabalho será realizado o estudos sobre os SI com foco no sistema ERP, a análise das plataformas disponíveis para desenvolvimento de aplicativos móveis, tendo em vista que elas devem suprir ou ajudar a implementar as demandas do projeto. Assim, vai ser realizada uma pesquisa para descobrir bibliotecas úteis na implementação.

Será realizada então a análise das soluções tecnológicas disponíveis para a integração de dados entre diferentes plataformas. Após será realizado a modelagem do sistema com base no levantamento de requisitos, contendo caso de uso, modelagem do banco de dados, e diagrama de sequência.

A seguir será feita a implementação de um *web service* para realizar a integração dos dados do sistema MS2 com um aplicativo móvel, e depois será implementado um aplicativo móvel na plataforma *Android*, para que o usuário realize o gerenciamento dos exames de refração através de *tablets*. Para que essa implementação possa ser realizada será utilizada a

ferramenta *Android Studio*, com a linguagem *Java* e com o banco de dados *SQLite*. Ao final o aplicativo móvel e o *web service* serão disponibilizados para os testes necessários, com a apresentação dos resultados obtidos reforçando suas vantagens.

1.4 ESTRUTURA DO TRABALHO

Este trabalho está dividido em 7 capítulos, além do capítulo que trata a introdução, os quais contemplam os seguintes temas:

Cap. 2 - A evolução dos sistemas e os tipos de sistemas, além de conceituar os sistemas de gestão.

Cap. 3 - Aplicativos móveis, detalhando o desenvolvimento dos aplicativos móveis e suas plataformas, o vínculo entre sistemas corporativos e os aplicativos móveis, além destacar alguns trabalhos com temas relacionados.

Cap. 4 - Integração entre sistemas, trazendo os conceitos e necessidades de integração nas organizações e as principais formas de integração entre sistemas e aplicativos.

Cap. 5 - Apresentação da proposta do aplicativo móvel e do *web service*, onde são detalhadas as funcionalidades do software MS2 Sistema de Gestão Clínico, além dos requisitos funcionais e não-funcionais propostos.

Cap. 6 - Apresentação das ferramentas e métodos utilizados no desenvolvimento do aplicativo móvel e do *web service*, como *software* e *hardware* e suas funcionalidades.

Cap. 7 - Apresentação das conclusões finais e trabalhos futuros.

2 SISTEMAS DE INFORMAÇÃO

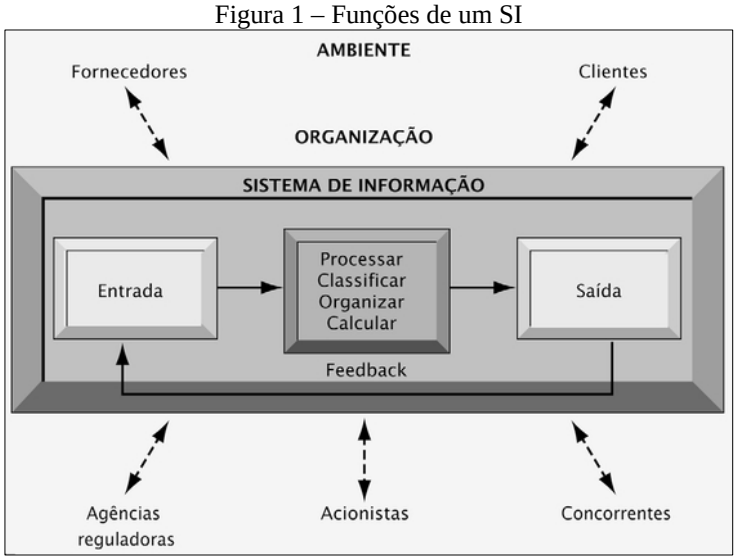
Os Sistemas de Informação (SI) e as tecnologias que surgem, passam a ser aliados das organizações para criação de novos produtos e serviços, ajudando a atingir grandes níveis de eficiência e competitividade. A organização passa a produzir, entregar e vender seus produtos e serviços, além da interação com fornecedores e clientes, o que gera um modelo de negócio.

Sendo assim, o SI é um conjunto de componentes relacionados entre si, para coletar, processar, armazenar e distribuir informações, auxiliando os gestores na tomada de decisões e no controle organizacional (LAUDON, 2010). Para geração de resultados realmente oportunos para as organizações existem três atividades em um SI:

a) entrada: coleta os dados.
b) processamento: converte os dados.
c) saída: transfere a informação.

O *feedback* é a resposta para determinadas áreas ou pessoas da organização, para avaliar e verificar se porventura é necessário ajustar a atividade de entrada do processo descrito anteriormente. A figura 1 apresenta o fluxo dessas atividades descritas.

Figura 1 – Funções de um SI



Fonte: (LAUDON, 2010, p. 10).

Conforme Laudon (2010), a amplitude que os SI podem alcançar vai muito além de apenas tecnologia e computadores. O SI possui três dimensões: organizacional, pessoal e TI.

A dimensão organizacional compreende processos e abordagem de gestão, e vale salientar que cada empresa tem uma cultura peculiar, uma série de princípios, valores e modos

de fazer as ações. As empresas possuem uma hierarquia na sua formação e os processos organizacionais coordenam a execução desse trabalho na hierarquia, e o SI passa a automatizar esses processos organizacionais. Por isso é preciso que SI esteja em conformidade com as ações efetuadas pela organização, pois caso não esteja pode acarretar com o fracasso da mesma.

A dimensão pessoal envolve as pessoas que utilizam os sistemas juntamente com quem o desenvolve. O sucesso que ocorre em uma organização está diretamente relacionado à capacidade técnica que seus colaboradores exercem nela, o mesmo se aplica ao SI, é improdutivo sem que existam pessoas capacitadas para desenvolvê-los e mantê-los.

E por fim a terceira dimensão, que representa a tecnologia em si a sua infraestrutura, com quatro recursos tecnológicos:

a) *hardware:* é o equipamento de entrada e saída e armazenamento, ou seja, computadores e outros equipamentos de vários tipos e diferentes desempenhos.
b) *software:* são os programas, através dos quais o hardware é controlado, através de instruções pré-programadas.
c) tecnologia de comunicação e rede*:* engloba *software* e *hardware* para a transferência de dados através de sua *intranet* e da *extranet* (ou *internet*).
d) *world wide web:* age como um integrador de informações, as quais estão disponíveis na internet.

A junção dessas tecnologias aliadas às pessoas que participam, gera o que chamamos de Infraestrutura da Tecnologia da Informação (LAUDON, 2010).

A figura 2 apresenta as três dimensões que representam um SI.

Figura 2 – Dimensões de um SI

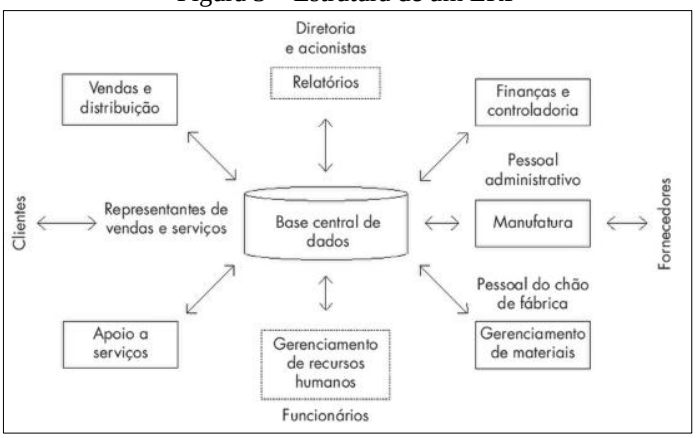
Fonte: (LAUDON, 2010, p. 11).

2.1 SISTEMA ERP

O ERP é a transformação do *Material Requirements Planning* (MRP), que deixou de atender a necessidade de gerenciamento empresarial, em meados de 1990 (CAIÇARA, 2015). Assim Caiçara determina o ERP como "[...] um sistema de informação adquirido na forma de pacotes comerciais de software que permitem a integração de dados dos sistemas de informação e de processos de uma organização" (CAIÇARA, 2015, p. 96-97).

Neste contexto, o ERP assume o papel de um sistema de informação gerencial, que foi criado para unir os departamentos de uma organização. Através dele, os processos de uma empresa passam a ser automatizados e integrados com a utilização de um único sistema. Desta forma, os processos da organização passam a ser contínuos e consistentes utilizando o mesmo sistema e um mesmo ambiente. A figura 3 apresenta uma estrutura simples de um sistema ERP, que possui uma base de dados central, onde todas as demais áreas estão interligadas à base central.

Figura 3 – Estrutura de um ERP



Fonte: (DAVENPORT, 1999 apud CAIÇARA, 2015, p. 98).

A arquitetura que é utilizada em um sistema de ERP é o modelo cliente-servidor e disposta em três camadas: camada de apresentação, camada de aplicação e camada de base de dados.

Na decisão de escolher um sistema de ERP as organizações buscam ganhar inúmeros benefícios, tais como: customização, redução de custos, integração com outros sistemas e a informação em tempo real que auxilia na escolha das decisões. Segundo Zwicker e Souza (2003), apesar de os ERP trazerem benefícios, existem problemas que devem ser levados em consideração. No que se refere a esse assunto são apresentadas pelos autores, através do quadro 1, as características desse tipo de sistema, comparando seus benefícios e problemas.

Quadro 1 – Benefícios e problemas dos ERP

Característica	Benefícios	Problemas
São pacotes comerciais	- redução de custos de informática; - foco na atividade principal da empresa; - redução do *backlog* de aplicações atualizações tecnológicas permanente, por conta do fornecedor.	- dependência do fornecedor; - empresa não detém o conhecimento sobre o pacote.
Usam modelos de processos	- difunde conhecimento sobre *best practices;* - facilita a reengenharia de processos; - impõe padrões.	- necessidade de adequação do pacote à empresa; - necessidade de alterar processos empresariais; - alimenta a resistência à mudança.
São sistemas integrados	- redução do retrabalho e inconsistências; - redução da mão-de-obra relacionada a processos de integração de dados; - maior controle sobre a operação da empresa; - eliminação de interfaces entre sistemas isolados; - melhoria na qualidade da informação; - contribuição para a gestão integrada; - otimização global dos processos da empresa.	- mudança cultural da visão departamental para a de processos; - maior complexidade de gestão da implementação; - maior dificuldade na atualização do sistema pois exige acordo entre vários departamentos; - um módulo não disponível pode interromper o funcionamento dos demais; - alimenta a resistência à mudança.
Usam bancos de dados corporativos	- padronização de informações e conceitos; - eliminação de discrepância entre informações de diferentes departamentos; - melhoria na qualidade da informação; - acesso à informação para toda a empresa.	- mudança cultural da visão de "dono da informação" para a de "responsável pela informação"; - mudança cultural para uma visão de disseminação de informações dos departamentos por toda a empresa; - alimenta resistência à mudança.
Possuem grande abrangência funcional	- eliminação da manutenção de múltiplos sistemas; - padronização de procedimentos; - redução de custos de treinamento; - interação com um único fornecedor.	- dependência de um único fornecedor; - se o sistema falhar toda a empresa pode parar.

Fonte: (ZWICKER; SOUZA, 2003).

Os fatores que interferem e podem ser considerados críticos ao implantar um ERP são a gerência do projeto considerar como crítico o compromisso da direção da organização, além é claro do comprometimento dos usuários e gerentes na busca por resultados satisfatórios. A não ocorrência desses fatores dificulta a implantação do ERP. Integrar os processos que são realizados internamente com os processos externos é o maior desafio para as organizações (SILVA, 2004).

3 APLICATIVOS MÓVEIS

Segundo LEE (2005) para um aplicativo ser considerado móvel ele deve conter características específicas, sendo que cada uma dessas características tem um significado próprio, essas características podem variar entre os aplicativos. Podem ser definidas como características de um aplicativo móvel:

a) portabilidade: conforme o autor a definição do termo é a capacidade de um aplicativo ser transportado. Alguns fatores podem interrompem a portabilidade de um aplicativo móvel: o seu tamanho e o seu peso;
b) usabilidade: um aplicativo móvel se torna usual quando ele pode ser acessado em diversos ambientes e por N pessoas. Essa funcionalidade está conectada à as características das pessoas que utilizam os aplicativos, tais características pertencem aos próprios aplicativos e passam a alterar a usabilidade num todo;
c) funcionalidade: um aplicativo móvel pode compreender inúmeras aplicações no momento em que está sendo executado;
d) conectividade: segundo o autor a troca de informações e interligar pessoas e sistemas é a primeira função de um aplicativo móvel.

Para El-Kassas et al. (2015) o desenvolvimento de um aplicativo móvel é diferente do que desenvolver um outro *software*. Ao se desenvolver um aplicativo móvel deve-se levar em consideração as capacidades que o mesmo deve ter como sua portabilidade, usabilidade, funcionalidade e conectividade. Deve-se levar em consideração algumas particularidades para desenvolver um aplicativo móvel:

a) limitações de recursos: seu processamento e armazenamento são limitados e a sua conectividade pode oscilar;
b) sistema heterogêneo: por possuírem diferentes sistemas operacionais (SO) e diversas configurações, ao desenvolver um aplicativo talvez seja preciso diferentes versões para um mesmo aplicativo;
c) manutenção: é necessário realizar frequentes atualizações e revisões nos aplicativos, pelo fato de as plataformas de aplicativos móveis sofrerem frequentes atualizações.

Para desenvolver um aplicativo móvel pode-se optar por plataformas de desenvolvimento nativo ou ambientes multiplataforma (plataformas híbridas).

3.1 DESENVOLVIMENTO NATIVO

Uma plataforma de desenvolvimento nativo possui seus próprios *frameworks* e seus *Software Development Kit* (SDK), sendo que os aplicativos passam a estar vinculados desde sua criação a essa plataforma, sendo projetados e executados apenas nela. Havendo a necessidade de se ajustar em diferentes plataformas o aplicativo precisa ser desenvolvido em separado para cada plataforma (EL-KASSAS et al., 2015).

Ao utilizar uma aplicação nativa o acesso entre o aplicativo e a plataforma torna-se mais simples, devido às *Application Programming Interface* (APIs) que plataforma nativa disponibiliza. Sendo assim alguns recursos que podem ser utilizados no aplicativo como câmeras, acesso a contatos, dentre outros, tem seu acesso facilitado (EL-KASSAS et al., 2015).

Neste trabalho serão descritas apenas duas plataformas nativas: a *Android* e iOs. O fator que levou a escolha dessas duas plataformas é que segundo a *International Data Corporation* (IDC) o mercado de *smartphones* no ano de 2018 teve um crescimento de 9,8%, e a plataforma *Android* dominou o mercado com 85,1% das quotas em comparação com o sistema iOS, tendo projeções para aumentar seu mercado em 2019 para 85,9%, colocando assim a plataforma *Android* como a mais utilizada no mundo, conforme o quadro 2.

Quadro 2 – Mercado de smartphones

Ano	**2018**	**2019**	**2020**	**2021**	**2022**	**2023**
Android	85,1%	85,9%	86,1%	86,4%	86,6%	86,7%
iOS	14,9%	14,1%	13,9%	13,6%	13,4%	13,3%

Fonte: IDC, Maio de 2019.

3.1.1 Android

O *Android* foi criado e desenvolvido pela empresa *Android* Inc. no ano de 2003, e em 2005 foi comprado pela empresa Google (PAPAJORGJI, 2015).

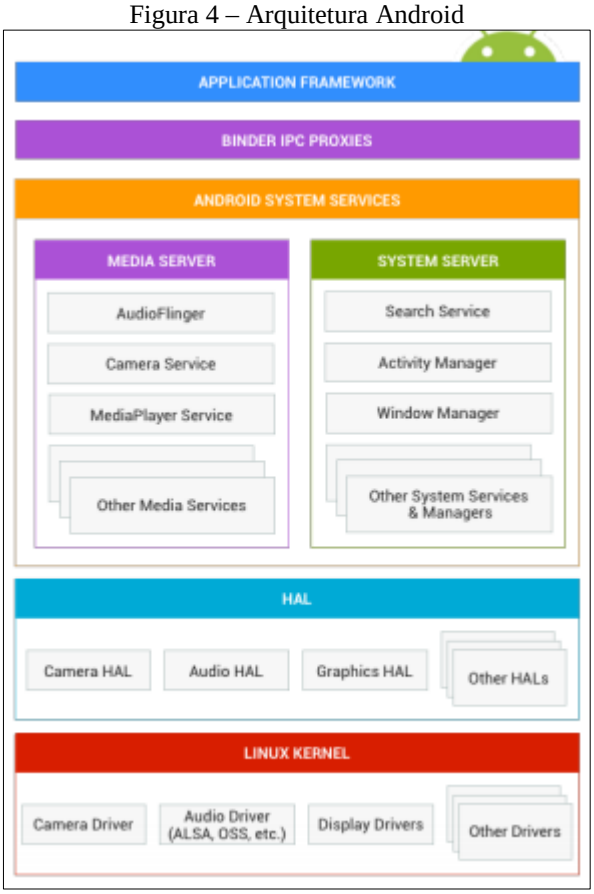
Ele opera através de versões, tendo cada nomenclatura de sua versão vinculada a um nome de um doce, como por exemplo a sua versão atual, que é denominada Pie que foi lançada em 2019. Na versão Pie dentro de todas as suas funcionalidades podemos destacar maior velocidade ao ligar o aparelho e o *picture in picture* que proporciona a utilização de duas funções ao mesmo tempo. Esse Sistema Operacional (SO) possui bibliotecas que

possibilitam aos desenvolvedores utilizarem um SDK que contém APIs para testar os aplicativos desenvolvidos. Existe ainda uma loja virtual chamada Google Play que é onde estão expostos os aplicativos próprios da organização e de terceiros (GOOGLE, 2019a).

3.1.1.1 Arquitetura Android

A figura 4 representa a arquitetura utilizada pela plataforma *Android*, que está definida em cinco camadas, englobando desde suas aplicações básicas até o seu *hardware*.

Figura 4 – Arquitetura Android



Fonte: (*ANDROID*, 2019a).

a) a primeira camada é a *Application Framework*, onde a aplicação é instalada (*ANDROID*, 2019a);
b) a camada *Binder IPC Proxies*, é responsável por interligar as aplicações que estão instaladas na primeira camada com terceira camada. É aqui que as APIs se comunicam com os serviços do sistema que residem na terceira camada (*ANDROID*, 2019a);

c) a terceira camada é a *Android System Services*, é nesta camada que se obtém acesso a *hardware* específicos. Existem duas classes nessa camada a *Media Server* e *System Server* (*ANDROID*, 2019a);
   - *media server:* aqui é onde encontram-se os serviços de mídia, como reprodução de áudio e vídeo;
   - *system server:* aqui estão os demais serviços do sistema, como notificações.

d) na camada *Hal*, é onde os fornecedores podem criar os *drivers* para realizar a integração com o *hardware* (*ANDROID*, 2019a).

e) e, por fim, a camada é a *Linux Kernel*, aqui estão as funcionalidades de gerenciamento de memória e funcionalidades para aplicativos embarcados (*Android*, 2019a).

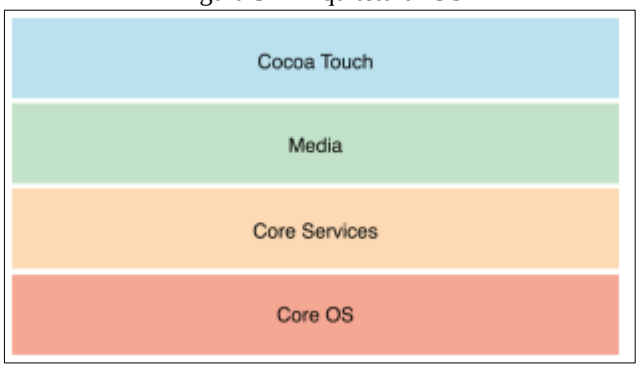
3.1.2 iOS

Todos os dispositivos móveis como iPhone e iPad dentre outros, que são da marca *Apple*, contém o mesmo SO que está na sua décima segunda geração. A família iOS além de possuir integração total entre o seu *hardware* e o seu SO possui outras qualidades, como por exemplo: processadores *dual core* e chip gráficos de altíssima velocidade. Existem APIs para cada dispositivo da família iOS, assim os desenvolvedores podem tirar o máximo proveito dessa tecnologia. A loja virtual é chamada de *App Store*, lá encontram-se aplicativos próprios e de terceiros. Antes da publicação de qualquer aplicativo a *Apple* verifica se aquele aplicativo possui algum vírus ou qualquer outra informação que não esteja de acordo com suas normas, para aí sim realizar publicação na sua loja virtual (APPLE, 2019).

3.1.2.1 Arquitetura iOS

A arquitetura iOS, que está ilustrada na figura 5, é estruturada em quatro camadas listadas que se apresentam na seguinte ordem: *Cocoa Touch, Media, Core Services e Core OS*.

Figura 5 – Arquitetura iOS



Fonte: (APPLE INC., 2019).

a) a primeira camada é a *Cocoa Touch* é onde estão situados os serviços que interagem com o usuário, ou seja, as tecnologias que melhoram a vivência dos usuários com multitarefas (APPLE INC., 2019);
b) a camada *Media* é a que contém os *frameworks* de multimídias como áudio, vídeo (APPLE INC., 2019);
c) na terceira camada, *Core Services*, são definidas as funcionalidades que interagem com o *hardware*, são as funcionalidades de baixo nível (APPLE INC., 2019);
d) e a última camada é a *Core OS*, na qual são criadas as funcionalidades de baixo nível aquelas que se aproximam mais da linguagem de máquina (APPLE INC., 2019).

3.2 DESENVOLVIMENTO HÍBRIDO

Utilizar o desenvolvimento em multiplataformas faz com que o aplicativo possa ter seu processamento e implantação em diversas plataformas. Criar uma página *web* e um *app* requer identificar o que torna a experiência do usuário amigável, o seu *front-end*. Para desenvolvimento multiplataforma, a interface que é apresentada aos usuários utiliza a linguagem *HiperText Markup Language* (HTML), *Cascade Style Sheet* (CSS) e *JavaScript* (EL-KASSAS et al., 2015). Segundo Heitkotter, Hanschke e Majchrzak (2013) os aplicativos desenvolvidos com esses recursos são definidos como *webapps*.

Para Juntunen (2013), HTML é uma linguagem de programação própria e muitas vezes é acionada para estender a sua utilização junto com CSS e *JavaScript*. Isso só é possível pois o HTML faz a definição do que será implantado, o CSS alinha a formatação e o *JavaScript* realiza a administração perante as tecnologias.

Os *webapps* são voltados para o desenvolvimento *web*, e utilizam as tecnologias mais recentes do HTML e CSS (HTML5 e CSS3) juntamente com o *JavaScript*, além de particularidades específicas a aplicativos móveis, como dimensões da tela. Esses *webapps* não ficam instalados no aplicativo e nem em suas lojas, eles são acessados pelo *browser*. Seu aspecto e funcionalidade são semelhantes a uma página *web*, o que os torna diferentes são os elementos que compõe a tela (HEITKOTTER; HANSCHKE; MAJCHRZAK, 2013).

A seguir são apresentadas duas ferramentas para desenvolvimento em multiplataformas. Essas ferramentas foram escolhidas por possuírem seu código fonte aberto, o que facilita o desenvolvimento e entendimento do código uma vez que o mesmo não é uma caixa preta. Por este motivo essas ferramentas são denominadas *open source*.

3.2.1 React Native

O *React* é uma biblioteca *open source* que utiliza *JavaScript* para criar a interfaces aos usuários. O engenheiro de *software* da empresa *Facebook* que desenvolveu esse biblioteca foi Jordan Walker. Na sua criação foi desenvolvida uma extensão do *JavaScript* denominada JSX, que tem como principal base o XHP, o qual é uma extensão da ferramenta PHP. O compilador que o *React* utiliza é denominado Babel, o qual tem suporte a essa extensão JSX. O *React Native* é um *framework* lançado pela empresa *Facebook*, que permite a criação de aplicativos híbridos (Facebook, 2019a).

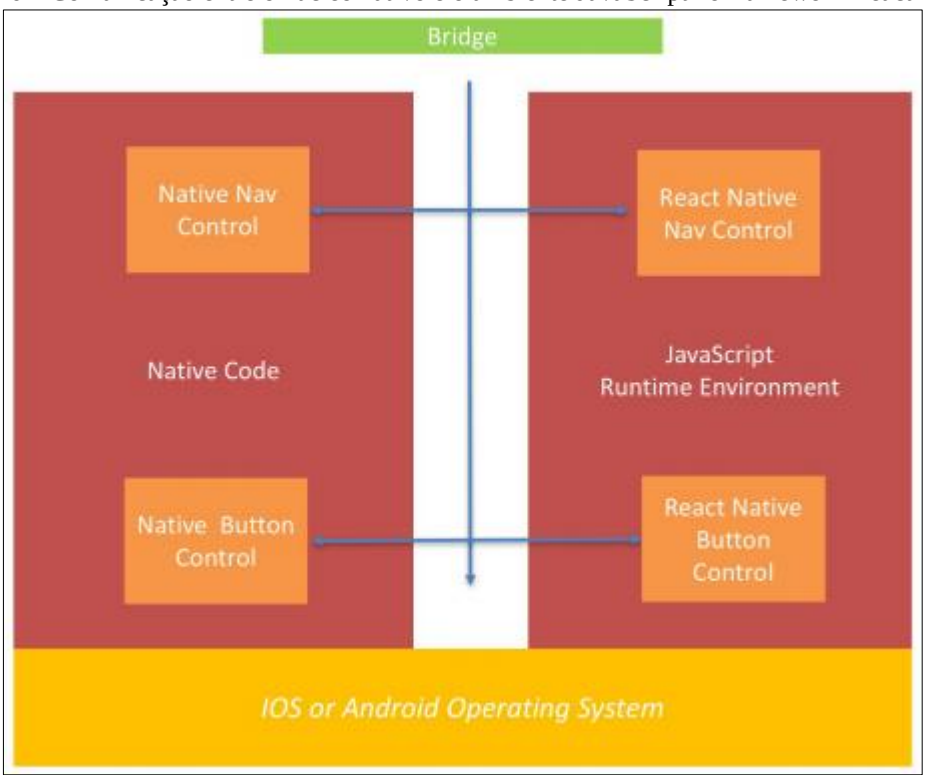
Outro fator importante é a capacidade de realizar a re-compilação da aplicação imediatamente. Essa funcionalidade é denominada *Hot Reloading*, que tem como objetivo realizar e injetar versões novas de arquivos na aplicação enquanto a mesma é executada, mantendo o seu estado. Para que isso seja possível o *React* utiliza o *Hot Module Replacement* (HMR) que verifica as alterações dos arquivos (Facebook, 2019b).

3.2.1.1 Arquitetura React Native

O *React* e o *JavaScript* são as principais linguagens do *React Native*. O *JavaScript* não é uma linguagem nativa dos *smartphones* e *tablets*, é preciso então um *bridge* que realiza a comunicação entre o processador dos dispositivos e essa linguagem. Neste *framework* existem dois núcleos: o nativo e o de execução do *JavaScript*, porém não existe apenas uma única ponte para realizar essa comunicação. Assim cada "*Control*" do *React Native* irá se conectar a um "*Control*" do *JavaScript* semelhante, conforme podemos observar na figura 6. Essa ponte

é determinante para realizar a ligação entre os dois núcleos, porém ela pode gerar atrasos no processamento (HEARD, 2019).

Figura 6 – Comunicação entre o núcleo nativo e o ambiente JavaScript no framework React Native



Fonte: (HEARD, 2019).

3.2.2 Flutter

O *Flutter* é um *framework* criado pela Google, para criação de aplicações de alta performance e lealdade às plataformas *Android* e iOS. A linguagem de programação que ele utiliza é a Dart, que também foi desenvolvida pela própria Google, a qual é reconhecida por ser clara, rápida e compatível com as multiplataformas. A funcionalidade *Hot Reloading*, que proporciona suporte a qualquer mudança no código repercutindo diretamente no estado da aplicação também está presente no *framework Flutter*. O Material, que é um sistema de design deste *framework*, auxilia no processo de desenvolvimento de aplicações por oferecer componentes simples de estilizar (GOOGLE, 2019b).

3.2.2.1 Arquitetura Flutter

A arquitetura do *framework Flutter* é disposta em camadas, como pode ser observado na figura 7, sendo cada uma delas as responsáveis por cada funcionalidade em uma aplicação.

Algumas destas camadas possuem sua nomenclatura autoexplicativas, como o caso das camadas "*Animation*" que realiza as animações nos *apps* e a "*Gesture*" que interpreta os gestos dos usuários. Já algumas outras não são tão claras como é o caso da camada "*Cupertino*" que possui *widgets* de alta fidelidade (GOOGLE, 2019c).

Figura 7 – Arquitetura Flutter

Fonte: (GOOGLE, 2019c).

3.3 APLICATIVOS MÓVEIS E SISTEMAS CORPORATIVOS

As organizações com o passar do tempo começaram a utilizar sistemas e aplicativos que são completamente distintos uns dos outros. Esse aumento na utilização dos diversos sistemas fez surgir um grande desafio: como integrar os diferentes tipos de sistemas e aplicações (DE SORDI; MARINHO, 2007).

Realizar tal integração é uma tarefa delicada, muitas vezes essa integração utiliza uma solução inadequada, o que passa a acarretar um enorme problema para a organização, ou até mesmo a falta de integração. Abaixo estão alguns problemas relatados na integração de sistemas (CUMMINS, 2002 Apud DE SORDI; MARINHO, 2007):

a) perda de competitividade pelo fato de não diminuir o tempo de processos que surgem da consequência da deficiência da integração imposta;
b) restrição da capacidade de firmar grandes parcerias, pelo fato de não se adequar a integração de informações, que são exigidas pelos seus parceiros;
c) aumento dos custos e os riscos pela reescrita de informações de uma base de dados para outra;

d) lentidão na organização ao detectar um problema pelo atraso na informação e ou pela dúvida na mesma, restringindo assim a evolução da organização;
e) dificuldade de evolução nos processos, referente ao fato de não aceitar mudanças no funcionamento, por receio de enfrentar problemas.

Com isto as organizações passaram a enxergar a integração com o SI, através de uma nova perspectiva, observaram que a integração se tornou um diferencial competitivo. Segundo (DE SORDI; MARINHO, 2007) a constante evolução tecnológica, por vezes torna arcaicas algumas formas de integração. É preciso que alguns processos sejam criados pelas organizações para que tornem a integração entre os sistemas de SI mais eficaz, alguns desses processos seriam:

a) aumento dos tipos de software dentro da organização;
b) órgãos reguladores na busca por maior rapidez no caminho da informação;
c) aumento da troca de informações com entidades organizacionais.

Pode-se definir essa integração entre aplicativos móveis e sistemas corporativos com o termo "mobilidade corporativa"[1], pois os aplicativos móveis tornaram-se uma extensão dos sistemas corporativos que estão presentes nas organizações. Essa mobilidade criada com a utilização dos aplicativos gerou mais liberdade aos colaboradores da organização, pelo fato de que as ações podem ser realizadas de qualquer lugar e a qualquer momento. Entre algumas vantagens da utilização de aplicativos móveis no ambiente corporativo estão: otimização de tempo, aumento dos lucros da empresa e maior gerenciamento dos processos.

3.4 TRABALHOS RELACIONADOS

No decorrer da pesquisa, foram encontrados outros trabalhos com o tema similar ao trabalho proposto e que por isso ajudaram na sua definição e realização, resumidas no quadro 3. No trabalho 1, de Da Silva (2016), foi realizada a integração através de uma API entre um sistema *mobile* desenvolvido na plataforma *Android* e um sistema legado. A conclusão dessa pesquisa foi que ao utilizar uma API para integrar os sistemas *mobile* e legado, ela fez com que a troca de informações entre os sistemas fosse mais segura e ágil.

[1] Termo extraído: https://blog.atmdigital.com.br/mobilidade-corporativa-o-que-sao-aplicativos-moveis-e-por-que-investir/

No trabalho 2, apresentado por Romão (2011), o objetivo era realizar a troca de dados entre um *smartphone* que possui o SO *Android* e um sistema de dados hospitalar, utilizando um *web service* para a comunicação entre os sistemas. O autor concluiu que os resultados foram satisfatórios e que a integração foi realizada com sucesso.

O trabalho 3, segundo Júnior (2014), utilizou a solução de multiplataformas para desenvolvimento do aplicativo de autoatendimento em bares, realizando integração com o sistema de controle de pedidos através de um *web service*. A conclusão apresentada foi a facilidade de obter o aplicativo móvel em diferentes plataformas e a distribuição do mesmo.

A partir desses trabalhos analisados, com a proposta de desenvolvimento e algumas funcionalidades e requisitos abordados similares ao trabalho proposto, é possível descrever no quadro 3 algumas características de cada trabalho e um breve comparativo entre eles que ajudaram a realizar a escolha das funcionalidades e recursos do aplicativo e do *web servic*e proposto no trabalho deste TCC, o que será apresentado no capítulo 5.

Quadro 3 – Características dos trabalhos relacionados

Trabalhos	**Tipo de Desenvolvimento**	**Plataforma**	**Integração**
Trabalho 1	Nativo	*Android*	API
Trabalho 2	Nativo	*Android*	*Web Service*
Trabalho 3	Multiplataforma	*Android*/iOS	*Web Service*

Fonte: o autor.

As principais informações e tecnologias a serem comparadas nos trabalhos relacionados são: o tipo de desenvolvimento, plataforma e as ferramentas de integração. Pode-se analisar nos trabalhos relacionados, que o desenvolvimento nativo como sendo o principal, e a plataforma de desenvolvimento mais utilizada foi a *Android*. Sendo que a ferramenta com maior destaque utilizada para realizar a integração entre os sistemas foram os *web services*. A análise dos trabalhos de conclusão citados ajudou no desenvolvimento do trabalho presente.

4 INTEGRAÇÃO ENTRE SISTEMAS

A integração das informações entre os diferentes tipos de sistemas, é essencial para as organizações que possuem uma gama diversificadas de sistemas. Se as informações não estiverem integradas, apesar de estarem disponíveis, o resultado será processos menos eficientes e com maiores gastos de tempo e dinheiro. O grande fluxo de informações nas organizações vem aumentando e a tendência é crescer ainda mais no decorrer dos próximos anos. Em uma organização pode haver inúmeros sistemas, a informação contida em cada um desses sistema muitas vezes precisa estar compartilhada nos diversos setores da empresa, por isso a necessidade de integração entre sistemas.

A integração entre os SI é definida como a expansão da informação e dos processos em uma organização, seja ela através de base de dados ou aplicativos (MARTINS, 2005).

A TI integrada às estratégias de negócio, passou a ser um método eficaz e de grande crescimento para as organizações, passando a gerar uma supremacia competitiva. Porém algumas razões fazem com que as organizações passem a buscar soluções de integração entre seus sistemas (ABILIO; MALHEIROS, 2015), são elas:

a) utilizar a tecnologia existente para redução de custos na implantação;
b) a interação com *stakeholders* aumenta a gama de serviços;
c) informações comuns vindas de diferentes sistemas, concentradas numa mesma plataforma.

O objetivo de realizar integração entre sistemas é facilitar a comunicação entre os dados e processos. Por ser um conceito muito amplo a integração dos SI, passa a ser classificada em quatro perspectivas tecnológicas (MARTINS, 2005) que estão citadas abaixo:

a) Integração de Informações (II): nesta perspectiva o destaque é a informação, com a conformação da gestão e disponibilidade.
b) Integração Aplicacional (IA): integrar as aplicações é o principal alvo nesta perspectiva.
c) Integração de Processos (IP): o processo organizacional é o foco nesta perspectiva.
d) Integração Inter Organizacional (IO): nesta perspectiva o destaque está na troca de informações entre as organizações.

O resultado esperado com a integração dos sistemas é que as organizações alcancem seus objetivos e necessidades, independentemente das perspectivas citadas anteriormente.

Cada uma delas possui vantagens e desvantagens, e cabe às organizações verificar a qual se enquadram melhor.

4.1 TIPOS DE INTEGRAÇÃO

As organizações, para compreenderem qual a melhor forma de integrar seus dados, precisam conhecer precisamente os seus processos de negócios, só assim saberão quais dados desejam integrar e em quais processos. A integração entre as diversas aplicações sempre foi um desafio, e a necessidade de integrar fez com que as soluções passassem a atender apenas às necessidades das partes interessadas, gerando assim um esquecimento para as ferramentas integradoras já existentes (KARL,1994).

Para que o processo de integração entre as aplicações tenha sucesso é preciso que tais ferramentas integradoras sejam compatíveis com as necessidades apresentadas, pois só mediante a isso é possível obter êxito na integração (KARL,1994). Diante deste fato pode-se mesclar os diferentes tipos de integração entre as aplicações e utilizar dois autores Silva (2004) e Aalst (2011) para destacar quatro tipos diferentes de integração: plataforma, dados, funcional e interface.

4.1.1 Plataforma

A integração realizada entre plataformas tem como base os processos distribuídos, pode-se destacar aqui a integração utilizando os *web services*. Esse tipo de integração está dividido em três camadas: primeira camada a aplicação central, segunda camada núcleo de integração e a terceira camada as ferramentas que realizam a integração. A primeira camada é onde está toda a informação necessária para realizar a troca de dados entre os sistemas, além de conter as informações de interfaces, na segunda é onde encontram-se os padrões para realizar a comunicação entre a primeira e a terceira camada, padronizando as integrações. E a terceira camada é aquela em que as ferramentas ficam responsáveis por implementar as funções que a primeira camada não forneceu.

4.1.2 Dados

Neste tipo de integração conseguir compartilhar os dados entre as aplicações é de suma importância. Os dados vindos de uma aplicação distinta devem ser compreendidos por

outras aplicações. Esse tipo de integração só é uma opção útil caso os dados que serão manipulados não possuam um alto nível de complexidade no banco de dados. A integração de dados pode estar dividida em diferentes níveis: arquivos compartilhados e repositórios compartilhados.

No nível de arquivos compartilhados as ferramentas foram desenvolvidas para compreender alguns arquivos, além de necessitarem conhecer a estrutura lógica destes arquivos. Utilizar repositórios compartilhados é uma das maneiras mais flexíveis de realizar uma integração, porém as aplicações devem acessar constantemente em busca de dados, o que gera uma perda de desempenho.

4.1.3 Funcional

Essa forma de integrar pode considerar os outros sistemas junto aos quais se está realizando a integração como uma extensão sua, onde aqui se encaixam as aplicações por APIs. Utilizar esse nível de integração não gera problemas futuros com versões inconsistentes, pela lógica e os dados estarem compartilhados, e a reescrita não é necessária com as mudanças de versões. Apesar deste modelo ser o mais ajustável e correto no momento de integrar, o trabalho para ser implementado na organização é grande (SILVA, 2004).

Algumas formas de utilizar essa integração são:

a) criar aplicações baseadas em aplicações existentes;
b) garantir a integridade dos dados nas mais diferentes aplicações;
c) utilizar funções já existentes para futuras aplicações.

4.1.4 Interface

A integração por interface, assim como a integração por dados, não necessita de um código fonte. Outro fator importante é que mesmo a base de dados estando inacessível é possível realizar a integração. Isso só acontece, pois, as validações são associadas a interface antes mesmo de haver uma leitura ou escrita dos dados, resultando em maior segurança na integração (SILVA, 2004).

Alguns dos cenários que é utilizado esse tipo de integração:

a) utilização de padrões de texto;
b) acesso *web*;
c) melhorar a interface de aplicações antigas.

4.2 TECNOLOGIAS DE INTEGRAÇÃO

Segundo Silva (2004) a integração entre sistemas tornou-se um tema inevitável, cada vez mais as organizações buscam por soluções e os fornecedores estão cada dia evoluindo mais em *software* que disponibilizam essas soluções. Este trabalho não tem como objetivo especificar cada uma das tecnologias, mas apresentar alguns atributos delas. As tecnologias citadas nas subseções a seguir possuem vantagens e desvantagens e são utilizadas muitas vezes em conjunto, umas com as outras nas organizações.

4.2.1 Chamada a procedimentos remotos (RPC)

A tecnologia RPC surgiu na década de oitenta com Birrel e Nelson e permitiu uma nova forma de comunicação entre os processos (TANENBAUM, 2007). Essa comunicação fez com que aplicações executassem processos em diferentes estações. O RPC funciona da seguinte forma: um aplicativo da estação “E1” aciona um processo executado na estação “E2”, o processo que foi executado em estação “E1” fica pendurado e o processo que foi solicitado passa a funcionar na estação “E2” (TANENBAUM, 2007). Esse protocolo ajusta-se a processos remotos como em processos locais (DE SORDI; MARINHO, 2007). Este tipo de comunicação, pelas suas características, define um processo síncrono, mas em algumas situações podem ser assíncronos, não requerendo uma resposta da chamada do processo (SILVA, 2004).

4.2.2 Troca eletrônica de arquivos (EDI)

A troca de dados entre as organizações passou a ser uma necessidade eminente, e a decisão encontrada para suprir tal necessidade foi a troca eletrônica de arquivos entre os sistemas. Os documentos para serem trocados pelas organizações, devem passar por uma padronização em seu formato através de um acordo pré-estabelecido pelas partes (MARTINS, 2005). Os arquivos EDI possui um *layout*, que está definido no seguinte formato: um cabeçalho (*header*) no qual estão contidas as informações de data e do remetente e do destinatário; um corpo (*detail*) no qual estão inseridas as informações do conteúdo que está presente no documento; e um sumário (*summary*) que contém informações de totalizadores e controle de linhas (MARTINS, 2005).

4.2.3 Middleware orientado a mensagem (MOM)

A tecnologia MOM realiza troca de mensagens entre cliente e servidor de forma síncrona e também assíncrona. Essas mensagens são enviadas ao MOM que as estrutura em filas, acessadas assincronamente, sendo gerenciado assim o fluxo de informações entre cada sistema/aplicação que está ligado a essa tecnologia (MARTINS, 2005). Essa integração utiliza *middleware* estruturado em mensagens, para gerar, utilizar, gravar e enviar/receber mensagens. A integração de aplicações que utilizam a tecnologia MOM proporciona uma enorme confiabilidade em relação aos dados (SILVA, 2004). Para essa tecnologia estar funcionando, ela precisa estar concretizada dos dois lados para permitir a comunicação.

4.2.4 Acesso direto a base de dados

A utilização dessa técnica é aproveitada se as aplicações manipulam a mesma base de dados, utilizando drivers ODBC, OLEDB e funções *Structured Query Language* (SQL). Essa técnica não é aconselhável, por risco de corromper a base de dados, pois as aplicações ao realizarem alterações na base de dados precisam comunicar as demais aplicações que estão conectadas à base que foi realizada uma alteração (SIMONNETTI, 2002, DE SORDI; MARINHO, 2007). Se a base de dados for controlada por um SGBD multiusuário, então o SGBD deve garantir que as alterações múltiplas mantenham a base de dados consistente. Mas essa técnica é útil quando as aplicações compartilham apenas os dados, e não outras funcionalidades.

4.2.5 Web services

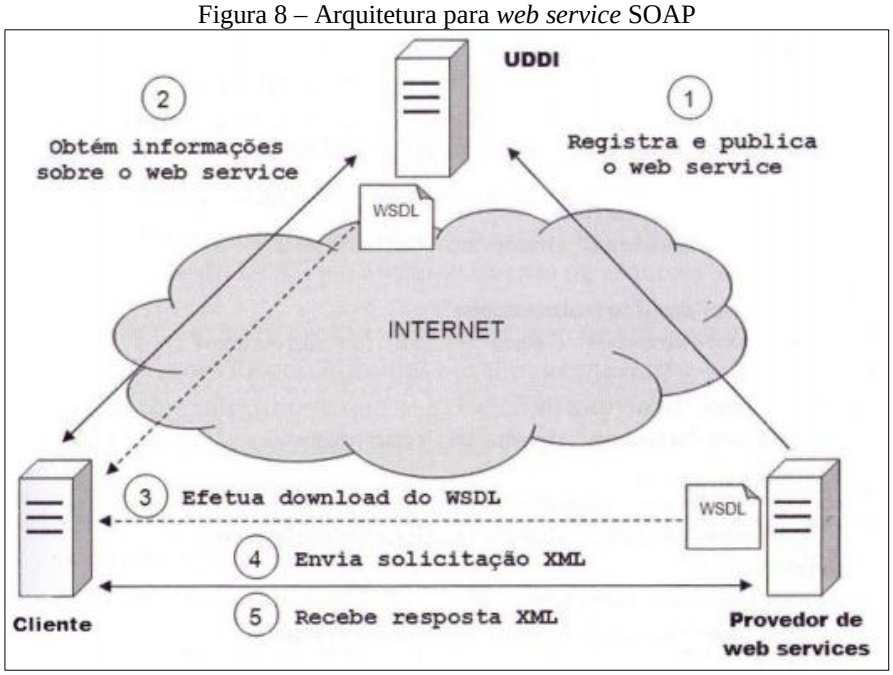
Os *web service* utiliza o conceito *Business to Business* (B2B) para realizar a integração de aplicações. De forma genérica, um *web service* é um serviço oferecido por um sistema para outro sistema, e foi desenvolvido para interligar aplicações implementadas em diferentes linguagens de programação. O objetivo de um *web service* é a comunicação entre as aplicações através da *internet*, e essa comunicação é realizada utilizando os protocolos padrão da *internet* (www.w3.og). A junção entre *web services* e aplicativos móveis fez com que o desenvolvimento de tais aplicativos ficasse muito mais simples, pois toda a integração fica por conta do *web service* (SADAGI, 2013). A comunicação entre *web services* em geral tem por base SOAP ou REST.

4.2.5.1 SOAP

A comunicação de um *web service* é definida através do *Simple Object Access Protocol* (SOAP), que organiza como enviar e receber mensagens num formato *Extensible Markup Language* (XML). O protocolo *Hyper Text Transport Protocol* (HTTP) é quem se encarrega de enviar os dados. Associado ao SOAP está o documento *Web Service Definition Language* (WSDL) que informa a localização do *web service* e os serviços que ele dispõe. Além disso, fornece a informação necessária para que a comunicação entre sistemas seja possível (MARTINS, 2005).

A figura 8 apresenta o funcionamento de um w*eb service* SOAP e os componentes que estão envolvidos em seu funcionamento. Os números que estão presentes na figura 8, são utilizados para gerar a ordem das operações.

Figura 8 – Arquitetura para *web service* SOAP



Fonte: (GOMES, 2010).

a) registrar e publicar o *web service*: ao criar um *web service* ele fica armazenado em um provedor junto com seu arquivo WSDL. Para que este *web service* esteja disponível para ser acessado e utilizado, é preciso saber o seu endereço URL. Após a criação e o armazenamento, é preciso registrá-lo e publicá-lo em um diretório denominado *Universal Description, Discovery and Integration* (UDDI) (GOMES, 2010). A utilização de UDDIs não se efetivou, e atualmente os *web services* são registrados e publicados em um servidor *web*, e sua divulgação ocorre através de

sites, portais, redes sociais, entre outros meios, através da disponibilização do *link* de acesso e outras informações relevantes;

b) obtém informações sobre o *web service*: para poder utilizar um *web service* é preciso saber onde ele está, ou seja, a sua URL e seu arquivo WSDL. Através de sua divulgação (via UDDI ou outro meio) é que terá essas duas informações necessárias para acessar o *web service* (GOMES, 2010);
c) efetua *download* do WSDL: obtendo a URL o desenvolvedor poderá realizar o download do arquivo WSDL, e posteriormente criar um cliente que irá consumir os serviços deste *web service* (GOMES, 2010);
d) envia solicitação XML: o *software* que o cliente possui realiza requisições ao *web service* com arquivos no formato XML (GOMES, 2010);
e) recebe resposta XML: após o *web service* receber a solicitação anterior, realiza o processamento desta solicitação e produz um resultado. O resultado deste processamento pode ser ou não enviado ao cliente dependendo da operação que *web service* executa, no formato XML (GOMES, 2010).

4.2.5.2 REST

O estilo arquitetural *Representational State Transfer* (REST) é uma outra forma de comunicação entre *web services*, o qual oferece funções mais simples e mais acessíveis (*RESTful APIs*) do que o padrão SOAP. Essas funções podem ser executadas de forma síncrona e assíncrona. O que é principal em um *web service REST* são suas *Uniform Resource Identifier* (URIs) e seus serviços (SADAGI, 2013).

O protocolo que o estilo arquitetural REST utiliza para transporte é o HTTP, além de ser um protocolo é considerado também um API, que compreende métodos de acesso. Esses métodos são consumidos como CRUD (*Create, Read, Update e Delete*) e são: *Post, Get, Put e Delete* (KALIM, 2010).

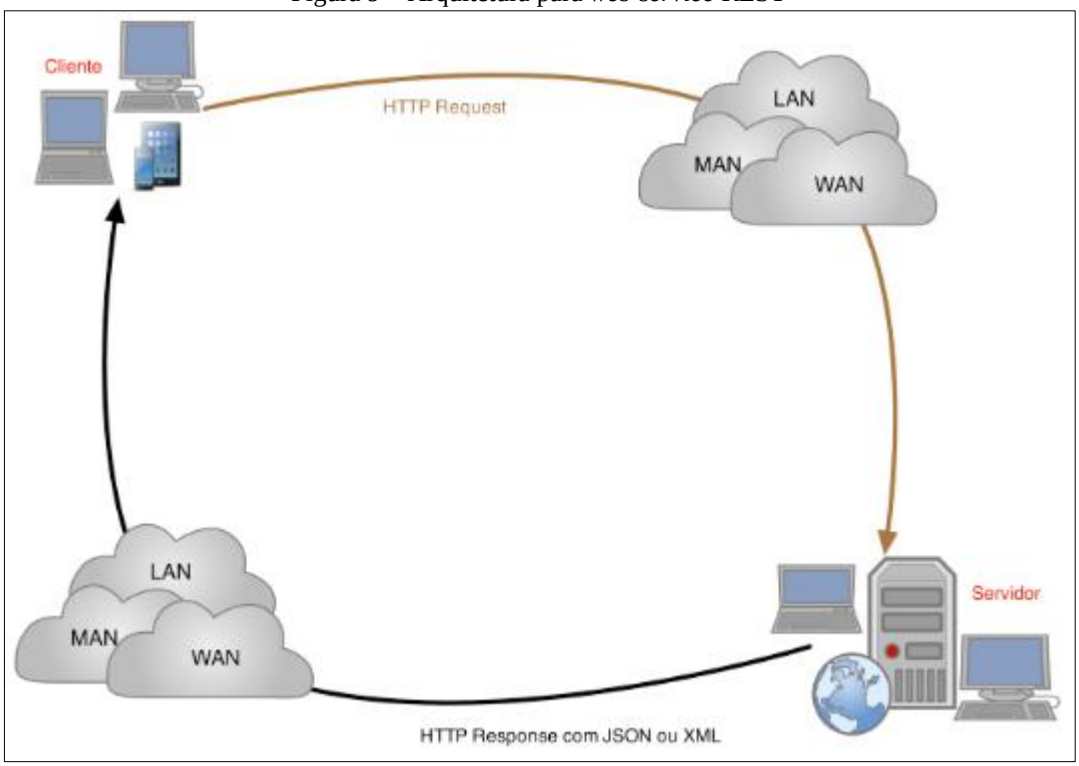
Para definir qual método a aplicação deverá executar é necessário que seja enviado no *Body* da solicitação o tipo de requisição realizada, que podem ser:

a) *get*: o método é uma operação apenas de leitura que obtém informações de um determinado recurso no servidor;
b) *put*: o método é utilizado para atualizar informações de um recurso no servidor;
c) *post*: o método é utilizado para criar um recurso;
d) *delete*: método utilizado para remover um recurso.

O REST é um estilo arquitetural híbrido que é originário de outros estilos arquiteturais que são baseados em rede que passam a definir uma interface de conexão uniforme. Para que seja possível a troca de dados entre webservices são utilizados os padrões XML ou *JavaScript Object Notation* (JSON), sendo que o uso de cada uma deve ser estudado e definido de acordo com a necessidade do ambiente (KALIM, 2010).

A figura 9 apresenta o funcionamento de um w*eb service* REST e os componentes que estão envolvidos em seu funcionamento. A aplicação cliente realiza o envio de uma requisição HTTP, com as informações necessárias, para o servidor de aplicação. O servidor realiza o processamento da requisição e devolve para o cliente uma resposta HTTP, contendo um arquivo XML ou um JSON, esse arquivo pode conter uma simples mensagem ou até mesmo um conjunto de informações difíceis (CARNEIRO, 2013).

Figura 9 – Arquitetura para *web service* REST



Fonte: (CARNEIRO, 2013).

Conforme Jakl (2005) o REST compreende três classes de elementos arquitetônicos: de dados, de conexão e de processamento.

Dentro da classe de dados os componentes se comunicam entre si. Este é o aspecto principal do REST, o estado dos elementos de dados, que podem ser transferidos para as atuais representações, ou do estado desejado dos elementos de dados. Através de identificadores capazes de diferenciar os diversos recursos.

Conhecidos como conectores, os elementos de conexão são responsáveis por gerenciar a rede de comunicação dos componentes. Eles podem apresentar uma interface abstrata, podendo assim oferecer uma melhor separação dos interesses da implementação.

Os componentes são os elementos de processamento, que podem ser identificados através das funções atribuídas a eles dentro de uma aplicação, que pode ser de: usuário, servidor ou de intermediário.

Assim como na figura 9 o processo se inicia através do usuário utiliza um conector de cliente para acionar um solicitação, que então se torna posteriormente o destinatário final da resposta. Neste meio tempo até a resposta final, o servidor de origem a um conector de servidor para receber a solicitação e representação de seus recursos. É importante destacar que cada servidor possuem uma hierarquia de recursos, fornecendo uma interface genérica para seus serviços. E por fim temos os componentes intermediários que atuam cliente e servidor afim de promover respostas às solicitações.

4.2.6 API

API é um conjunto de funcionalidades e padrões utilizadas por um aplicativo. As APIs são muito utilizadas para realizar a integração entre os aplicativos, porém para que essa integração funcione um dos aplicativos precisa disponibilizar suas funcionalidades para os demais. A utilização dessa forma de integração contém algumas desvantagens como por exemplo: requer mão de obra específica, pode gerar dificuldades na sua implantação e retrabalho com interfaces caso as aplicações sejam alteradas. Essa integração contém duas direções, a unidirecional e a bidirecional, sendo a primeira a mais utilizada por sua simplicidade. Atualmente as APIs são criadas em conjuntos/*kits* de integração para diminuir o grau de complicação na sua integração, tornando-as mais flexíveis e fáceis de alterar, o que torna o seu processo implantação menos complexo (ZANCUL; GUERRERO; ROSZENFELD; OLIVEIRA, 2000).

4.3 DIFERENÇA ENTRE WEB SERVICE E API

A utilização de diversos sistemas nas organizações para atender as demandas do negócio e a busca por vantagem competitiva levantam um ponto crucial no momento de realizar uma implantação entre esses diversos sistemas: qual ferramenta utilizar para tornar esse processo dinamizado. As APIs e os *web services* são uma boa opção para realizar essa

integração, foram criados com o intuito de facilitar a transferência de informação entre os sistemas no dia-a-dia das empresas.

A API visa facilitar a comunicação com uma aplicação, já os *web services* seriam a própria aplicação. Apesar de essas duas ferramentas de integração serem confundidas, elas possuem algumas diferenças, como está descrito no quadro 4.

Quadro 4 – Principais diferenças entre web service e API

Web Service	**Web API**
Todo o tipo de *web Service* é uma API.	Nem todas as APIs podem ser consideradas com um *web Service.*
O *web service* necessita de uma rede para seu funcionamento.	Uma API não precisa de acesso à uma rede para funcionar.
Principais tipos de comunicação: SOAP, REST.	Qualquer tipo de comunicação pode ser desenvolvido.

Fonte: Blog Test Automation Resources[2].

[2] Quadro extraído: https://testautomationresources.com/api-testing/differences-web-services-api/.

5 PROPOSTA DE DESENVOLVIMENTO

Nos capítulos desenvolvidos anteriormente foram apresentadas as tecnologias que irão compor a proposta de desenvolvimento deste projeto.

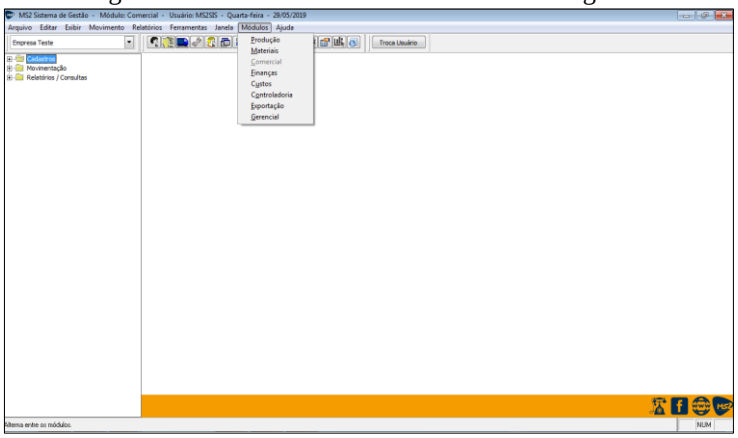
A proposta de desenvolvimento está dividida em duas partes, a parte do aplicativo móvel (fundamentada no capítulo 3), e a parte da ferramenta para integração (fundamentada no capítulo 4) com o ERP MS2 Sistema de Gestão Clínica. O protótipo do aplicativo móvel será desenvolvido na plataforma *Android* (descrito na seção 3.1.1). Após realizar a revisão bibliográfica, optou-se por realizar a comunicação entre o aplicativo e o sistema MS2 através de um *web service* REST (descrito na seção 4.3.5).

Neste capítulo serão apresentadas as propostas do aplicativo móvel e do *web service*, juntamente com o levantamento de informações: requisitos funcionais e não funcionais, protótipos de telas e diagramas para o desenvolvimento: caso de uso, e de atividades, além de ferramentas e técnicas a serem utilizadas para obtenção do resultado e especificações do aplicativo para o ERP MS2 Sistema de Gestão Clínico.

5.1 ERP MS2 SISTEMA DE GESTÃO CLÍNICO

O ERP MS2 Sistema de Gestão está dividido em módulos que são selecionados por seus clientes de acordo com sua necessidade, tornando-o assim um produto específico e sob medida. São sete os módulos: produção, materiais, comercial, custo, controladoria, exportação e gerencial, sendo que dentro de cada módulo existem funcionalidades específicas como cadastros, lançamentos de atividades e geração de relatórios. A figura 10 exemplifica a disposição dos módulos no sistema MS2.

Figura 10 – Módulos do ERP MS2 sistema de gestão



Fonte: o autor.

O software de ERP MS2 Sistema de Gestão Clínico é uma solução que tem como base o ERP principal da empresa, sendo que esse sistema de gestão é voltado para a administração das informações de clínicas e consultórios oftalmológicos. O MS2 Clínico é dividido em módulos conforme o ERP principal, tendo o módulo agenda como seu diferencial. É neste módulo que é realizada a gestão do paciente, configuração de horários, controle e inclusão de exames, dentre outras atividades.

5.1.1 Exame de refração

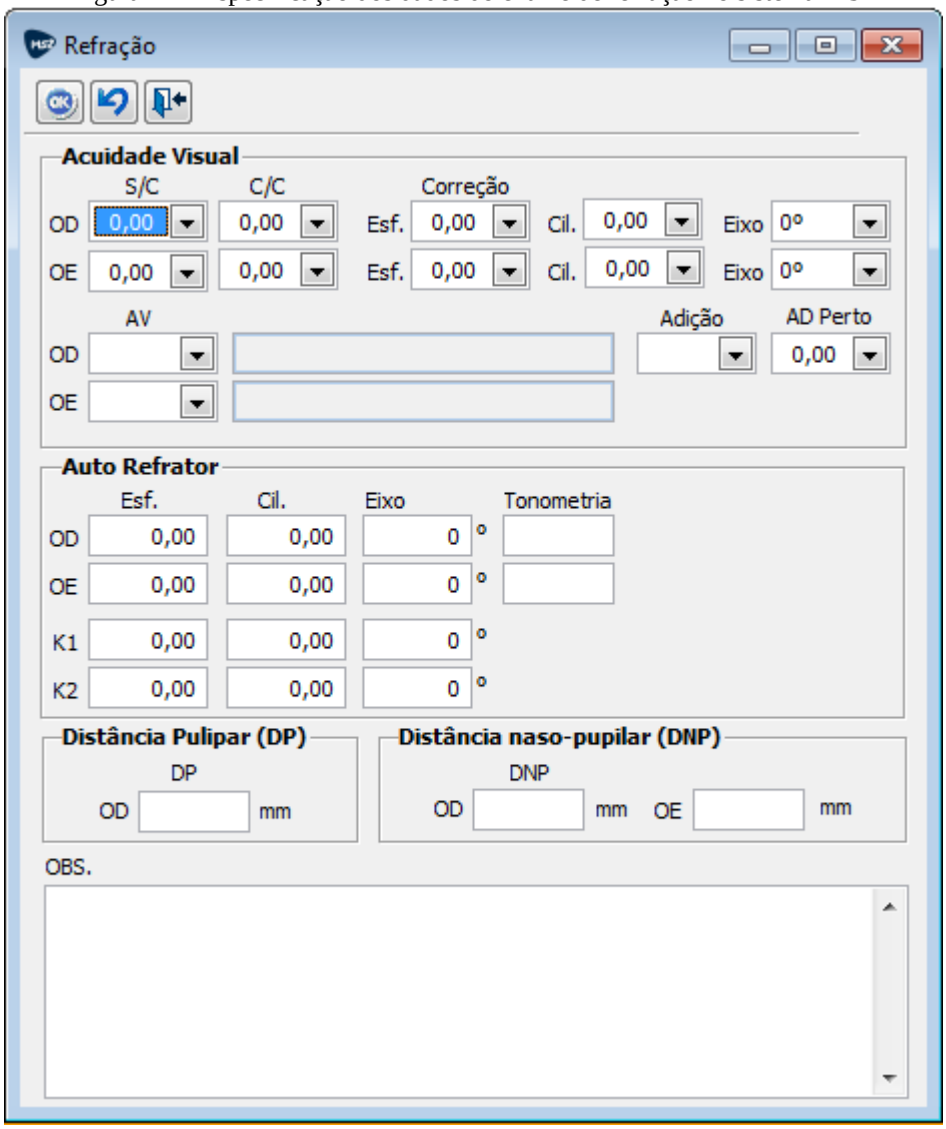
Conhecido como exame de vista, ele serve para detectar o quanto o paciente consegue enxergar e determinar o grau do óculos, caso seja necessário. Alguns dos possíveis problemas que são detectados na realização desse exame são: miopia, hipermetropia, astigmatismo e a presbiopia.

Para realizar o exame é preciso dois equipamentos: o autorefrator e o refrator. Com base nos dados coletados dos equipamentos é feito o diagnóstico e prescrita a receita que determina a acuidade visual e a refração estática do olho direito e no olho esquerdo, grau esférico e cilíndrico e sua correção. O exame também permite determinar a distância pupilar e a distância naso-pupilar.

Essas informações são preenchidas a cada exame realizado em um paciente, para assim alimentar o ERP. Porém tais informações não são enviadas imediatamente ao ERP. Após as atendentes realizarem o exame, os equipamentos onde são realizados esses exames geram um comprovante, e só mais tarde as atendentes alimentam o sistema, que nem sempre está próximo do local da realização do exame. Nesse processo, perde-se alguns exames, e compromete-se a precisão das informações do sistema.

A figura 11 apresenta a disposição de como a informação é apresentada ao usuário do *software* MS2 Sistema de Gestão Clínica.

Figura 11 – Especificação dos dados do exame de refração no sistema MS2



Fonte: o autor.

5.2 REQUISITOS DO APLICATIVO MÓVEL E DO WEB SERVICE

Com base no capítulo 3 deste documento, onde são apresentadas as plataformas que compõem os aplicativos móveis e suas devidas funcionalidades e aplicabilidades, optou-se por desenvolver uma solução através de um aplicativo móvel utilizando SO *Android*, para coletar os dados referentes ao exame de refração.

A escolha da plataforma *Android* para desenvolvimento do aplicativo móvel foi determinada, pela mesma ser gratuita, não precisando assim realizar a compra de outras ferramentas para desenvolvimento, ou qualquer outro *software*.

Para acessar os serviços que o sistema MS2 Sistema de Gestão Clínica disponibiliza, ou seja, as regras de negócios do sistema, será implementado como ferramenta de integração

um *web service REST* desenvolvido na linguagem de programação *Java*, que terá todos os serviços necessários para realizar a integração, conforme foi estudado no capítulo 4 deste documento. Os *web services* são os recursos mais utilizados atualmente para integração de sistemas pela sua flexibilidade: permitem a comunicação de sistemas implementados em diferentes linguagens e abstraem vários níveis de comunicação.

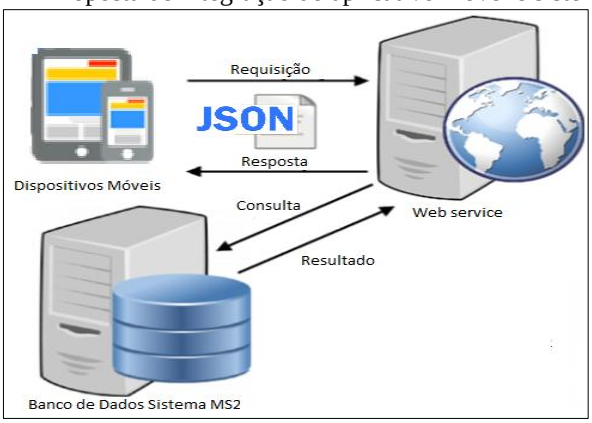
O *web service REST* estará alocado junto ao servidor onde encontra-se a base de dados do sistema MS2, o qual disponibilizará as funções para envio e recebimento de informações. Como o sistema MS2 contém a base de dados local e não possui um servidor *web*, é preciso que o *web service* seja registrado e disponibilizado em um servidor de aplicação, que foi definido como sendo o *Apache Tomcat*.

Vantagens da utilização de um aplicativo móvel e um *web service*:

a) mobilidade: o usuário pode realizar a inserção dos dados obtidos no exame na própria sala onde encontra-se o aparelho que realiza o exame, evitando assim o deslocamento até o *desktop* mais próximo;
b) usabilidade: através do *touchscreen*, o usuário poderá interagir com a interface;
c) integração: o aplicativo móvel fará o envio das informações para um *web service*, que se comunicará com o sistema ERP;
d) segurança: o aplicativo será acessado apenas por senha, de usuários já estejam cadastrados no sistema MS2, impedindo assim o acesso por pessoas não autorizadas.

A figura 12 representa as partes envolvidas na proposta desse aplicativo, o dispositivo móvel aciona os serviços que estarão presentes em um único *web service*, que contém os serviços que realizam as consultas e inserem os dados no banco de dados do sistema MS2.

Figura 12 – Proposta de integração do aplicativo móvel e sistema MS2



Fonte: o autor.

A disponibilização desse aplicativo segue as seguintes etapas:

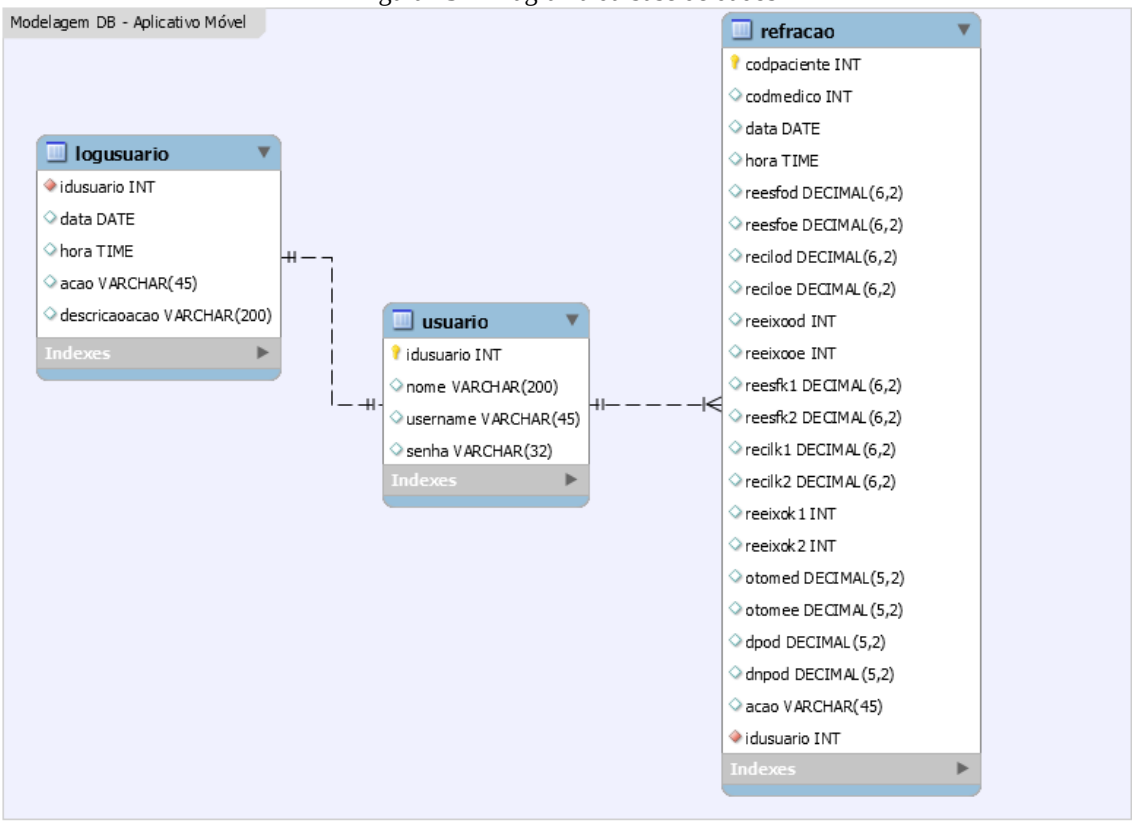
a) após a desenvolvimento do *web service* que não é público, basta registrá-lo e disponibilizá-lo no servidor *web Apache Tomcat*, que contém a descrição de todas as operações e serviços que compõem o *web service* além de sua URI;
b) com a URI do *web service*, o aplicativo móvel prepara a solicitação de um serviço contendo as informações necessárias;
c) o aplicativo inicia a conexão enviando uma mensagem de solicitação para determinado serviço do *web service*;
d) o serviço do *web service* recebe a mensagem de solicitação de serviço e encaminha imediatamente para processamento do serviço solicitado;
e) o sistema MS2 Sistema de Gestão Clínica recebe a mensagem de solicitação e realiza o processamento, devolvendo uma mensagem de resultado do processamento ao serviço do *web service*;
f) o serviço do *web service* recebe a mensagem de resultado do processamento e o encaminha ao aplicativo;
g) o aplicativo recebe a mensagem de resultado do processamento.

5.2.1 Requisitos do sistema

Foram definidos os requisitos para atender a necessidade do projeto através de entrevistas informais realizadas com o gestor da clínica, observou-se então que o principal gargalo no sistema MS2, estava no lançamentos dos dados do exames de refração. Por este exame ser exclusivamente realizado e lançado no sistema por um único usuário capacitado e habilitado, e em um tempo posterior à realização do exame, a informação em alguns casos acaba sendo perdida.

A comunicação entre o aplicativo e o sistema MS2 será assíncrona, conectando com o ERP e enviando/solicitando os dados. A figura 13 apresenta a base de dados do aplicativo móvel, que é bem simples e só irá armazenar as informações necessárias da realização dos exames. Caso não haja conexão com a *internet*, esses dados serão armazenados um banco *SQLite*, e posteriormente com conexão à *internet* restabelecida será realizada a sincronização e envio dos dados ao ERP.

Figura 13 – Diagrama da base de dados



Fonte: o autor.

A tabela *logusuario* foi modelada para armazenar as ações e as funcionalidades que o usuário acessa dentro do aplicativo móvel, um "*log*", do aplicativo. Vinculada a esta tabela está a tabela *usuario* que armazena os dados dos usuários que acessaram o aplicativo.

A tabela *refracao* possui vínculo com a tabela usuário, e é nela que ficaram armazenadas as informações referentes aos dados do exame de refração de cada paciente. Para que posteriormente, seja feita a sincronização com o sistema MS2 caso não haja conexão com *internet*, essa sincronização irá alterar, incluir ou deletar dados no banco de dados do sistema MS2.

No quadro 5 estão retratados os requisitos funcionais da proposta apresentada.

Quadro 5 – Requisitos funcionais

Identificação	Nome
RF01	Autenticação de usuário e senha.
RF02	Listar médicos em atendimento
RF03	Listar agenda dos médicos com seus respectivos pacientes
RF04	Gerenciar exames de refração

Fonte: o autor.

No quadro 6 estão retratados os requisitos não funcionais da proposta apresentada.

Quadro 6 – Requisitos não funcionais

Identificação	Nome
RNF01	O aplicativo móvel deverá ser compatível com a plataforma *Android*.
RNF02	O sistema deverá permitir a integração de dados entre aplicativo e o sistema MS2 Sistema de Gestão Clínica via *web service*.
RNF03	O sistema necessita de conexão com internet para seu funcionamento.
RNF04	O aplicativo móvel deverá utilizar o banco de dados SQLite para armazenamento.
RNF05	O *web service* deverá utilizar como servidor de aplicação *Apache Tomcat*.
RNF06	O *web service* deverá utilizar a arquitetura REST.
RNF07	O *web service* deverá retornar os dados consultados do banco de dados do sistema MS2 no formato JSON.

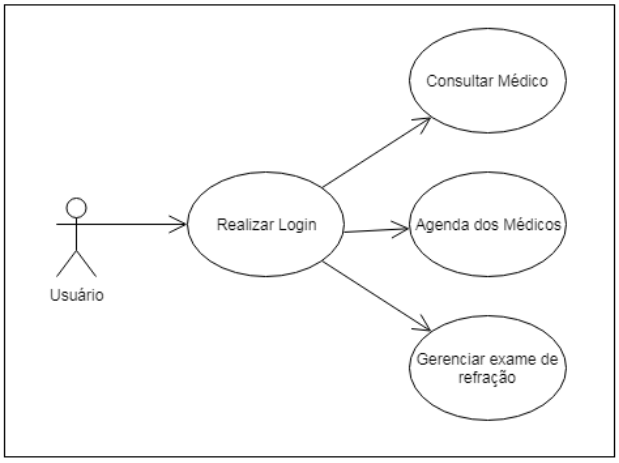
Fonte: o autor.

5.2.2 Casos de uso

Conforme os requisitos funcionais apresentados, foram definidos os casos de uso para ajudar a compreensão das funcionalidades do sistema. A figura 14 representa uma perspectiva geral dos casos de uso do sistema, onde compreende um autor que se comunicara com o sistema:

a) usuário (especialista): usuário que utilizará o sistema para lançar os dados referentes ao exame de refração.

Figura 14 – Caso de uso do aplicativo móvel

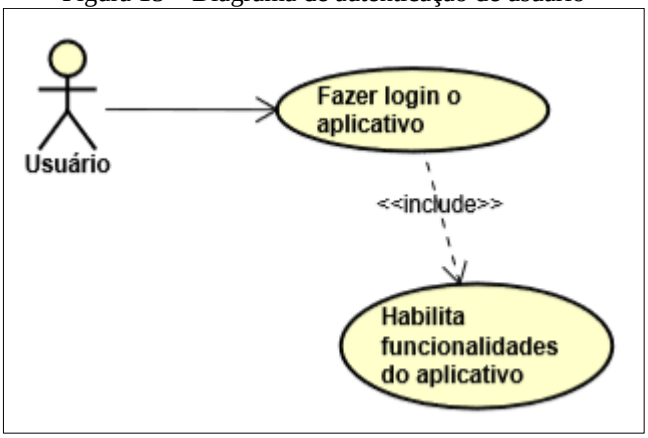


Fonte: o autor.

5.2.2.1 Caso de uso 01

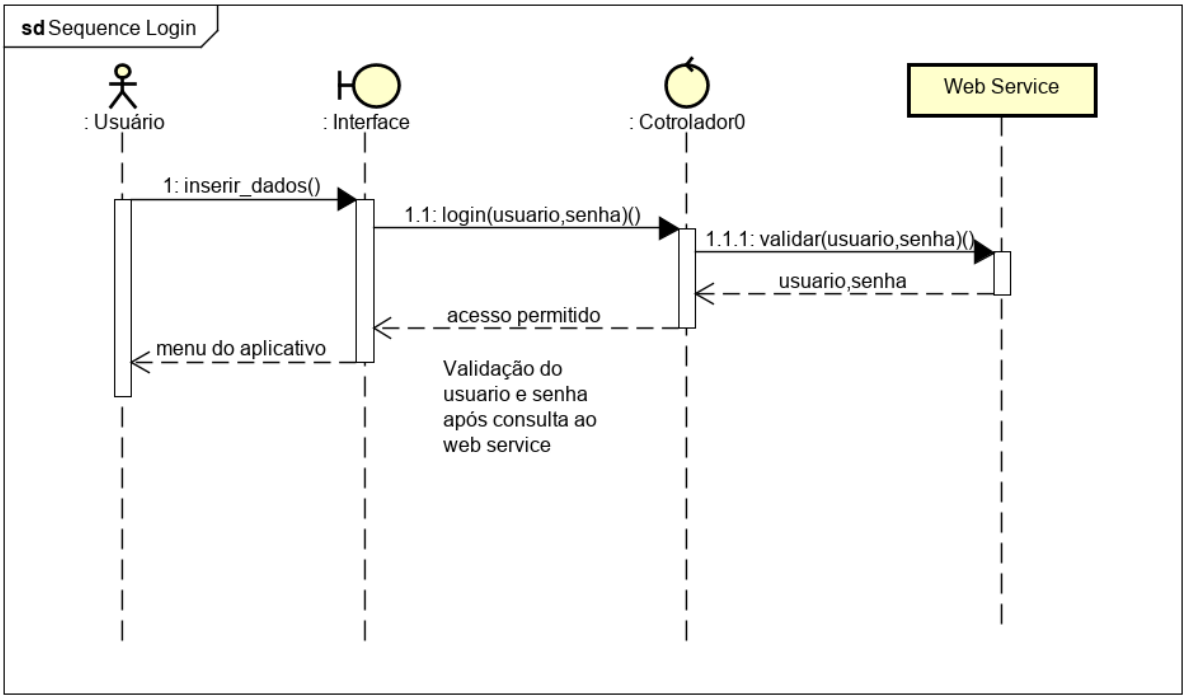
Autenticação de usuário e senha: o aplicativo deve autenticar o usuário e senha do usuário que deseja acessá-lo. Esse usuário deve estar cadastrado no sistema MS2 Sistema de Gestão Clínica para possuir acesso.

Figura 15 – Diagrama de autenticação de usuário



Fonte: o autor.

Figura 16 – Diagrama de sequência: caso de uso 01 (autenticação de usuário)



Fonte: o autor.

Abaixo serão apresentadas as partições do diagrama de atividade da figura 16.

Realizar *login* no aplicativo móvel

Escopo: aplicativo móvel.

Nível: objetivo do usuário.

Ator principal: usuário (especialista que realiza o exame de refração).

Lista de interessados:

a) especialista: realiza o *login* para consultar os pacientes agendados do dia conforme cada médico e inserir os dados obtidos no exame do paciente. Não necessita realizar o *login* toda vez que acessa o aplicativo, pois precisa ser prática essa funcionalidade a ela.

Pré-condições:

a) o usuário estar cadastrado no sistema MS2 Sistema de Gestão Clínica;
b) *o web service* deve estar disponível.

Garantia de sucesso: o usuário foi autenticado e está capacitado a acessar o aplicativo.

Cenário de sucesso:

a) o usuário digita seu usuário e senha, aperta o botão OK e aguarda;
b) o sistema verifica se os dados informados se encontram no banco de dados do sistema MS2;
c) o sistema verifica que está OK e o usuário passa a acessar o sistema.

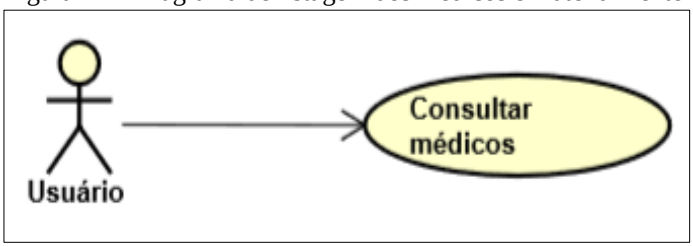
Fluxo alternativo:

a) o usuário digita seu usuário e senha, aperta o botão OK e aguarda;
b) o sistema verifica se os dados informados se encontram no banco de dados do sistema MS2;
c) o sistema não encontra os dados e retorna ERRO informando: “Usuário e Senha não cadastrado, verifique”;
d) os campos de usuário e senha são limpos.

5.2.2.2 Caso de uso 02

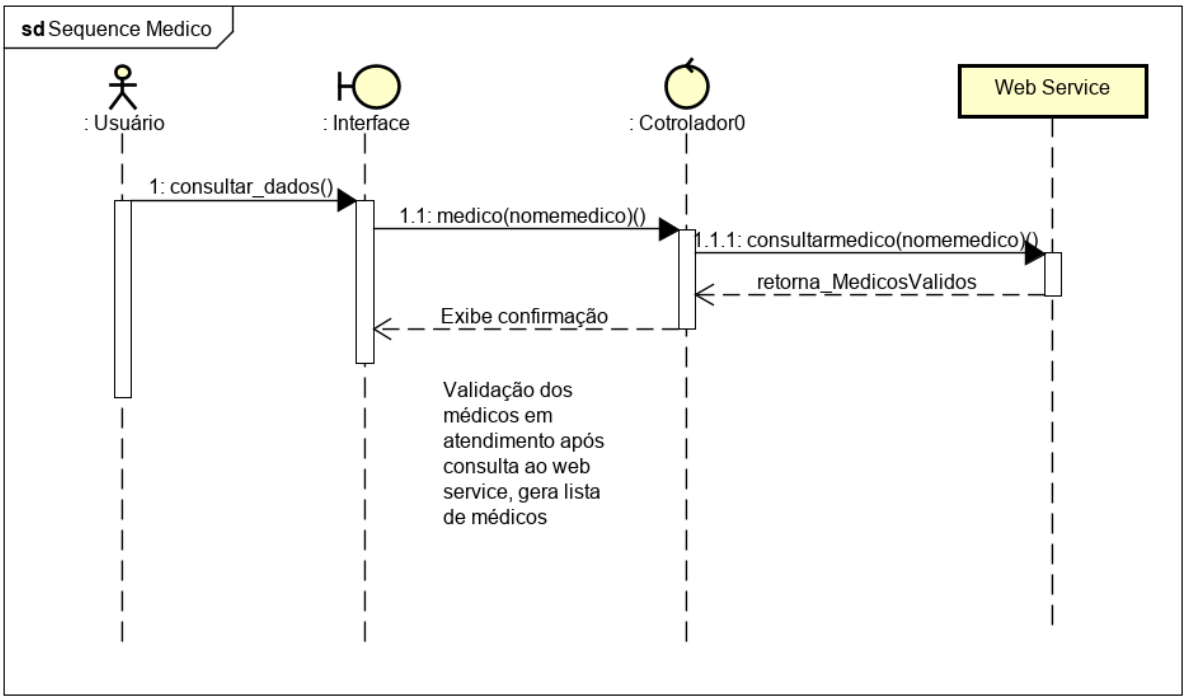
Listar médicos em atendimento: o aplicativo mostra os médicos que possuem em sua agenda algum paciente com consulta.

Figura 17 – Diagrama de listagem dos médicos em atendimento



Fonte: o autor.

Figura 18 – Diagrama de sequência: caso de uso 02 (consultar os médicos)



Fonte: o autor

Abaixo serão apresentadas as partições do diagrama de atividade da figura 18.

Consultar os médicos

Escopo: aplicativo móvel.

Nível: objetivo do usuário.

Ator principal: usuário (especialista que realiza o exame de refração).

Lista de interessados:

a) especialista: deseja visualizar os médicos que possuem pacientes com consultas já incluídas anteriormente pelo sistema MS2.

Pré-condições:

a) pelo menos um paciente possuir horário na agenda de algum médico;
b) o *web service* deve estar disponível.

Garantia de sucesso: o médico foi consultado com sucesso.

Cenário de sucesso:

a) o usuário informa o nome médico que deseja consultar;
b) aperta o botão “Consultar Médico” e aguarda;
c) o sistema verifica se o médico informado se encontra no banco de dados do sistema MS2;
d) visualiza se o médico possui alguma consulta agendada, para determinada data.

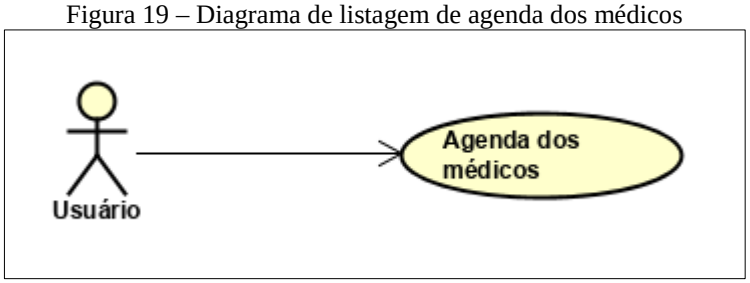
Fluxo alternativo:

a) o usuário não informa nenhum nome de médico;
b) o sistema retorna todos os médicos que possuem consultas agendadas no dia.

5.2.2.3 Caso de uso 03

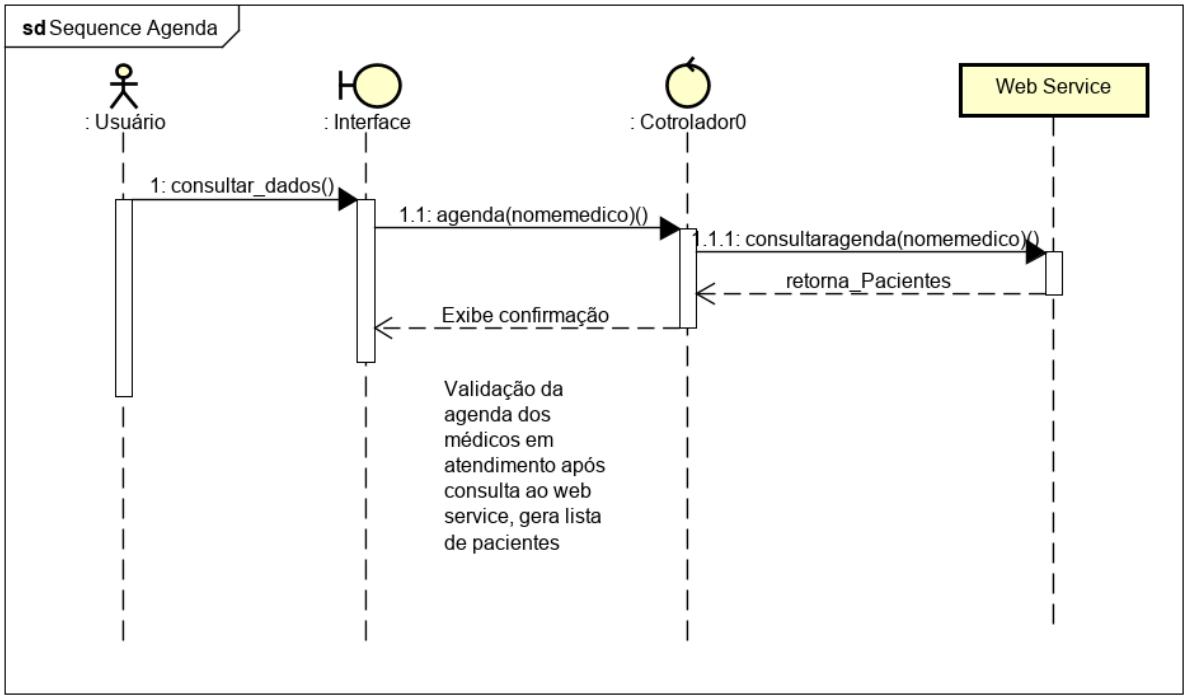
Listar agenda dos médicos com seus respectivos pacientes: após os pacientes já possuírem seu horário cadastrado através do sistema MS2, o aplicativo mostra os horários de cada paciente conforme o médico selecionado.

Figura 19 – Diagrama de listagem de agenda dos médicos



Fonte: o autor.

Figura 20 – Diagrama de sequência: caso de uso 03 (agenda dos médicos)



Fonte: o autor

Abaixo serão apresentadas as partições do diagrama de atividade da figura 20.

Consultar agenda dos médicos com seus respectivos pacientes

Escopo: aplicativo móvel.

Nível: objetivo do usuário.

Ator principal: usuário (especialista que realiza o exame de refração).

Lista de interessados:

a) especialista: deseja visualizar os pacientes com consultas já incluídas anteriormente pelo sistema MS2.

Pré-condições:

a) pelo menos um paciente possuir horário na agenda de algum médico;
b) o *web service* deve estar disponível.

Garantia de sucesso: a agenda do médico foi consultada com sucesso.

Cenário de sucesso:

a) o usuário informa o nome do médico que deseja consultar a agenda;
b) visualiza seus pacientes agendados.

Fluxo alternativo:

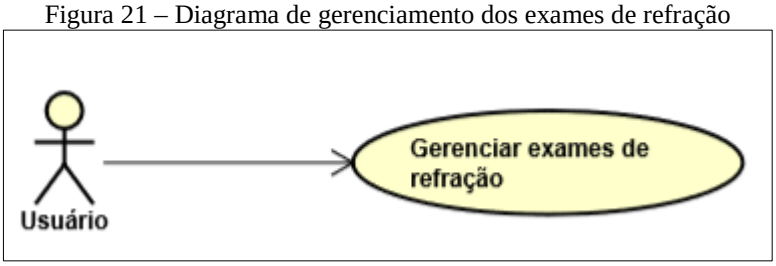
a) o usuário não informa nenhum nome de médico;

b) o sistema retorna todos os paciente que possuem consultas agendadas no dia.

5.2.2.4 Caso de uso 04

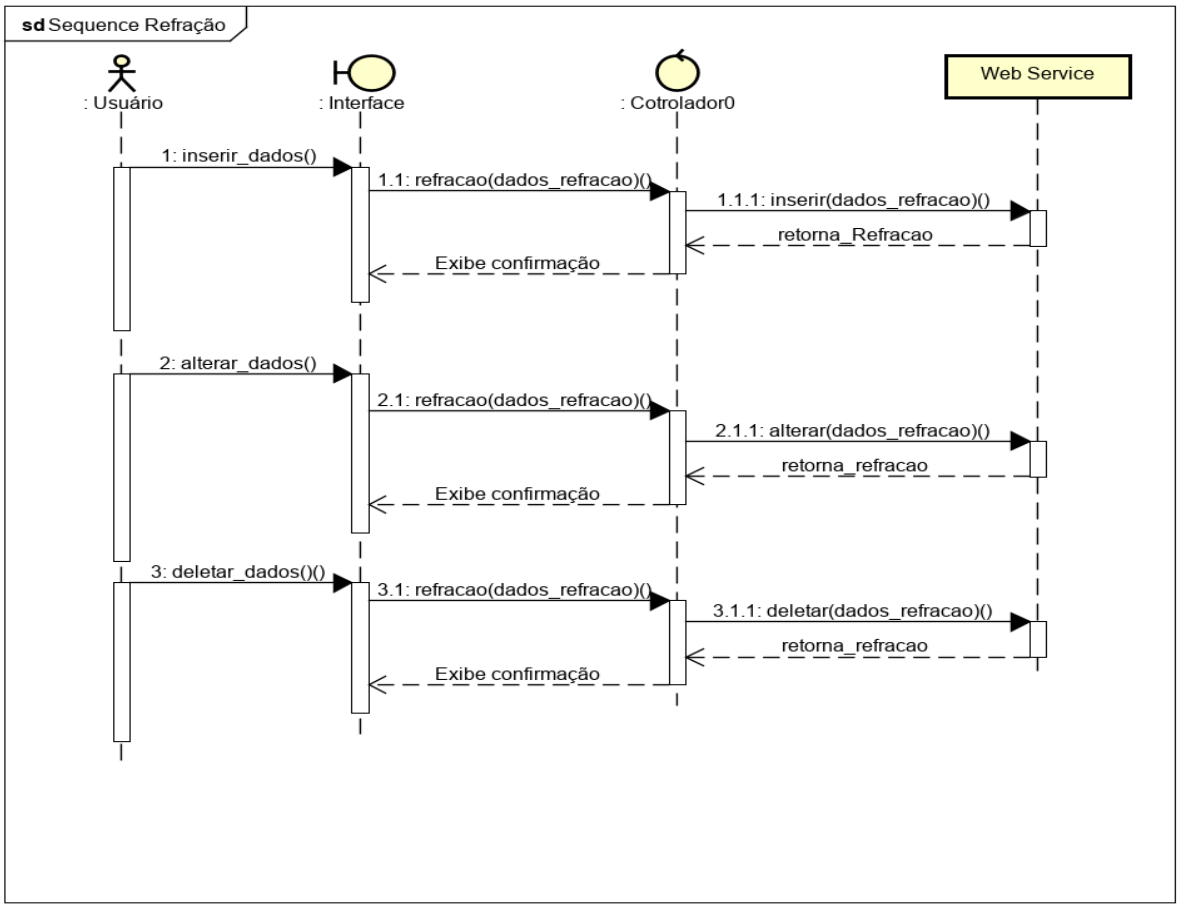
Gerenciar exames de refração: o aplicativo deve permitir ao usuário que o está acessando para incluir, alterar e excluir os exames de refração dos pacientes.

Figura 21 – Diagrama de gerenciamento dos exames de refração



Fonte: o autor.

Figura 22 – Diagrama de sequência: caso de uso 04 (gerenciamento exame de refração)



Fonte: o autor.

Abaixo serão apresentadas as partições do diagrama de atividade da figura 22.

Gerenciar exame de refração
Escopo: aplicativo móvel.
Nível: objetivo do usuário.
Ator principal: usuário (especialista que realiza o exame de refração).

Lista de interessados:

a) especialista: deseja gerenciar o exame de refração dos pacientes que possuem consulta agendada de forma prática. Deseja incluir o resultado desse exame quando o mesmo for finalizado.

Pré-condições:

a) o paciente possuir horário na agenda do médico;
b) o *web service* deve estar disponível.

Garantia de sucesso: os exames de refração foram gerenciados com êxito.
Cenário de sucesso:

a) usuário seleciona o paciente e seleciona qual dado do Exame de Refração deseja inserir;
b) digita os dados do exame de refração;
c) usuário clica no OK;
d) a mensagem “Registro Adicionado ao Paciente.” é exibida.

Fluxo alternativo:

a) usuário seleciona o paciente, e seleciona qual dado do Exame deseja alterar;
b) clica em Limpar, para limpar o conteúdo dos campos;
c) digita os dados corretos do exame de refração;
d) usuário clica no OK;
e) a mensagem “Registro Alterado para esse Paciente.” é exibida.

Ou:

a) usuário seleciona o paciente e o Exame.
b) clica em Excluir, para excluir o registro deste paciente.

c) confirma a exclusão.

d) a mensagem “Registro Excluído do Paciente.” é exibida.

5.3 PROTOTIPAÇÃO DE TELAS

Após a definição dos casos de uso, foram desenvolvidos os protótipos das telas do aplicativo móvel. Realizar a prototipação no desenvolvimento de um software é uma metodologia que tem como escopo analisar as ideias elaboradas, validadas ou não, com todas as condições estabelecidas. No protótipo é que passam e são definidos os elementos e onde eles devem estar situados, quais campos devem existir e quais ações serão realizadas quando o ator da ação realizar alguma interação com o sistema (PERNICE, 2016). Segundo Pernice, “[...] o protótipo é a tangibilização de uma ideia, a passagem do abstrato para o físico de forma a representar a realidade, mesmo que simplificada e propiciar validações”.

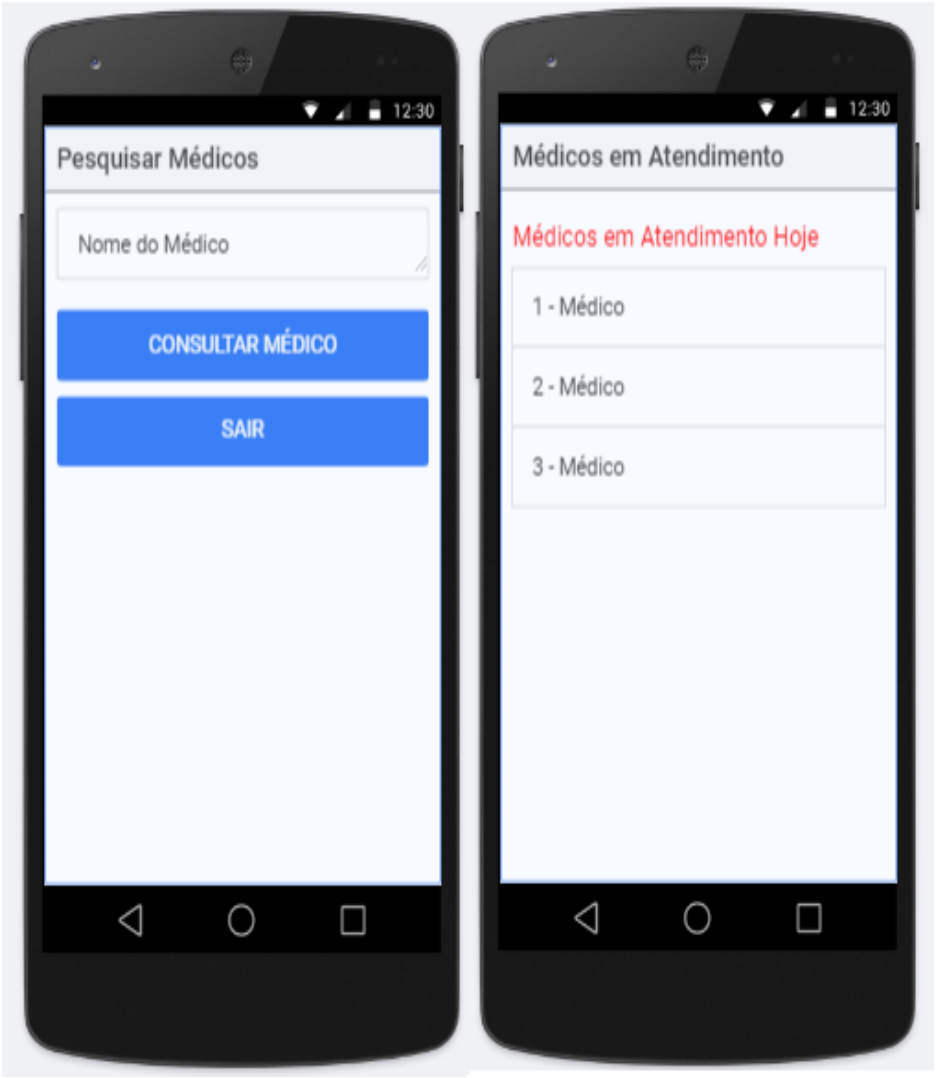
A figura 23 representa o protótipo de tela do *login* do aplicativo e o *menu* do aplicativo. O usuário do Aplicativo MS2 informa o seu usuário e a sua senha e clica no botão OK, o usuário aguarda a resposta do aplicativo que irá indicar através de um *alertdialog* se os dados estão corretos ou incorretos. Após o acesso ao aplicativo o usuário será direcionado ao *menu* iniciar do aplicativo, que contém três botões: o primeiro é Pesquisar Médicos, o segundo é Agenda e o terceiro Sair.

Figura 23 – Protótipo caso de uso CDU01

Fonte: o autor.

A figura 24 representa o protótipo de tela de listagem de médicos no aplicativo. Quando não informado o campo Nome do Médico e clicado no botão Consultar Médico, o aplicativo retorna todos os médicos cadastrados no sistema MS2 que possuem consultas agendadas no dia. Caso seja informado o médico retorna na lista apenas se o médico em questão possui ou não consulta agendada. Ainda é apresentado um *alertdialog* indicando se não há dados na consulta realizada.

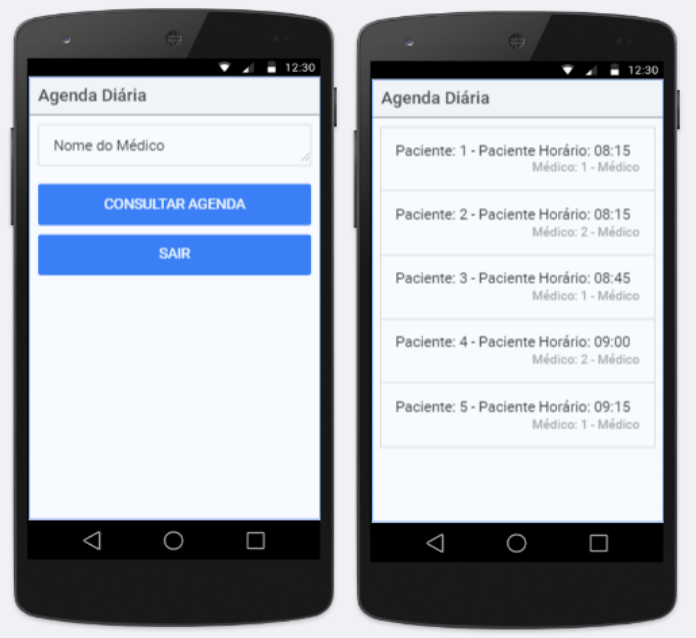
Figura 24 – Protótipo caso de uso CDU02



Fonte: o autor.

A figura 25 representa o protótipo da agenda médica no aplicativo. Quando não é informado o campo Nome do Médico e clicado no botão Consultar Agenda, retorna todos os pacientes que possuem algum horário cadastrado no sistema MS2 no dia. Caso seja informado o médico, retorna na lista apenas os pacientes que possuem consultas vinculadas aquele médico. Ainda é apresentado um *alertdialog* indicando se não há dados na consulta realizada.

Figura 25 – Protótipo caso de uso CDU03



Fonte: o autor.

A figura 26 apresenta o protótipo da tela do exame de refração que é gerado após o usuário selecionar um paciente na listagem da agenda. A tela indica o nome do paciente, horário de sua consulta e o *menu* com quatro opções: a primeira é o Auto Refrator, a segunda é a Distância Pupilar, a terceira é a Distância Naso-Pupilar e a quarta é a opção de Sair que retorna a listagem de paciente. As três primeiras opções do *menu* apresentam a mesma composição em sua tela, que é a seguinte: os campos referentes a cada exame e três botões com a opção de OK (grava os dados da tela), Limpar (limpa os dados da tela) e Excluir (exclui os registros do sistema ERP MS2).

Figura 26 – Protótipo caso de uso CDU04

Fonte: o autor.

6 DESENVOLVIMENTO

Neste capítulo do desenvolvimento do trabalho será abordado a parte de desenvolvimento dos métodos que compõem o *web service* e o aplicativo móvel de nomenclatura MS2 Aplicativo. O desenvolvimento dessas soluções, iniciou-se pelo *web service*, que é responsável pela coordenação dos dados e das requisições que são realizadas pelo aplicativo móvel.

Com as características/funcionalidades do aplicativo e os métodos do *web service* definidos, foi possível dividir essa construção em *sprints*, que são espaços de tempo definidos para a execução de cada atividade (DESENVOLVIMENTO ÁGIL, 2019), divergindo entre uma, duas, três e quatro semanas cada atividade. Elas são apresentadas a seguir.

Desenvolvimento dos verbos HTTP do *web service* REST: nesta etapa foi desenvolvido a conexão entre o *web service* e o banco de dados do sistema ERP MS2 utilizando uma biblioteca .jar (*Java Archive*) que se encarrega de realizar a conexão entre os dois sistemas, o *web service* e o sistema ERP MS2. Assim foi possível a implementação dos métodos (GET, POST, PUT e DELETE), que realizam toda a comunicação com o sistema MS2, através de consultas SQL. Foram definidos o formato do recebimento e do envio das mensagens do *web service*, que é o JSON. Essa etapa foi à que teve o seu tempo de duração mais longo com aproximadamente 4 semanas, por se tratar da construção de todo o *web service*.

Testes do *web service* REST: no desenvolvimento desta etapa foram feitos testes para apontar possíveis *bugs* no *web service.* A ferramenta utilizada foi o Postman que através de requisições HTTP testa os serviços/métodos do *web service* REST. Essa etapa teve a duração de uma semana aproximadamente.

Desenvolvimento do *login* e do *menu* principal do aplicativo móvel: essa etapa do desenvolvimento, teve como um dos objetivos a criação da tela de *login* que é composta pelos campos de usuário e senha. Após informados esses dois campos o aplicativo realiza uma solicitação HTTP ao *web service*, a fim de verificar se os mesmo estão corretos. Com a resposta da requisição do *web service* no formato JSON, é possível determinar se o usuário poderá acessar o aplicativo, ou não. Com o usuário logado no aplicativo, a primeira tela que ele tem acesso é o *menu* do aplicativo, que contém três funcionalidades: a consulta de médicos, a agenda e a opção de sair. Ainda nesta etapa foram feitas as customizações iniciais com as cores dos botões e textos e nomes de cada tela. Essa etapa teve a duração de aproximadamente 3 semanas.

Desenvolvimento da consulta de médicos do aplicativo móvel: nesta etapa foi desenvolvida a funcionalidade que permite ao usuário consultar os médicos que possuem pacientes agendados no dia. A tela possui um campo para consultar o médico, a consulta pode ser realizada informando o nome do médico ou não. Com a requisição HTTP do aplicativo ao *web service*, e recebendo a resposta no formato JSON, é gerado um novo layout no formato de lista mostrados todos os médicos que atendem a determinada consulta do aplicativo. Como no *sprint* anterior foi desenvolvida a requisição HTTP entre o aplicativo e o *web service*, foi possível diminuir o tempo de duração dessa etapa para 2 semanas aproximadamente.

Desenvolvimento da agenda dos médicos do aplicativo móvel: nesta etapa foi desenvolvida a funcionalidade que permite ao usuário do aplicativo, consultar a agenda dos médicos no dia. Na tela há um campo para informar o nome do médico em específico, ou a consulta pode ser realizada sem o nome do médico, retornando a agenda de todos os médicos. Como nas *sprints* anteriores o aplicativo realiza uma requisição HTTP ao *web service* e recebe a resposta no formato JSON, gerando um novo layout no formato de lista com os pacientes agendados, apresentados da seguinte forma: seu nome, horário e o médico ao qual esse paciente está agendado. Essa etapa teve a duração de aproximadamente 2 semanas, pois foi possível reaproveitar o código da *sprint* anterior.

Desenvolvimento do gerenciamento dos dados do exame de refração do aplicativo móvel: nesta etapa permite que o usuário realize todo o gerenciamento do exame de refração, em relação a um determinado paciente. Ao selecionar o paciente da agenda, é gerado um novo layout com um submenu das opções que o exame de refração oferece que são elas: auto refrator, distância pupilar, distância naso-pupilar e sair. Cada opção do submenu redireciona para um novo *layout* com os campos específicos a serem preenchidos. Neste layout além dos campos pré-determinados existem alguns botões com as funcionalidades de gravar, limpar, excluir e sair. Ao acessar o layout com os campos específicos, o aplicativo realiza requisições HTTP ao *web service* para consultar se o paciente já possui dados do exame de refração lançados no sistema, o paciente possuindo dados do exame de refração, os mesmos são inseridos nos respectivos campos da tela. Essa etapa teve a duração de aproximadamente 3 semanas, pois apesar de reaproveitar códigos das *sprint* anteriores, as funcionalidades de inserir e editar valores precisaram mais tempo de desenvolvimento.

Testes e correções de *bugs* do aplicativo móvel: no desenvolvimento desta etapa foram feitos testes para apontar possíveis *bugs* no aplicativo móvel. Juntamente com as correções de bugs foram realizados pequenos ajustes, como posicionamento de campos e validações dos formatos deles. Essa etapa teve a duração de uma semana aproximadamente.

6.1 RECURSOS DE HARDWARE E SOFTWARE

Nesta seção serão detalhadas a ferramenta utilizada para o desenvolvimento do software MS2 Sistema de Gestão Clínica e a ferramenta escolhida para desenvolver o aplicativo móvel e o *web service*, além de algumas tecnologias utilizadas.

6.1.1 Visual FoxPro

O sistema MS2 sistema de Gestão Clínico foi desenvolvido utilizando a ferramenta Visual FoxPro na sua oitava versão. Essa ferramenta é comercializada pela Microsoft, possuindo uma linguagem de programação própria orientada a objetos, juntamente com um banco de dados relacional integrado na própria ferramenta. Como é uma ferramenta totalmente orientada a objetos, gera uma facilidade na criação de aplicações locais e cliente/servidor.

6.1.2 Servidor de aplicação Apache Tomcat

O *web service* será hospedado em um servidor local com a aplicação *Apache Tomcat*. O *Apache Tomcat* foi criado e desenvolvido pela *Apache Software Foundation*, é um *container* de *servlets* que tem sua distribuição *open source*. Esse servidor de aplicação tornou-se uma referência para a implementação das tecnologias *servlet* e *Java Server Pages* (APACHE TOMCAT, 2019).

O *servlet* por sua vez é uma aplicação desenvolvida na linguagem de programação *Java*, que tem como base o fornecimentos de recursos adicionais para servidores *web*, ou seja, expandir sua capacidade. Já a tecnologia *Java Server Pages* por sua vez é utilizada em aplicações para *web*, permitindo o gerenciamento das informações do servidor como: controle de acessos de visitantes e manipulação de arquivos (APACHE TOMCAT, 2019).

6.1.3 JSON

Os dados manipulados entre a aplicação móvel e o *web service* utilizaram o formato JSON. O JSON é um formato compacto e simples de transmitir os dados, fundamentado em texto e padronizado. Com a existência de limitações de processamento nos aplicativos móveis, se faz necessário a utilização de um padrão leve e de rápido processamento, no qual o

JSON se encaixa perfeitamente, alcançando um funcionamento superior ao padrão XML (A. JSON, 2019).

6.1.4 SQLite

O banco de dados do aplicativo móvel para armazenamento dos dados caso não tenha internet e posteriormente realizar a sincronização será o SQLite. O SQLite implementa um banco de dados relacional que utiliza a linguagem SQL, e é uma biblioteca desenvolvida na linguagem C. Esse banco de dados realiza a leitura a gravação diretamente em disco, não possuindo um processo separado de servidor como os demais bancos SQL. Por se tratar de um banco de dados independente, veloz e alta confiabilidade ele está embutido na maioria dos *smartphones* e *tablets* (SQLITE, 2019).

6.1.5 Android Studio

O aplicativo móvel será desenvolvido no ambiente *Android* Studio. Em 2013 a Google anunciou essa plataforma. O *Android* Studio é um ambiente de desenvolvimento que facilita a programação de aplicativos em *Android*. Uma das vantagens é que através dos plug-ins existe a facilidade de integração com outras linguagens e processos (*ANDROID* STUDIO, 2019).

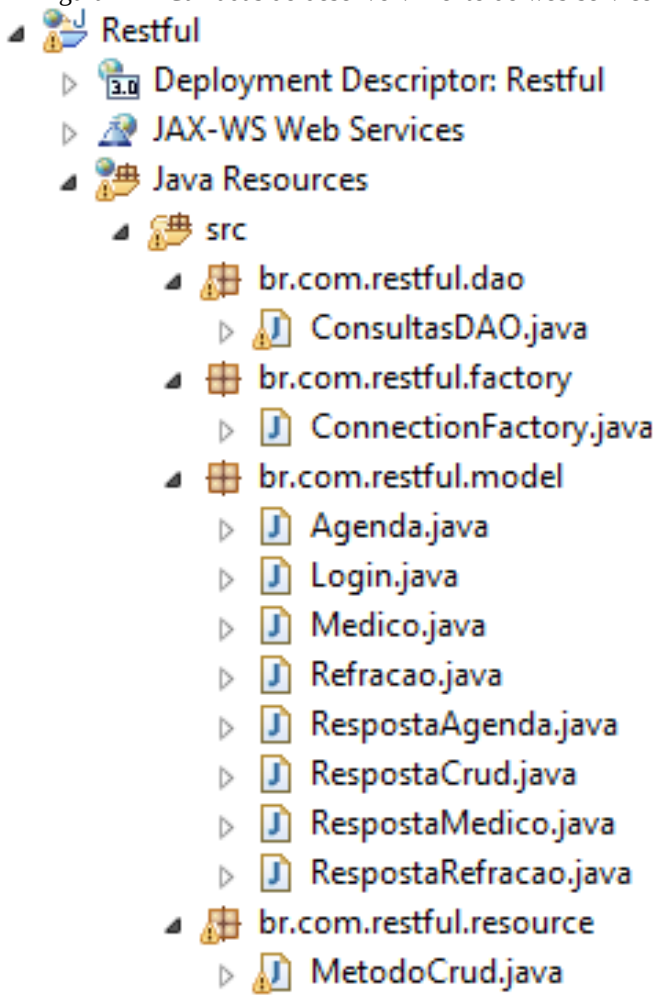
6.2 DESENVOLVIMENTO DO WEB SERVICE

Todo o desenvolvimento do *web service* foi realizado utilizando a ferramenta IDE Eclipse, na linguagem de programação Java, utilizando a API JAX-RS, que simplificou o seu desenvolvimento. Essa API possui registros que representam quais ações poderão ser executadas, que geram as classes e os artefatos do *web service*, essas classes ficam em um arquivo com a extensão .WAR (*web application archive*), através do qual passam a disponibilizar os dados aos seus consumidores, neste caso o aplicativo móvel.

Na figura 27, pode-se observar que o *web service* está dividido em quatro camadas que compõem a sua estrutura. São elas: Dao, Factory, Model e Resource. Cada camada é responsável por funcionalidades específicas dentro do *web service*. Na camada Dao é onde estão as requisições SQL do *web service* ao banco de dados do sistema ERP MS2. Dentro dessa camada está a classe ConsultasDAO, que realiza o gerenciamento dessas requisições, com consultas, alterações, inserções e remoções dos dados no sistema MS2. A Factory é a

camada onde está a conexão com o banco de dados do sistema ERP MS2. A classe ConnectionFactory que pertence a camada Factory, contém os métodos para criar, abrir e fechar a conexão com o banco de dados. A terceira camada, Model, contém os atributos dos objetos das classes. Estão presentes nesta camada as classes: Agenda, Login, Medico, Refracao, RespostaAgenda, RespostaCrud, RespostaMedico, RespostaRefracao que realizam o gerenciamento das requisições solicitadas pela classe ConsultasDAO. A quarta e última camada, Resource, é responsável por conter verbos HTTP (GET, POST, PUT e DELETE) do *web service* REST. Dentro dessa camada está a classe MetodoCrud que é responsável por realizar o gerenciamento destes verbos.

Figura 27 – Camadas do desenvolvimento do web service



Fonte: o autor.

A figura 28 está representando uma classe do *web service* que lista a agenda diária dos médicos. A classe consiste em buscar uma lista de objetos, com os pacientes que possuem horário na agenda do sistema MS2, através de uma consulta SQL.

Como no desenvolvimento do *web service* foram utilizadas as anotações da API JAX-RS com as classes @GET, @Path, @Consumes, @Produces e @PathParam. A seguir descreve-se o funcionamento de cada uma das classes.

O @GET faz com que seja feita uma requisição HTTP feita por uma URI que ligará o método listarAgenda.

@Path irá apontar para o servidor onde estão os recursos desta classe. As solicitações são feitas através de HTTP para esta URI, que manipula esse caminho indicado para acessar os serviços.

O @Consumes determina o formato de envio das mensagens para o servidor, o formato JSON.

O @Produces determina o formato JSON para enviar ao cliente o formato das mensagens do *web service*.

O @PathParam define quais parâmetros serão enviados para a URI.

Figura 28 – Código de serviço do web service

```
1  @GET
2  @Path("/agenda/{dia}")
3  @Consumes(MediaType.APPLICATION_JSON + CHARSET_UTF8)
4  @Produces(MediaType.APPLICATION_JSON + CHARSET_UTF8)
5  public RespostaAgenda listarAgenda(@PathParam("dia") String dataAgenda) {
6
7      RespostaAgenda resposta = null;
8      try {
9          resposta = consultasDAO.listarAgenda(dataAgenda);
10     } catch (Exception e) {
11         e.printStackTrace();
12     }
13     return resposta;
14 }
15
```

Fonte: o autor.

O quadro 7 apresenta todos os serviços disponíveis no *web service*, com o endereço web, o verbo HTTP (GET, POST, PUT e DELETE), o nome de cada serviço e a funcionalidade de cada serviço. Alguns serviços são acessados através de consultas, onde são passados por parâmetros algumas chaves e dados. Para realizar a inclusão e alteração é esperado uma conexão contendo como parâmetros um arquivo JSON, e seus dados são adicionados ao banco de dados MS2.

Quadro 7 – Serviços do web service

Endereço	Verbo HTTP	Nome	Funcionalidade
/login/{nome}/{senha}	GET	Login Aplicativo	Consulta se o usuário e senha informados estão presentes no sistema MS2.
/consultaRefracao/{paciente}/{médico}/{hora}/{data}	GET	Consulta Refração	Consulta se possui exame de refração informada para um determinado paciente em uma determinada hora e data com um determinado médico.
/medicoConsultaNome/{dia}/{nome}	GET	Consulta Nome do Médico	Consulta se o médico possui atendimento em uma determinada data, filtrando pelo nome do médico.
/agendaNome/{dia}/{nome}	GET	Consulta Agenda pelo Nome do Médico	Consulta a agenda dos médicos, por uma determinada data e pelo nome do médico.
/agenda/{dia}	GET	Consulta Agenda	Consulta a agenda dos médicos, por uma determinada data.
/adicionarRefracao/{paciente}	POST	Incluir Refração	Adiciona os dados do exame de refração para um determinado paciente.
/editarRefracao/{paciente}/{médico}/{data}/{hora}	PUT	Editar Refração	Edita os dados do exame de refração, conforme o paciente, a data, a hora e o médico.
/deletarRefracao/{paciente}/{médico}/{data}/{hora}	DELETE	Deletar Refração	Deleta os dados do exame de refração conforme o paciente, a data, a hora e o médico.

Fonte: o autor.

Na figura 29 é possível observar o retorno gerado pelo *web service* no formato JSON. Com essa estrutura foi implementado um *header* padrão em todos as respostas dos serviços com a *tag*, "*codRetorno"*. Ela representa o código de retorno que pode assumir dois valor, 200 para as interações de consultas, alterações, inserções e remoções que forem realizadas com sucesso no banco de dados do sistema MS2, e o valor 400, para quando apresentar erro nas interações com sistema MS2.

Figura 29 – Código de retorno do web service

```
{
 "codRetorno": 200,
 "respostaAgenda": [
    {
        "data": "09-10-2019",
        "hora": "08:00",
        "paciente": 1,
        "nomepaci": "PACIENTE 1",
        "medico": 1,
        "nomemedico": "MÉDICO 1"
    },
    {
        "data": "09-10-2019",
        "hora": "09:00",
        "paciente": 2,
        "nomepaci": "PACIENTE 2",
        "medico": 2,
        "nomemedico": "MÉDICO 2"
    }
    ]
}
```

Fonte: o autor.

O servidor de aplicação que foi utilizado foi o Apache Tomcat 7. O servidor de aplicação Tomcat é instalado no local onde está o banco de dados, facilitando a implantação de novas versões, fazendo com que não seja necessário configurar diversas funcionalidades para executar a aplicação novamente.

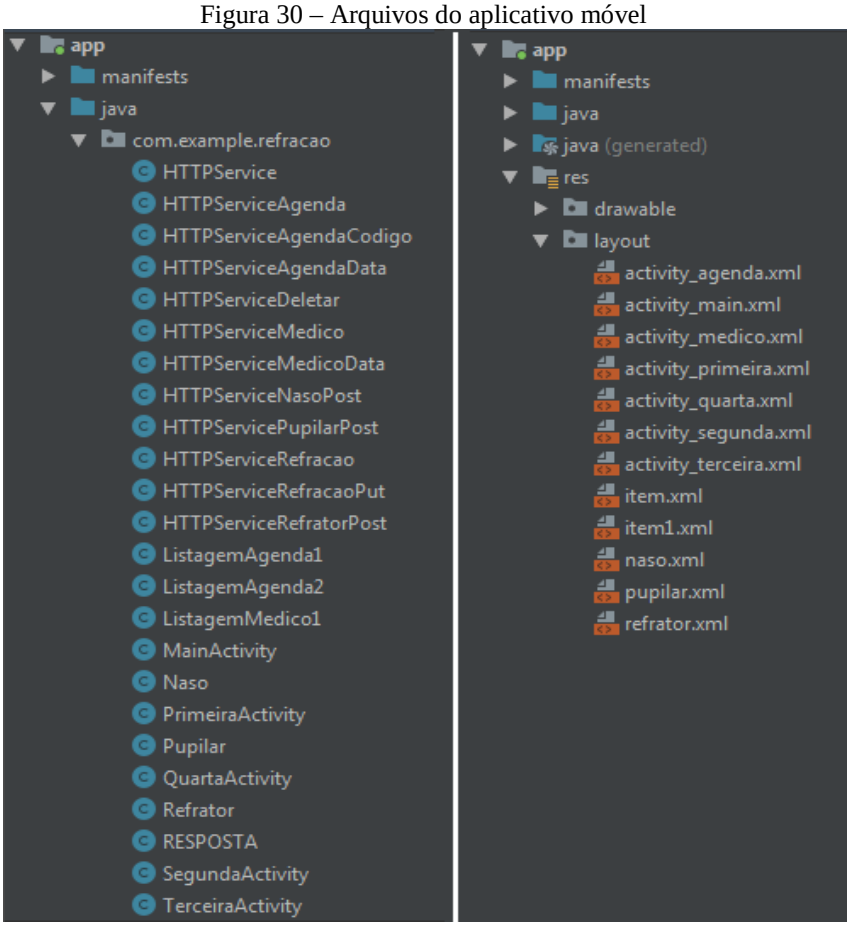
6.3 DESENVOLVIMENTO DO APLICATIVO MÓVEL

Para realizar o desenvolvimento do aplicativo móvel foi utilizado a ferramenta de desenvolvimento IDE *Android* Studio na linguagem de programação Java. A seguir serão descritas algumas rotinas e a implantação de algumas funcionalidades do aplicativo desenvolvido.

Ao trabalhar com o *Android* Studio utiliza-se uma estrutura de arquivos em .xml que facilitam a criação do layout do aplicativo. Utilizando o recurso "*drag and drop*", que consiste em arrastar os elementos e posicionar no local desejado, o editor da aplicação se encarrega de arquitetar os *layout* para outros tamanhos de tela, com o mesmo código. Existem também arquivos .java que contém funcionalidades do aplicativo, como transações entre as *activity* e conexões HTTP com o *web service* REST. Nos arquivos .java através da funcionalidade *findViewByid* é possível acessar os dados que foram delimitados no arquivo

.xml. A figura 30 apresenta os arquivos .xml e .java que foram utilizados na geração do aplicativo.

Figura 30 – Arquivos do aplicativo móvel



Fonte: o autor.

Ao definir o código fonte do Aplicativo MS2, podemos dividi-lo em três grupos: *activity*, conexões e objetos, cada grupo com suas devidas responsabilidades. Abaixo será exemplificado o funcionamento de uma classe do aplicativo para o grupo *activity*. Vale destacar que as classes pertencentes a este grupo possuem características de implementação semelhantes.

Activity: são definidas neste trabalho como as classes responsáveis por apresentarem as telas de interface do aplicativo e suas funcionalidades ao usuário, as listas geradas no aplicativo também se enquadram neste grupo.

Na figura 31 está representada a classe denominada TerceiraActivity. Ela é responsável por dispor os campos na tela através da função *onCreate*, que recebe como parâmetro o arquivo .xml, nesse caso o arquivo "*activity_terceira.xml",* que contém campos

como botões, *textview* e legendas. Nesse grupo as classes são responsáveis por realizar o tratamento de cada interação que o usuário tem com tela.

Figura 31 – Classe Activity do aplicativo móvel

```
public class TerceiraActivity extends AppCompatActivity {

    EditText edt3;
    Button b3, b4;

    @Override
    protected void onCreate(@Nullable Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_terceira);

        edt3 = findViewById(R.id.txtmedico);
        b3 = findViewById(R.id.btnconsultar);
        b4 = findViewById(R.id.btnsair);

        b3.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                boolean i = PesquisarAgenda();

                if (i == true){
                    Intent it = new Intent(TerceiraActivity.this, ListagemAgenda1.class);
                    it.putExtra("PARAM_ListagemMedico1", edt3.getText().toString());
                    startActivityForResult(it, 1);
                }
            }
        });

        b4.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                startActivity(new Intent(getBaseContext(),PrimeiraActivity.class));
            }
        });

    }
```

Fonte: o autor.

Na figura 32, ainda pertencente à classe TerceiraActivity, se observamos o método PesquisarAgenda(), que referencia outra classe do aplicativo responsável pelas conexões, ao receber o retorno independente de ter obtido sucesso, ou não na requisição realiza, na tela será exibido um *toast* com uma mensagem. E no clique botão “Consultar” na figura 32 será criado uma *intent* para acessar uma nova *activity* do aplicativo. Utilizando o método *putExtra* será possível configurar parâmetros para essa nova *activity*, porém essa funcionalidade só será implementada e ativada, se o retorno do método PesquisarAgenda() obtiver sucesso.

Figura 32 – Classe PesquisarAgenda do aplicativo móvel

```
public boolean PesquisarAgenda(){
    Date data = new Date();

    boolean lEntrou = false;

    String nomemed = edt3.getText().toString();

    if(nomemed.length() > 0){
        HTTPServiceAgendaData service = new HTTPServiceAgendaData(data);
        try {
            RESPOSTA retorno = service.execute().get();

            int Cod = retorno.getCodRetorno();

            if(Cod == 200){
                Alert("Consulta realizada com sucesso!");
                lEntrou  = true;
            }else {
                Alert("Não existem dados na consulta!");
                lEntrou = false;
            }
        } catch (ExecutionException e) {
            e.printStackTrace();
        } catch (InterruptedException e) {
            e.printStackTrace();
        }
    }
    return lEntrou;
}

public void Alert(String msg){
    Toast.makeText(this, msg, Toast.LENGTH_SHORT).show();
}
```

Fonte: o autor.

As classes que representam as activitys no Aplicativo MS2 são: MainActivity, PrimeiraActivity, SegundaActivity, TerceiraActivity, QuartaActivity, ListagemAgenda1, ListagemAgenda2 e ListagemMedico1.

Conexões: são as classes responsáveis por realizar as requisições HTTP entre o aplicativo e o *web service*.

Figura 33 – Classe de conexão do aplicativo móvel com o web service

```
public class HTTPServiceAgendaData extends AsyncTask<Void, Void, RESPOSTA> {
    public HTTPServiceMedico(Data dia) {
        this.dia = dia;
    }

    @Override
    protected RESPOSTA doInBackground(Void... voids) {
        StringBuilder resposta = new StringBuilder();
        URL url = null;
        HttpURLConnection connection = null;

        try {
            url = new URL("http://192.168.255.160:8080/Restful/ms2/agenda/" +  this.dia);
            connection = (HttpURLConnection) url.openConnection();
            connection.setRequestMethod("GET");
            connection.setRequestProperty("Accept", "application/json");
            connection.connect();

            Scanner scanner = new Scanner(url.openStream());
            while(scanner.hasNext()){
                resposta.append(scanner.next());
            }
        } catch (MalformedURLException e) {
            Log.e("LOG_TAG","Erro na criação da URL", e);
        } catch (IOException e) {
            e.printStackTrace();
        }finally {
            connection.disconnect();
        }
        return new Gson().fromJson(resposta.toString(), RESPOSTA.class);
    }
}
```

Fonte: o autor.

A figura 33, apresenta o método RESPOSTA() da classe HTTPServiceAgendaData, que realiza uma requisição HTTP ao *web service*, com o serviço Consulta Agenda apresentado no quadro 7. A classe HTTPServiceAgendaData é acionada pelo método PesquisarAgenda() da classe TerceiraActivity, que passa os parâmetros necessários para realizar a consulta da agenda dos médicos. A classe HTTPServiceAgendaData recebe como resposta do *web service* uma mensagem no formato JSON, com o código de retorno indicando se obteve sucesso, ou não obteve sucesso, ao realizar a requisição. Essa mensagem é convertida para o formato Java, para ser manipulada no aplicativo. A conversão utiliza a biblioteca de código open-source Gson, que é capaz de serializar e deserializar objetos JSON para Java e vice-versa.

Todas as classes que realizam conexões HTTP no Aplicativo MS2, são: HTTPService, HTTPServiceAgenda, HTTPServiceAgendaCodigo, HTTPServiceAgendaData, HTTPServiceDeletar, HTTPServiceMedico, HTTPServiceMedicoData, HTTPServiceNasoPost, HTTPServicePupilarPost, HTTPServiceRefracao, HTTPServiceRefracaoPut e HTTPServiceRefratorPost. O que as diferenciam são os parâmetros recebidos e enviados, os verbos HTTP (GET, POST, PUT, DELETE) e o endereço do serviço.

Objetos: no desenvolvimento deste trabalho as classes definidas como objetos são responsáveis por tornar simplificada a manipulação dos dados, realizando a ponte de comunicação entre a resposta do *web service* e o aplicativo.

A figura 34, representa a classe RESPOSTA, que tem como objetivo tratar o código de retorno das conexões que são realizadas entre o aplicativo e o *web service*. Esse código de retorno está descrito na seção 6.1 deste trabalho.

Figura 34 – Classe que trata código de retorno do web service no aplicativo

```
public class RESPOSTA {
    private int codRetorno;
    private String msgRetorno;

    public int getCodRetorno(){
        return codRetorno;
    }

    public void setCodRetorno(int codRetorno){
        this.codRetorno = codRetorno;
    }

    public int toInteger(){
        return getCodRetorno();
    }
}
```

Fonte: o autor.

Na figura 35, o método MontaCampo() que pertence a classe Pupilar recebe como parâmetro uma *string* com a resposta da requisição realizada ao *web service*. Nesse caso o serviço realizado é o Consulta Refração (Quadro 7). O objeto JSONArray se encarrega de procurar no arquivo recebido por parâmetro a *tag* principal definida como "respostaRefracao". O objeto ao encontrar a *tag* principal no arquivo realiza um laço de repetição para encontrar a *subtag* definida, e então atribui valor a um *textview* o valor encontrado na *tag*. Esse textview será mostrado ao usuário através de uma activity.

Figura 35 – Classe que converte manipula resposta do web service no aplicativo

```
public class Pupilar extends AppCompatActivity {

    protected void MontaCampo(String respsotaWebService) {

        final TextView t = (TextView) findViewById (R.id.txtod);

        try {
            JSONObject json = new JSONObject(respsotaWebService);
            JSONArray jArray = json.getJSONArray("respostaRefracao");

            for(int i = 0; i < jArray.length(); i++){
               JSONObject friend = jArray.getJSONObject(i);
               t.setText(friend.getString("dpod"));
            }
        } catch (JSONException e) {
            e.printStackTrace();
        }
    }
}
```

Fonte: o autor.

As classes que representam essas grupo são: Naso, Pupilar, Refrator e RESPOSTA.

6.4 TESTE DO APLICATIVO DESENVOLVIDO

No processo de desenvolvimento do *web service* e do protótipo do aplicativo móvel foi necessário a utilização de técnicas para a realização dos testes. A técnica utilizada foi o Desenvolvimento Orientado a Testes (TDD). Por esse projeto ter sido dividido em *sprints*, e cada *sprint* representar uma pequena parte do projeto, essa técnica de testes se adapta facilmente, pois o projeto passa a ficar mais regular, progressivo e mensurável, focando em partes específicas e aumentando as suas funcionalidades. (Koskela, 2007).

No TDD para esse projeto foi utilizado o *framework* JUnit, que suporta a linguagem de programação Java, já que o *web service* e o aplicativo utilizam essa linguagem. Nesse *framework* os testes tornaram-se automatizados e passam a ser executados continuamente. Foi definido um ciclo de teste que é composto por 5 passos: escrever o teste, o teste falhar, o teste passar, reescrever o código se necessário e no final implementar o código. A partir dessa implementação ao desenvolver o código fonte, foi possível aumentar a tomada de decisões nas funcionalidades específicas dos softwares envolvidos.

Com a aplicação dessa técnica no desenvolvimento do *web service* e do protótipo do aplicativo móvel, foi possível impedir que fossem criados e implementados códigos fontes desnecessários, tendo como ponto de partida os requisitos do sistema. As vantagens da

utilização do TDD foram que os códigos escritos realmente passam a atender a necessidade do projeto, um controle sobre mudanças feitas futuras tanto no *web service* quanto no aplicativo móvel e a credibilidade nas entregas feitas ao cliente.

Para obter um *feedback* do MS2 Aplicativo, foram elaboradas seis perguntas realizadas com o usuário responsável pelo lançamento do exame de refração no sistema ERP MS2. O usuário em questão utilizou o aplicativo durante uma semana, realizando os testes e descobrindo suas funcionalidades, essas perguntas foram respondidas através de áudios por um aplicativo de mensagens. As perguntas em questão estão relacionadas às dificuldades encontradas, facilidade no uso e se atendeu às necessidades iniciais do projeto. Com o *feedback* apresentado pelo usuário será possível realizar um remodelagem do aplicativo caso necessário. As perguntas foram:

a) questão 1: qual foi sua experiência com a interface do aplicativo?
b) questão 2: quais os benefícios na utilização do aplicativo?
c) questão 3: quais as maiores dificuldades na utilização do aplicativo?
d) questão 4: foi fácil visualizar os médicos em atendimento e a agenda de pacientes?
e) questão 5: foi possível editar, gravar e excluir os dados do exame de refração?
f) questão 6: quais sugestões de melhorias e funcionalidades para o aplicativo você propõe?

As respostas apresentadas pelo usuário que testou o aplicativo foram as seguintes:

Resposta Questão 1: O usuário relatou que o aplicativo é intuitivo e que a sua proposta atendeu a necessidade do negócio, porém relatou que apesar do aplicativo ser prático foi preciso um tempo para se adaptar e interagir com ele.

Resposta Questão 2: O maior benefício na utilização do MS2 Aplicativo é a praticidade na realização do exame de refração, pois o *tablet* ao qual foi instalado o aplicativo fica junto a sala onde é feito o exame, não precisando ter o deslocamento até o computador mais próximo para lançar os dados.

Resposta Questão 3: O usuário não apresentou nenhuma dificuldade, apenas relatou em que alguns momentos ao logar no aplicativo ele apresentava uma demora de quinze a vinte segundos.

Resposta Questão 4: A lista com os médicos e a agenda dos mesmos, foi visualizada de forma fácil e ágil, apresentando um tempo de resposta entre três e cinco segundos, o que não atrapalhou no processo de lançamentos dos dados do exame.

Resposta Questão 5: Todas as funcionalidades propostas estavam de acordo não apresentando erros. O tempo de resposta para realizar a gravação, edição e deletar os dados também foi aceitável ficando na faixa de quatro a seis segundos.

Resposta Questão 6: As sugestões apresentadas pelo usuário foram em adições de campos para filtros da agenda, inclusão de um campo para consultar outras datas e filtragem pelo horário da consulta.

7 CONCLUSÃO

Nesta seção é realizada uma síntese do trabalho realizado. Além disso, são apresentadas discussões em torno das contribuições deste trabalho e os trabalhos futuros.

7.1 SÍNTESE DO TRABALHO

Este projeto teve como objetivo principal desenvolver um protótipo de um aplicativo móvel que suportará a coleta dos dados de um exame clínico de refração que se comunicará com o ERP MS2 através de um *web service.*

Com a aplicação dos estudos realizados sobre a integração entre sistemas e plataformas, foi possível desenvolver um aplicativo móvel capaz de suportar a coleta de dados de exames de refração, e integrá-lo a um ERP através de um *web service* desenvolvido especialmente para tal. O *web service* REST desenvolvido neste trabalho possibilitou a almejada integração entre o aplicativo móvel e o banco de dados do sistema Ms2.

A linguagem de programação utilizada tanto para o desenvolvimento do *web service* quando do aplicativo móvel foi *Java*. Esta linguagem foi aplicada às duas plataformas por se tratar de uma linguagem *open source,* pois esta tecnologia tem a capacidade de adaptar-se a diferentes tipos de plataformas.

A plataforma escolhida pelo acadêmico para o desenvolvimento do aplicativo móvel deste trabalho foi a *Android* Studio, devido ao grande número de usuários que utilizam esta plataforma e levando em consideração também o cliente em questão da empresa Sartor Assessoria e Comércio de Informática LTDa já estar habituado a trabalhar com esta plataforma em seu cotidiano.

O que se constata através do caso de uso é que a integração proposta entre os sistemas atende a necessidade de um processo mais ágil e prático exigido pelo cliente da empresa Sartor Assessoria e Comércio de Informática LTDa. Anteriormente era preciso imprimir o resultado do exame, levá-lo até outro setor e então lançar os dados no sistema, devido ao equipamento do exame de refração não poder sofrer interferências. Agora o processo de lançamento de dados pode ser feito dentro da própria sala de exames com o uso do aplicativo que se integra com o base de dados, onde todos os dados/informações dos procedimentos são inseridos, atualizados ou removidos.

Visto desta maneira o aplicativo atrelado ao *web service* apresentou-se como uma solução apropriada ao que se pretendia, além de ser uma solução simples e mais adequada para as exigências da sociedade moderna assim como a teoria de Sadagi (2013) propõe.

7.2 CONTRIBUIÇÃO DO TRABALHO

Os estudos realizados permitiram elaborar uma relevante fundamentação teórica nos capítulos II Sistema de Informação, III Aplicativos Móveis e IV Integração entre Sistemas, a qual pode contribuir em diversos tipos de projetos.

O presente trabalho contribui com a disponibilização de um aplicativo que tornam o ERP "responsável pela informação" e não mais apenas "dono da informação". Assim seus dados ficam mais disponíveis onde precisam estar mais disponíveis. Além disto, o aplicativo promove uma disseminação de informações para todos os setores da empresa.

O ERP teve que ser modificado para permitir a integração com o aplicativo móvel através de *web services*. Anteriormente o sistema MS2 não disponibilizava integração com outros sistemas, nem mesmo com dispositivos móveis. Ao término deste trabalho de conclusão de curso é possível dizer que o sistema MS2 começa a integrar-se com outros sistemas e dispositivos móveis.

Os usuários da clínica agora dispõem de um aplicativo capaz de comunicar-se com o ERP, que através do uso do *Tablet* com o aplicativo instalado promove a mobilidade requerida pelo cliente, além de gerar credibilidade aos dados, já que são lançados em tempo real para o aplicativo, interligando os setores da clínica e promovendo a troca de informações entre o sistema e os usuários. Segundo as características que Lee (2005) definiu para um aplicativo móvel que são: portabilidade, usabilidade, funcionalidade e conectividade, o aplicativo desenvolvido apresenta tais características defendidas pelo autor.

7.3 TRABALHOS FUTUROS

Como sugestão para trabalhos futuros propõe- se adicionar novas funcionalidades ao aplicativo, tais como a leitura do cupom impresso do exame através da captura de uma foto, tornando-o mais completo, e podendo ser aproveitado em outros setores da organização. Além disto, também pode-se desenvolver a disponibilização desse aplicativo para outros SO, utilizando as ferramentas *React Native* ou *Flutter* que foram apresentadas nesse trabalho.

Como ganhos futuros para a empresa Sartor Assessoria e Comércio de Informática LTDa, considera-se o desenvolvimento de novos aplicativos para integrar-se aos demais sistemas da empresa, como por exemplo aplicativos relacionados a força de venda e controle de rotas de entregas.

REFERÊNCIAS

AALST, W. V. D.; MYLOPOULOS, J.; SADEH, N. **Advanced information systems engineering workshops**. 2011.

A. JSON, **Introdução. Introdução a Json**. Disponível em: https://www.json.org/json-pt.html. Acesso em: 01 ago. 2019.

ANDROID. **Android interfaces and architecture**. Android Open Source Project. 2019. Disponível em: <https://source.Android.com/compatibility>. Acesso em: 27 abr. 2019.

ANDROID STUDIO. Disponível em:<https://developer.Android.com/>. Acesso em: 30 ago. 2019.

APACHE TOMCAT. Disponível em: <http://tomcat.apache.org>. Acesso em: 26 jun. 2019.

APPLE INC. **About the iOS technologies**. 2019. Disponível em: <https://developer.apple.com/documentation/>. Acesso em: 28 abr. 2019.

APPLE. **iOS 12**. Disponível em: <https://www.apple.com/br/ios/ios-12/>. Acesso em: 26 abr. 2019.

BITENCOURT, Ariovaldo Carlos. **Proposta de interface de integração entre sistema CAD e ERP**. Graduação do Curso de Sistemas de Informação, Universidade de Caxias do Sul, 2013.

CAIÇARA JÚNIOR, Cícero. **Sistemas integrados de gestão ERP**: uma abordagem gerencial. 2. ed. Curitiba: Intersaberes, 2015.

Carneiro, A. (2013). **Web services em aplicações Android e ios**. Disponível em: <https://www.devmedia.com.br/web-services-em-aplicacoes-Android-e-ios/28901>. Acesso em: 18 ago. 2019.

DA SILVA, Eduardo de Santana. **Integração de sistemas legados e mobile utilizando uma API Restful.** Trabalho de Conclusão de Curso em Engenharia de Computação, Universidade de Araraquara, 2016.

DESENVOLVIMENTO ÁGIL. Disponível em: <http://www.desenvolvimentoagil.com.br/scrum/>. Acesso em: 19 ago. 2019.

DE SORDI, J.; MARINHO, B. Integração entre sistemas: análise das abordagens praticadas pelas corporações brasileiras. RBGN **Revista Brasileira de Gestão de Negócios**, América do Norte, 923 07 2007. Disponível em: <https://rbgn.fecap.br/RBGN/article/download/75/228>. Acesso em: 18 abr. 2019.

EL-KASSAS, W. S. et al. Taxonomy of cross-platform mobile applications development approaches. **Ain Shams Engineering Journal**, 2015. ISSN 2090-4479. Disponível em: <https://www.sciencedirect.com/science/article/pii/S2090447915001276>. Acesso em: 27 abr. 2019.

FACEBOOK. (2019a). **React native official webpage**. Disponível em: <https://facebook.github.io/react-native/>. Acesso em: 05 ago. 2019.

FACEBOOK. (2019b). **React native hot reloading**. Disponível em: <http://facebook.github.io/react-native/blog/2016/03/24/introducing-hot-reloading.html>. Acesso em: 05 ago. 2019.

FONSECA, J. J. S. **Metodologia da pesquisa científica**. Fortaleza: UEC, 202. Apostila.

GOMES, D. A. **Web services SOAP em Java**: guia prático para o desenvolvimento de web services em Java. Novatec, 2010.

GONÇALVES, André Luz. **Desenvolvimento de um aplicativo Android utilizando banco de dados não-relacional para organização e controle de presença de um time de futebol**. Trabalho de conclusão de curso Ciência da Computação. Universidade Federal do Rio Grande do Sul, 2016. Disponível em: <https://www.lume.ufrgs.br/bitstream/handle/10183/150930/001009684.pdf?sequence=1>. Acesso em: 25 set. 2019.

GOOGLE. (2019a). **Android**. Disponível em: <https://www.Android.com/>. Acesso em: 26 abr. 2019.

GOOGLE. (2019b). **Flutter API documentation**. Disponível em: <https://docs.Flutter.io/>. Acesso em: 06 ago. 2019.

GOOGLE. (2019c). **Flutter technical overview**. Disponível em: <https://Flutter.io/technical-overview/>. Acesso em: 06 ago. 2019.

HEARD, P. 2019. **React native architecture**. Disponível em: <https://www.logicroom.co/react-native-architecture-explained/>. Acesso em: 05 ago. 2019.

HEITKOTTER, H.; HANSCHKE, S.; MAJCHRZAK, T. A. Evaluating cross-platform development approaches for mobile applications. In: **Web information systems and technologies**. Springer, 2013. p. 120-138. Disponível em: <http: //link.springer.com/chapter/10.1007/978-3-642-36608-6_8>. Acesso em: 28 abr. 2019.

HYPÓLITO, Christiane Mendes. **Sistemas de gestão integrada**: conceitos e considerações em uma implantação. Minas Gerais, MG, 1999. 13. Tese de Mestrado, Escola Federal de Engenharia de Itajubá. Disponível em: <http://www.abepro.org.br/biblioteca/ENEGEP1999_A0357.pdf>. Acesso em: 13 mar. 2019.

IDC. **Smartphone market share**. 2019. Disponível em: <https://www.idc.com/promo/smartphone-market-share/os>. Acesso em: 27 maio 2019.

JAKL, M. **REST representational state transfer**. Vienna, 2005.

JUNIOR, Jauri da Cruz. **Solução multiplataforma para smartphone utilizando os frameworks Sencha Touch e PhoneGap integrado à tecnologia web service Java**. Trabalho de Conclusão de Curso de Análise e Desenvolvimento de Sistemas, Fundação Educacional do Município de Assis, 2014.

JUNTUNEN, Antero; JALONEN, Eetu; LUUKKAINEN, Sakari. HTML5 in mobile devices: drivers and restraints. In: **46th Hawaii International Conference on System Sciences**, 2013.

KALIM, Martin. **Java web services implementando**. Tradução: Raquel Marques Primeira Edição. Editora: Alta Books, 2010.

KARL Kurbel, T. S. Integration issues of information engineering based i-case tools. **Working Papers of the Institute of Business Informatics**. 1994.

KOSKELA, L. **Test Driven: TDD and Acceptance TDD for Java Developers**. São Paulo. Manning Publications, 2007.

LAUDON, Kenneth; LAUDON, Jane. **Sistemas de informação gerenciais**. Trad. Luciana do Amaral Teixeira. Rev. Belmiro Nascimento João. 9. ed. São Paulo. Pearson Prentice Hall, 2010.

LAUDON, Kenneth; LAUDON, Jane. **Sistemas de informação gerencial**: administrando a empresa digital. São Paulo: Prentice Hall, 2004.

LECHETA, Ricardo R. **Google Android**: aprenda a criar aplicações para dispositivos móveis com Android SD. São Paulo: Novatec, 2009.

LEE, Valentino; SCHNEIDER, Heather; SCHELL, Robbie. **Aplicações móveis**: arquitetura, projeto e desenvolvimento. São Paulo: Pearson Makron Books, 2005.

MARTINS, Victor Manuel Moreira. **Integração de sistemas de informação**: perspectivas, normas e abordagens. 2005. 218 f. Dissertação (Mestrado). Universidade do Minho Guimarães. Disponível em: <https://repositorium.sdum.uminho.pt/bitstream/1822/5657/3/tese_mestrado_victor_martins_2005.pdf>. Acesso em: 18 abr. 2019.

MARTINY, Ismael. **Proposta de um módulo de business intelligence compatível com dispositivos móveis para o software de gestão elementare**. Trabalho de Conclusão de Curso em Sistemas de Informação, Universidade de Caxias do Sul, 2014.

ROMÃO, Roni de Lima. **Desenvolvimento de uma solução para integração de dados hospitalares utilizando dispositivos móveis**. Trabalho de Conclusão de Curso em Ciências da Computação, Universidade de Caxias do Sul, 2011.

ROSA, David Jonatan da; FAVARO, Everton Niotti. **Desenvolvimento de um aplicativo móvel para food service utilizando a plataforma Android**. Trabalho Conclusão do Curso de Graduação em Tecnologias da Informação e Comunicação do Centro de Araranguá da Universidade Federal de Santa Catarina, 2018. Disponível em: <https://repositorio.ufsc.br/bitstream/handle/123456789/192039/TCC_DeividJonatandaRosa_EvertonNiottiFavaro.pdf?sequence=1&isAllowed=y>. Acesso em: 25 set. 2019.

SILVA, Firmino Oliveira da. **Integração de sistemas e plataformas como solução para a gestão da informação de clientes**. 2004. 215 f. Dissertação (Mestrado). Universidade do Porto. Disponível em: <https://repositorio-aberto.up.pt/bitstream/10216/11378/2/Texto%20integral.pdf>. Acesso em: 18 abr. 2019.

PAPAJORGJI, P. **Automated enterprise systems for maximizing business performance**. 1 ed. Hershey, PA: IGI Global, 2015. ISBN 978-1-4666-8841-4.

PERNICE, K. (2016). **Ux prototypes: Low fidelity vs. high fidelity**. Disponível em: <https://www.thoughtworks.com/insights/blog/nosql-databases-overview>. Acesso em 12 ago. 2019.

SADAGI, I. et. Al. **Análise de bibliotecas para web services no desenvolvimento em smartphones baseado no sistema operacional Microsoft Windows Phone**. 2013. Disponível em: <http://lyceumonline.usf.edu.br/salavirtual/documentos/2455.pdf>. Acesso em: 27 abr. 2019.

SENCHA TOUCH. Disponível em: <https://www.sencha.com/products/touch/>. Acesso em: 28 abr. 2019.

SOUZA, César Alexandre de; SACCOL, Amarolinda Zanela (Org.). **Sistemas ERP no Brasil**: teoria e casos. São Paulo: Atlas, 2003.

SQLITE. Disponível em: <https://www.sqlite.org/index.html>. Acesso em: 01 ago. 2019.

TANENBAUM, Andrews S.; STEEN, Maarten van. 1944. **Sistemas distribuídos**: princípios e paradigmas. Trad.: Arlete símile Marques; Rev. Tec.: Wagner Zucchi. 2. ed. São Paulo. Pearson Prentice Hall, 2007.

WILDE, Erik; PAUTASSO, Cesare. **REST**: from research to practice. Springer, 2011.

ZANCUL Eduardo Senzi, GUERRERO Vander, ROZENFELD Henrique, OLIVEIRA Cristiano Bevitori M. **Análise das abordagens de integração entre sistemas PDM e ERP**. Núcleo de Manufatura Avançada - USP. 2000. Artigo. Disponível em: <https://pt.scribd.com/document/386671562/Integracao-Entre-Sistemas-PDM-e-ERP>. Acesso em: 25 abr. 2019.

ZWICKER, R.; SOUZA, C. A. **Sistemas ERP**: conceituação, ciclo de vida e estudos de casos comparados. In: SOUZA, Cesar Alexandre e SACCOL, Amarolinda Zanela (Org.). Sistemas ERP no Brasil: teoria e casos. São Paulo: Atlas, 2003.