PROSPECTUS
ISSN 2674-8576 Gestão e Tecnologia

Artigo Original

YELLOW LIST: APLICATIVO ANDROID PARA LOCALIZAR PRESTADORES DE SERVIÇO UTILIZANDO GPS E INTEGRAÇÃO COM REDES SOCIAIS

YELLOW LIST: ANDROID APPLICATION TO LOCATE SERVICE PROVIDERS USING GPS AND INTEGRATION WITH SOCIAL NETWORKS

André Felipe Angeloni Rodrigues[1]; Fábio Stefanini da Silveira[1], Mateus Guilherme Fuini[2]

1- Graduados em Gestão da Tecnologia da Informação pela FATEC de Itapira "Ogari de Castro Pacheco"; 2- Coordenador e docente da FATEC de Itapira "Ogari de Castro Pacheco"

Contato: fabssilveira@gygatech.com.br

RESUMO

No projeto "Yellow List" propõe-se o desenvolvimento de uma aplicação *Android*, visando agilizar a busca de prestadores de serviço de uma maneira mais refinada do que pesquisar no Google e mais eficiente do que perguntar para outras pessoas. Para isso, o aplicativo levará em conta a localização atual do usuário, utilizando o GPS do smartphone, e indicará profissionais que estão próximos para poder atender rapidamente o chamado solicitado. Isso também contribui para que seja uma maneira dos prestadores de serviço poderem divulgar suas habilidades. O usuário terá a opção de fazer a autenticação no aplicativo utilizando as redes sociais: Facebook ou Google+. Caso queira, também será possível realizar um cadastro simples sem vincular as redes sociais, porém o usuário ficará um pouco mais restrito para utilizar todas as opções da plataforma. Para gerar maior confiança, o aplicativo terá a opção de o usuário qualificar o serviço prestado, com o intuito de contribuir com os outros usuários e manter a excelência nos serviços prestados.

Palavras-chave: Aplicativo Android. Prestador de Serviço. Rede Social. Geolocalização.

PROSPECTUS
ISSN 2674-8576 Gestão e Tecnologia
Artigo Original

ABSTRACT

The "Yellow List" project proposes the development of an Android App, designed to accelerate the search of service providers through an advanced filter process. This process will be faster than Google & will be more efficient than looking for the right professional to talk to. The App will use the GPS from the smartphone, consider the current location of the user and will indicate nearby professionals who can respond quickly to the requested call. This App also helps service providers share their abilities. The user will have the option to authenticate in the App using social networks: Facebook or Google+, the user can simply register without linking to social networks, but the user will be a little more restricted to use all the options in the platform. To create greater confidence, the App will have an option to write a review of the service provided or contribute to what other users discussed thereby maintaining excellence in the services provided.

Keywords: Android App. Service Provider. Social Network. Geolocation.

INTRODUÇÃO

Atualmente, não podemos mais ignorar que o uso da tecnologia está mudando a maneira como levamos nossas vidas. Estamos cercados por dispositivos eletrônicos dos mais variados tipos, capazes de desempenhar as mais variadas funções (DEITEL et al., 2015).

Temos redes de dados interligando estes dispositivos e permitindo uma troca de informações em tempo real praticamente em todos os lugares que frequentamos. A informação circula de uma maneira muito mais rápida e eficiente e podemos ter acesso a tudo isso quando e onde desejarmos ou precisarmos.

Um dos dispositivos mais reconhecidos e que está no centro de toda essa revolução é o *smartphone*. Hoje não mais apenas um aparelho de telefone celular capaz de permitir a comunicação por voz livre de fios ou outras barreiras físicas, mas tornou-se um dispositivo computacional complexo que permite cada vez mais funções e se integra totalmente nesta infindável rede de dispositivos, trazendo um mundo de informações para as nossas mãos.

> O aumento do uso de *smartphones* no Brasil tem impactado diretamente o cotidiano dos brasileiros: 42% dos usuários de *smartphones* acessam diariamente a Internet a partir de seus dispositivos móveis; 80% deles pesquisam um produto no dispositivo antes de comprar; e 31% fazem compras a partir do dispositivo [*Google Inc*. a 2012]. Essa nova forma de executar atividades comuns do dia a dia influencia o relacionamento de empresas com

> seus clientes e também tem potencial para influenciar os relacionamentos governo-governo, governo-empresa e governo-cidadão. (ANDRADE; AGRA; MALHEIROS, 2012, p 781).

Eles permitem comunicação por voz, texto ou imagem, permitem acesso a mídias de interação social como *Facebook*, *WhatsApp* e outras, acesso a *internet* para busca de informações, compra de produtos ou serviços e muito mais. Todo esse conteúdo está à disposição do usuário sempre que ele precisar.

Hoje é muito comum utilizar o *smartphone* para a busca de produtos que se deseja comprar. Procurar por lojas mais recomendadas e por menores preços nos mesmos produtos. Existem muitas opções de aplicativos para atender a essa necessidade.

Quando pensamos no ramo da prestação de serviços, esse conceito já é um pouco diferente. Existem alguns aplicativos para localização de serviços, mas em geral apenas em algumas áreas de atuação mais específicas, como por exemplo para o ramo hoteleiro e viagens.

> O uso de dispositivos móveis possibilita o acesso a serviços e informações a qualquer momento, por meio de redes sem fio e de diversos recursos, como: texto, voz, vídeo, internet, GPS, câmera, música e televisão. Estes recursos integrados podem melhorar significativamente a prestação de serviços (ANDRADE; AGRA; MALHEIROS, 2012, p 780).

Para os serviços mais comuns do dia-a-dia, como eletricistas, encanadores, chaveiros, pintores, mecânicos entre outros, as opções são bem menores. Temos alguns sites buscadores que retornam resultados para essas consultas, mas em geral esses resultados são confusos e pouco relevantes, pois podem mostrar profissionais dispersos por uma área geográfica muito diversa, ou então com informações desatualizadas.

Na maioria das vezes que acontece uma emergência e precisa-se entrar em contato com algum profissional que possa resolver o problema, as pessoas não sabem quem contratar; e para aquelas que tem alguém para chamar, se o prestador de serviço não puder atendê-la, acaba ficando sem saber uma outra alternativa. Para tentam suprir isso, as pessoas têm o hábito de procurar na *internet* ou pedir indicação para algum conhecido, só que quase nenhum desses prestadores de serviços estão com anúncio na *internet*, e quando se trata de cidades pequenas, os buscadores indicam empresas de outras cidades maiores da região, ou seja, tornando a busca irrelevante (AMORIM; SANTOS, 2015).

A utilização de um aplicativo para dispositivos móveis pode tornar as consultas mais fáceis e possibilitar resultados mais relevantes, pois estes dispositivos possuem funções como GPS, que podem ajudar a filtragem dos resultados por região geográfica, por exemplo.

É possível reduzir a área de busca e tornar os resultados mais úteis, principalmente em cidades menores. Assim os profissionais locais terão preferência nos

resultados em relação a profissionais de outras cidades que não fazem parte de onde a pesquisa foi iniciada (AMORIM; SANTOS, 2015).

Para melhorar esse gargalo entre a população e o prestador de serviço, o aplicativo *Yellow List* pretende facilitar a busca desses profissionais. Por exemplo: A pessoa está na rua num domingo à tarde e acidentalmente tranca o carro com a chave dentro, para resolver esse problema, ela precisará entrar em contato com um chaveiro, aí é só acessar o aplicativo que ele irá localizar qual a posição que a pessoa se encontra e indicar chaveiros que estão próximos ao local para poder resolver o problema.

O aplicativo possibilitaria a integração que os *smartphones* têm com as mídias sociais e a familiaridade dos usuários para permitir um poderoso sistema de indicação e recomendação de profissionais que possuem qualidade, melhorando continuamente com o retorno dos resultados pesquisados.

> Os acessos a serviços de instituições financeiras e redes sociais, por exemplo, podem ser facilitados pelo uso de aplicativos que são executados em aparelho celular. Devido a esta evolução, um aparelho celular se transformou em uma oportunidade de entretenimento, acesso a informação e solução de problemas, integrando-se assim ao cotidiano das pessoas e facilitando diversas tarefas do dia a dia (SILVA; SANTOS, 2014, p 162).

Este trabalho visa propor o desenvolvimento de uma aplicação para dispositivos móveis com o sistema *Android*, que possa catalogar profissionais e empresas prestadoras de serviço, inicialmente em uma pequena cidade do interior, para que com o uso das características de geolocalização e integração social desses dispositivos os resultados das consultas possam ser otimizados e a localização dos prestadores de serviço seja facilitada e útil para as necessidades dos usuários da aplicação.

A aplicação será de distribuição gratuita e fará uso da integração com a plataforma do *Facebook* e *Google*, tornando o uso bem familiar e confortável para os usuários. Será feita uma divulgação para encontrar prestadores de serviço interessados em divulgar seus serviços na plataforma. Essa será uma forma de demonstrar seus serviços para um enorme público de clientes em potencial e adicionar mais uma opção de uso para esses dispositivos tão comuns hoje em dia.

PROSPECTUS
ISSN 2674-8576 Gestão e Tecnologia
Artigo Original

METODOLOGIA

Tipo de Pesquisa

Trata-se de um estudo experimental, cujas etapas de pesquisa iniciaram "pela formulação exata do problema e das hipóteses, que delimitam as variáveis precisas e controladas que atuam no fenômeno estudado" (GERHARDT; SILVEIRA, 2009, p. 36). O projeto também se enquadra na categoria de pesquisa aplicada e tecnológica (SOUZA et al., 2013; VALERIANO, 1998; CERVO; BERVIAN, 1983), pois se busca a operacionalização de um determinado sistema, a manipulação de tratamentos de dados e a materialização do aplicativo que se utiliza da Plataforma Android.

A revisão bibliográfica narrativa foi elaborada a partir de livros, artigos científicos de periódicos e material disponibilizado pelas bibliotecas virtuais online. Os descritores utilizados para a busca foram: Aplicação Móvel, Plataforma *Android*, *Web Service*, Aplicações Moveis e o Mercado de Serviços, Prestadores de Serviço, Geolocalização, *Ranking* por Serviço Prestado, API do *Facebook*, API do *Google*+. Os trabalhos incluídos foram publicados entre os anos de 2006 a 2016 no idioma português e inglês. Em relação aos aspectos éticos, as normas de autorias foram respeitadas sendo que todas as obras utilizadas têm seus autores referenciados e citados de acordo com a ABNT/NBR 6023/2002 e NBR 10520/2002.

Características do Aplicativo

O Aplicativo proposto neste trabalho chama-se *Yellow List,* e trata-se de um aplicativo para dispositivos móveis da plataforma *Android*, que permite a localização fácil de prestadores de serviços, de acordo com a necessidade pesquisada pelo usuário e dentro de uma área geográfica que permita seu rápido atendimento.

Serão utilizados os sistemas de geolocalização disponíveis na plataforma *Android* e nos dispositivos suportados, para verificar com precisão a posição geográfica do usuário e então realizar filtros para garantir que os resultados devolvidos pela aplicação são próximos e viáveis para atendimento de uma possível solicitação.

A aplicação utilizará a API do *Facebook* e do *Google* para garantir uma experiência de autenticação segura e familiar, garantindo uma sensação agradável e de confiança para os usuários. Será disponibilizada também uma opção de *login* interna com um cadastro simplificado para os usuários que escolham não utilizar as opções anteriores.

A integração com o *Facebook* e *Google* permitirá que os usuários publiquem recomendações dos prestadores de serviço encontrados na aplicação para suas listas de amigos e assim divulguem os profissionais recomendados e o próprio aplicativo.

Será disponibilizado um sistema de qualificação para os serviços prestados e os profissionais que os realizaram. Essa qualificação será analisada pelo aplicativo e

influenciará os resultados das pesquisas realizadas, possibilitando que os usuários participem ativamente na melhoria da qualidade dos resultados retornados pela aplicação.

As informações sobre os prestadores de serviços serão colocadas em um banco de dados que será mantido pela equipe do aplicativo. Os profissionais ou empresas prestadoras de serviços interessadas em serem divulgadas no aplicativo terão seus dados analisados para garantir a veracidade das informações e então serão cadastradas no aplicativo.

Os processos de geolocalização, a qualificação realizada pelos usuários e a integração com as mídias sociais são os diferenciais para garantir a qualidade das pesquisas no aplicativo, permitindo resultados relevantes e que possam solucionar a necessidade de serviço do usuário da aplicação.

Requisitos do sistema/linguagem operacional

O Aplicativo será desenvolvido para plataforma *Android*, que possui várias características específicas que serão abordadas a seguir.

Linguagem *Java*

Os aplicativos desenvolvidos para *Android* são escritos utilizando a linguagem *Java*, que possui a característica de ser orientada a objetos e possuir um enorme acervo de bibliotecas de classe que facilitam o desenvolvimento de aplicações. Segundo (DEITEL *et al*., 2015, p 05), “Essa linguagem foi uma escolha lógica para a plataforma *Android*, pois é poderosa, gratuita, de código-fonte aberto e milhões de desenvolvedores já a conhecem.”

O fato da linguagem *Java* ser de código aberto a torna alinhada com o ambiente previsto para a plataforma *Android*, pois o próprio sistema operacional segue essa proposta. Isso garante o acesso a diversas bibliotecas e ferramentas que auxiliam o desenvolvimento, incluindo as APIs do *Facebook* e do *Google* que serão utilizadas no aplicativo.

A escolha da linguagem *Java* é uma característica da Plataforma *Android*, que a utiliza como base para o desenvolvimento de aplicações compatíveis. Embora existam outras linguagens e ferramentas que permitem o desenvolvimento de aplicações simplificadas para o ambiente *Android*, a linguagem *Java* é a escolha oficial da *Google* para o seu ambiente e por isso também é mais recomendada para o suporte a todos os recursos disponíveis na plataforma (DEITEL *et al*., 2015).

Android Studio

Para realizar o desenvolvimento de aplicações ou qualquer projeto de *software*, são utilizadas ferramentas e bibliotecas que auxiliam o desenvolvedor independente da linguagem de programação escolhida. Essas ferramentas podem ser simples como um editor de texto ou então muito complexas, capazes de verificar erros de sintaxe nos comandos e até sugerir alterações para os desenvolvedores.

Uma das ferramentas mais utilizadas hoje em dia pelos programadores são chamadas de IDEs que são compostas por várias pequenas ferramentas agrupadas em um grande e complexo *software* capaz de auxiliar os programadores em quase todas as etapas do desenvolvimento de uma aplicação. De acordo com (SILVA; SANTOS, 2014), a IDE ou ambiente de desenvolvimento integrado fornece ferramentas que auxiliam o desenvolvedor a criar aplicativos utilizando uma das linguagens de programação disponíveis no mercado.

Nelas o código pode ser escrito e verificado quanto a erros de sintaxe nos comandos, pode ser executado em modo de depuração para verificar erros em tempo de execução e muitas outras funções de teste são permitidas. Normalmente as IDEs são especificas para uma linguagem ou plataforma de desenvolvimento apenas, mas há IDEs multiplataforma, que suportam o desenvolvimento em várias linguagens diferentes na mesma ferramenta.

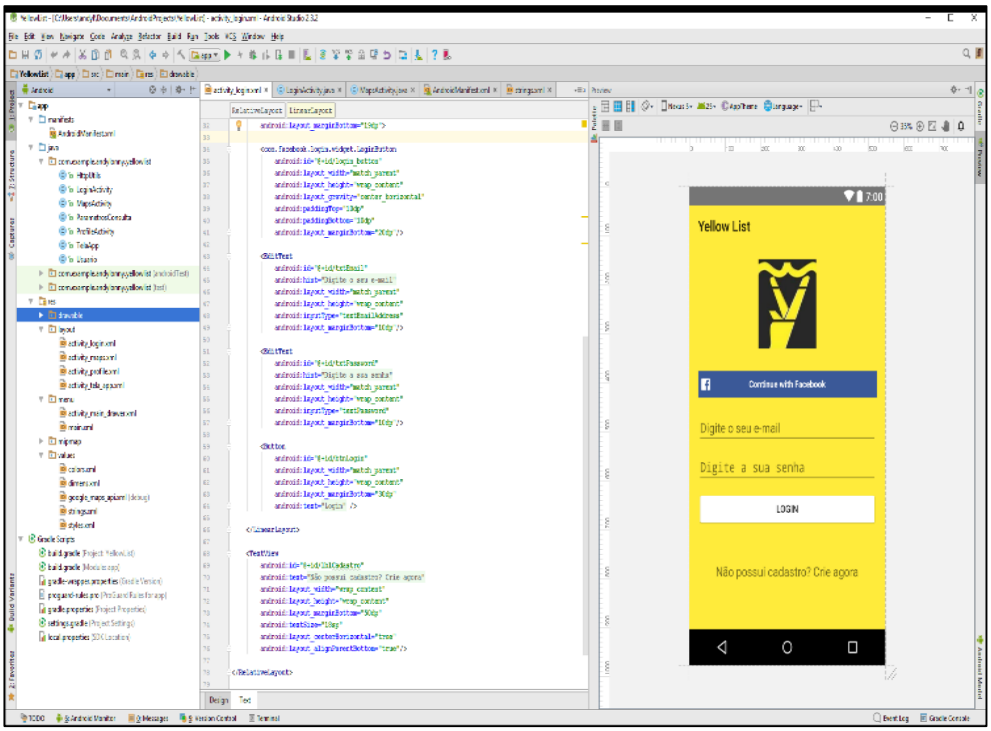
Originalmente para a plataforma *Android* a IDE utilizada para desenvolvimento era o *Eclipse* da *Eclipse Foundation*, uma organização dedicada ao desenvolvimento de *software open-source*. No *Eclipse* eram instaladas as bibliotecas ou SDK do *Android* e alguns *plugins* que permitiam o desenvolvimento de aplicações para Android. O *Eclipse* originalmente é utilizado para desenvolvimento Java e funcionava como uma opção também para a plataforma *Android*.

Atualmente isso mudou, a *Google* encerrou o suporte ao *Eclipse* e hoje a IDE oficial para desenvolvimento *Android* é o *Android Studio*[1]. Ele é mantido pela *Google*, utilizando de várias ferramentas para melhorar a produtividade dos desenvolvedores e garantir compatibilidade com os novos recursos disponibilizados regularmente para a plataforma.

[1]https://developer.android.com/studio/intro/index.html

PROSPECTUS
ISSN 2674-8576 Gestão e Tecnologia
Artigo Original

Figura 1 - Tela IDE *Android Studio*.



Fonte: Capturada em junho de 2017.

O *Android Studio* é baseado na ferramenta *IntelliJ*[2], que também é uma outra IDE para desenvolvimento *Java*, e incorpora diversos recursos específicos da plataforma *Android*, incluindo as mais recentes bibliotecas e SDKs disponibilizadas.

As bibliotecas são pequenos pacotes de *software* que fornecem acesso à funções específicas e podem ser incorporadas ao projeto para facilitar seu desenvolvimento. A SDK do *Android* é o conjunto de ferramentas e bibliotecas da plataforma que são a base sobre o qual a IDE *Android Studio* é construída, e fornece as funcionalidades que permitem o desenvolvimento das aplicações para *Android*.

Para possibilitar o teste das aplicações que estão em desenvolvimento, o *Android Studio* possui uma ferramenta chamada AVD. O AVD é uma máquina virtual que simula um dispositivo móvel com o sistema operacional *Android* e permite que a aplicação seja testada em tempo real sem a necessidade de instalação em um dispositivo físico, embora seja possível a conexão de um *smartphone* para o uso das funções de *debug* da aplicação.

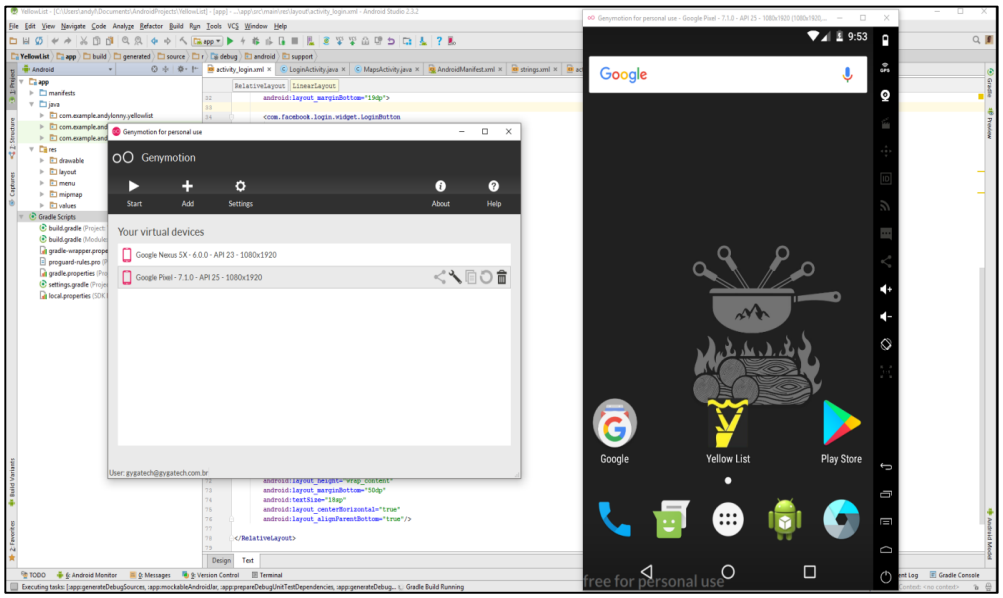
O uso da ferramenta AVD torna mais fácil e rápido o teste da aplicação, mas seu desempenho não é muito bom. Existem *softwares* de terceiros que fornecem essa

[2]https://www.jetbrains.com/idea/

funcionalidade e com melhor desempenho e são preferidos pelos desenvolvedores. Um dos mais conhecidos é o *Genymotion*[3].

O *Genymotion* utiliza virtualização para disponibilizar dispositivos virtuais com qualquer versão do sistema *Android* que o desenvolvedor queira testar sua aplicação, com a vantagem de ter um desempenho superior ao da ferramenta AVD do *Android Studio*. É disponível em uma versão gratuita para uso pessoal, sendo necessária a aquisição de licença para uso profissional.

Figura 2 - Tela Genymotion.



Fonte: Capturada em junho de 2017.

Web Service e Banco de Dados

A maioria das aplicações precisa de algum método para persistir informações de modo que possam ser acessadas posteriormente quando necessárias. Existem informações que estão relacionadas mais com a execução do aplicativo em si e geralmente são armazenadas localmente no dispositivo. Outras são relacionadas com as regras de negócio da aplicação, por exemplo os dados de usuários entre outros. Essas informações são disponíveis para todos os dispositivos nos quais a aplicação é executada,

[3]https://www.genymotion.com

sempre respeitando regras de acesso e para isso em geral são utilizados sistemas de bancos de dados complexos, os SGBDs.

Em aplicações *Java* escritas para ambiente *desktop*, é relativamente fácil utilizar conexões diretas ao SGBD e em geral estão disponíveis muitos recursos computacionais como memória e processamento para executar essas funções complexas.

Quando estamos utilizando o ambiente móvel como o *Android*, em geral temos limitações nos recursos computacionais disponíveis e o acesso precisa ser realizado de outra forma.

Para armazenar as informações locais, o *Android* possui um sistema de banco de dados de menor porte chamado *SQLite*. Ele é um sistema de banco de dados SQL, multiplataforma e que permite muito facilmente gerenciar as informações que o aplicativo precisa em tempo de execução, tudo que é armazenado localmente no dispositivo em que o aplicativo é executado, por exemplo configurações e informações de *log* (ANDRADE; AGRA; MALHEIROS, 2012).

> O SQLite é um banco de dados transacional leve e multiplataforma utilizado pelas principais plataformas móveis, como Android e iPhone [SQLite 2012]. Ele permite o armazenamento de textos, números e dados binários e pode ser consultado através da linguagem SQL diretamente da aplicação ou via linha de comando (ANDRADE; AGRA; MALHEIROS, 2012, p 783).

Para o Banco de Dados geral da aplicação, que irá gerenciar as informações comuns e disponíveis para todos os usuários do sistema, será necessário um sistema SGBD de grande porte, capaz de gerenciar um grande volume de transações e com excelente desempenho.

Como o aplicativo será utilizado em dispositivos móveis, o acesso ao banco de dados será através da *internet*. Isso significa que o banco de dados terá que ser hospedado em algum tipo de serviço remoto, e não em uma infraestrutura local. Hoje existe no mercado vários provedores de serviços de Nuvem (*Cloud Computing*) que oferecem opções de bancos de dados relacionais gerenciados, que facilitam o desenvolvimento das aplicações.

Os serviços de bancos de dados gerenciados oferecidos por essas empresas executam todo o trabalho de implantação e manutenção da infraestrutura do SGBD e garantem o seu funcionamento, abstraindo esse trabalho de rotina e entregando tudo pronto para o desenvolvedor que pode focar apenas em sua aplicação. Este precisa se preocupar apenas com suas tabelas e no modo como sua aplicação utilizará o serviço.

Acessar o banco de dados diretamente através da aplicação *Android* é possível utilizando as ferramentas existentes no *Java* como o JDBC. O problema dessa abordagem é o consumo de recursos computacionais e a dificuldade de gerenciar as conexões para evitar problemas de desempenho ou mesmo de segurança, visto que o acesso nos

aparelhos móveis nem sempre acontece em redes seguras e com alta velocidade garantida.

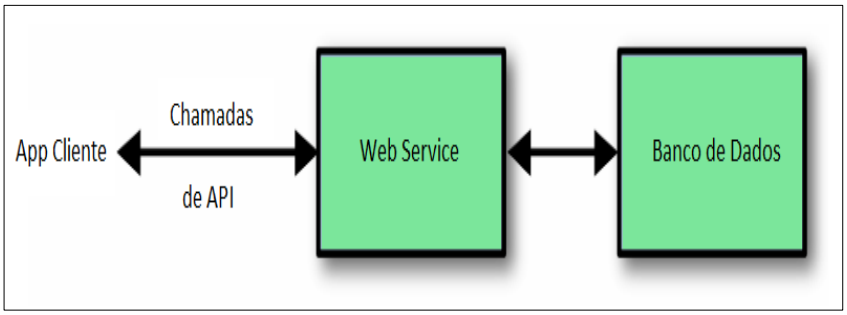
Outro problema é que caso no futuro a aplicação venha a ser portada para outras plataformas, como por exemplo o iOS, todo o código relativo ao acesso ao banco de dados não poderá ser aproveitado e terá que ser escrito do zero novamente. Uma forma de resolver estes problemas é com o uso de uma aplicação chamada *Web Service* (KALIN, 2013).

Web Service é uma aplicação *online* que funciona como um serviço geralmente hospedado em um servidor remoto. Para acessar um *Web Service* são utilizadas mensagens simples e com regras bem definidas que podem ser utilizadas por qualquer aplicação, independente da linguagem de programação em que foram escritas ou da plataforma em que são executadas.

Certamente, um grande apelo dos *Web Services* é a transparência da linguagem: o serviço e seus clientes não precisam ser escritos na mesma linguagem. A transparência da linguagem é um contribuinte fundamental para a interoperabilidade dos *Web Services* - isto é, a capacidade dos *Web Services* e seus consumidores de interagir sem problemas, apesar das diferenças nas linguagens de programação, bibliotecas de suporte, sistemas operacionais e plataformas de *hardware* (KALIN, 2013).

O serviço recebe as requisições da aplicação *Android* e então se encarrega do acesso ao Banco de Dados, que pode até estar hospedado na mesma estrutura de Nuvem que ele. Nesse caso o acesso entre o Banco de Dados e o *Web Service* será de alta velocidade e segurança e ficará escondido das aplicações móveis e restritos às regras definidas pelo serviço, conforme figura 3.

Figura 3 - Interação entre Aplicação, Web Service e Banco de Dados.



Fonte: Adaptado de Kalin (2013).

Existem dois tipos principais de *Web Services*, o tipo SOAP e o tipo REST como por exemplo o *RESTful*. A principal diferença entre eles é no tipo de mensagem suportada. Ambos trocam mensagens utilizando o protocolo HTTP padrão, mas no SOAP são utilizados documentos XML na troca de informações entre o serviço e o cliente (KALIN, 2013).

As aplicações Android podem utilizar ambos os tipos facilmente, pois existem bibliotecas que facilitam o desenvolvimento para qualquer tipo de *Web Service*, ficando a escolha mais relacionada com a familiaridade do desenvolvedor.

Dessa forma a aplicação fica independente da forma de acesso do banco de dados principal, facilitando a portabilidade para novas plataformas quando necessário sem que grandes alterações na estrutura de *back-end* e servidores seja realizada.

Atualmente muitos aplicativos móveis utilizam *Web Service* como forma padrão de trocar informações com um servidor. O servidor fica responsável por grande parte do processamento da lógica de negócio da aplicação e o aplicativo se encarrega da interface com o usuário, recebendo os comandos e entregando as respostas.

Spring Framework

O *Spring* é um *framework* de código aberto que simplifica o desenvolvimento de aplicativos *Java* no ambiente corporativo. Com ele é possível desenvolver utilizando simples *JavaBeans*[4] e pode ser utilizado em qualquer tipo de aplicação, seja ela do tipo *web*, *desktop* ou corporativa (BRAGA, 2011).

Outra vantagem do uso do *Spring* é que ele se integra e permite o uso de vários outros *frameworks*, automatizando tarefas complexas e liberando o programador para se preocupar mais com suas regras de negócio.

O *Spring* oferece pontos de integração com diversos outros *frameworks* e bibliotecas, de maneira que não tenha que desenvolvê-los por si próprio (WALLS; BREIDENBACH, 2008).

O *Spring* é totalmente modular, sendo assim o programador pode utilizar apenas os módulos responsáveis pelas funções desejadas. Neste projeto, o *Spring* é utilizado no desenvolvimento da aplicação *Web Service* com os módulos de DAO, ORM e Web.

DAO e ORM são responsáveis pelo controle do acesso ao Banco de Dados e a forma como a aplicação interage com o acesso aos dados. O Módulo *Web* será o responsável pelo funcionamento do *Web Service*, tendo o controle completo do ciclo de vida da aplicação e do modo como as informações são recepcionadas e processadas pelo *Web Service* (BRAGA, 2011).

O uso do *Spring* permite aos desenvolvedores utilizar classes *Java* simples para a manipulação dos dados e com pouco trabalho obter resultados muito complexos, tudo devido a automação proporcionada pela ferramenta.

Como ponto negativo, fica a necessidade de configuração do *framework* via alteração de arquivos XML específicos e não muito amigáveis. No entanto, quando a configuração é terminada ela pode ser aproveitada em outros projetos sem grandes dificuldades.

[4] Componentes reutilizáveis de software

PROSPECTUS
ISSN 2674-8576 Gestão e Tecnologia
Artigo Original

Integração com APIs do *Facebook* e *Google*

Para realizar o controle de acesso e *login* dos usuários na aplicação, serão disponibilizados dois métodos independentes e com pequenas diferenças na funcionalidade. Devido à popularidade das redes sociais e da familiaridade e confiança que os usuários possuem em empresas como o *Facebook* e *Google*, a integração do aplicativo com as tecnologias disponibilizadas por essas empresas se torna muito benéfica ao projeto.

Para essa integração são disponibilizadas pelo *Facebook* e *Google* ferramentas e SDKs com documentação detalhada que permite a utilização dos recursos de forma fácil pelos desenvolvedores interessados. Ambas as empresas disponibilizam um portal para que o desenvolvedor realize um cadastro e possa dar início a configuração a aplicação. O controle dos recursos utiliza a geração de chaves de autenticação que validam o acesso e permitem o acompanhamento com relatórios personalizados para o desenvolvedor verificar o desempenho de sua aplicação (FIGUEIREDO, 2015).

Para o *Facebook* é utilizado o acesso ao portal do desenvolvedor[5] que permite o cadastro da aplicação, a geração das chaves de segurança e o acesso a documentação auxiliar.

A utilização das APIs do *Google* acontece de maneira semelhante. É disponibilizado um portal[6] que permite o acesso a todas as APIs fornecidas pela empresa, como por exemplo o *Google Maps*[7] que será necessário ao projeto da aplicação.

Como uma opção para os usuários que não queiram optar pelo uso das redes sociais, será disponibilizado o acesso mediante um cadastro interno no banco de dados da aplicação. É previsto apenas um formulário básico de cadastro com informações de *login* e contato para garantir o mínimo de segurança na utilização do aplicativo e também o controle e registro das consultas para futura utilização.

Google Firebase

O *Firebase* é um conjunto de serviços e APIs dispostos em uma plataforma de nuvem, com a finalidade de auxiliar os desenvolvedores de aplicações móveis a construir rapidamente aplicações de qualidade. Atualmente a plataforma pertence ao *Google* e seu uso está ficando cada vez mais comum por parte da comunidade de desenvolvedores *Android* (GONÇALVES, 2016).

A documentação e as referências sobre os serviços podem ser encontradas no site[8] oficial da plataforma. Vários serviços disponíveis no *Firebase* podem auxiliar no

[5] https://developers.facebook.com
[6] https://developers.google.com
[7] https://developers.google.com/maps
[8] https://firebase.google.com

desenvolvimento do aplicativo, sendo o *Firebase Authentication* um dos mais importantes.

Com o *Firebase Authentication* fica mais fácil gerenciar as APIs necessárias para o controle de acesso e *login* na aplicação. A ferramenta do *Facebook* descrita no capitulo acima fica integrada a API do *Firebase* e o gerenciamento é feito automaticamente dependendo da escolha do usuário. No *Firebase* também é possível utilizar a autenticação simplificada com *e-mail* e senha descrita no capítulo anterior, servindo como mais uma opção aos usuários da aplicação.

Também serão utilizados os serviços de *Realtime Database* e *Notifications* para armazenar dados comuns a todos os usuários em tempo real e disparar notificações quando necessário.

Por fim, o *Firebase* possui uma ferramenta chamada *Analytics*, que pode ser utilizada para gerar gráficos e monitorar várias informações sobre o aplicativo e os recursos utilizados, bem como a quantidade de instalações efetuadas permitindo verificar a popularidade da aplicação na *Play Store*.

***Interface* de Usuário**

Um dos aspectos mais importantes no desenvolvimento de uma aplicação para a plataforma móvel é a experiência que essa aplicação passa ao usuário. O modo como o usuário visualiza as telas, como ele interage com os botões e componentes dispostos pela tela e o fluxo de trabalho que ele tem para a correta utilização dos recursos disponíveis são determinantes para o sucesso e a aceitação da aplicação (LIMA JUNIOR; SILVA, 2016).

Todos esses fatores devem ser considerados durante as fases de planejamento da *interface* de usuário.

Essa fase do desenvolvimento tem tanta importância que as empresas que controlam as plataformas móveis mais utilizadas, como a *Apple* e a *Google*, especificam regras e procedimentos que os desenvolvedores devem seguir para garantir uma padronização e a qualidade mínima necessária nas interfaces dos aplicativos lançados aos usuários.

No caso da plataforma *Android*, essas regras não são obrigatórias, mas é desejável que os desenvolvedores estejam cientes e as utilizem durante o planejamento de suas *interfaces* de usuário.

> Também neste sentido, a Google apresenta um guia chamado de "*Android Design*", que trata de princípios básicos de design, materiais para design, dispositivos, estilos, padrões, entre outros. Em contraponto do que acontece na Apple, os aplicativos desenvolvidos para Play Store não passam por nenhum processo de aprovação. Sendo assim, o guia da Google não é imposto como obrigação em relação às observações nela feita para o desenvolvimento de interface do usuário (LIMA JUNIOR; SILVA, 2016, p. 32).

O *design* e a aparência da aplicação devem ser levados em consideração na temática proposta, a escolha do logo, das cores e quantidade de componentes visuais e garantir uma utilização agradável que não sobrecarregue o usuário com informações confusas e desnecessárias.

É importante tomar cuidado com avisos e informações que possam aparecer repentinamente e em excesso, como por exemplo propagandas caso sejam utilizadas como forma de monetização da aplicação. Segundo (RECH, 2013, p 25) "...é preciso garantir que os anúncios colocados no aplicativo não afetem a experiência do usuário. As pessoas não gostam de ser "bombardeadas" com anúncios, e normalmente optam por ficarem longe deste tipo de app."

Também deve ser considerado o modo como o usuário utiliza a aplicação e como as informações são exibidas de volta. Deve-se pensar quais os componentes serão utilizados para entrada de dados, como a utilização da tela e dos atalhos gestuais permitidos pelo sistema Android serão aproveitados e como o fluxo de acesso e exibição das telas ocorrerá.

> São utilizados elementos da interface, como botões, tipografia, efeitos sonoros, ícones e cores para o usuário interagir com estes e manipular o conteúdo da aplicação. Uma aplicação móvel com uma boa interface deve ter uma boa interação do usuário com a tela, para que o mesmo possa fazer suas tarefas da melhor forma possível. (LIMA JUNIOR; SILVA, 2016, p 32).

Como todos esses fatores são de caráter pessoal e subjetivo, existe uma dificuldade natural em determinar exatamente como atender a todos esses requisitos. É por esse motivo que a utilização dos documentos e regras fornecidos pelo mantenedor da plataforma deve ser efetuada, de modo a garantir ao menos uma padronização com os demais aplicativos existentes no ecossistema da plataforma *Android*.

Como será abordado no capítulo abaixo, a realização de testes de usabilidade com um protótipo é um caminho muito válido e utilizado pelos desenvolvedores para tentar atingir esses objetivos e garantir a aceitação de seu aplicativo em um mercado tão cheio de opções.

PROSPECTUS
ISSN 2674-8576 Gestão e Tecnologia
Artigo Original

Validação da Prototipagem

Para realizar uma avaliação sobre o desenvolvimento da aplicação proposta neste artigo deve-se definir alguns parâmetros sobre o que realmente pretende-se obter e quais as funcionalidades poderão ser realizadas.

Será desenvolvido um protótipo da aplicação e também dos sistemas que funcionam no *back-end* e permitem a utilização das funcionalidades básicas esperadas (SOARES, 2007).

O protótipo terá uma *interface* de usuário com as opções de *Login* implementadas, incluindo a integração com as APIs do *Facebook* e *Google*. Essa *interface* possibilitará a realização das funcionalidades principais como a busca por prestadores de serviço, a seleção por localidade e visualização dos mapas e das recomendações. Todos os resultados serão exibidos adequadamente após a realização das consultas.

Esse protótipo será avaliado em testes de usabilidade, com usuários reais em um programa de voluntariado, para testar e validar a interface da aplicação e o modo como a experiência de uso está funcionando.

O Banco de Dados e o *Web Service* associado serão disponibilizados, visto que são essenciais para a operação do protótipo. Serão definidas as tabelas básicas e as regras de acesso via *Web Service*. Será preparado em nuvem um ambiente para operação dos sistemas *back-end* em modo de teste e permitir assim verificar o protótipo em situação real de uso, que permitirá verificar o desempenho e suas funcionalidades básicas.

DESENVOLVIMENTO TEÓRICO

Por que Dispositivos Móveis?

Atualmente a tecnologia está presente em praticamente todos os aspectos da vida cotidiana das pessoas, sem distinções de idade, condição financeira ou qualquer outro aspecto sociológico. Não conseguimos mais vislumbrar uma situação em que a tecnologia não tenha alguma influência sobre as atividades do dia-a-dia (LEMOS; JOSGRILBERG, 2009).

A evolução da *internet* na última década propiciou uma forma muito mais direta de acesso a informação, permitindo que qualquer pessoa tenha a sua disposição informações antes restritas a locais como universidades ou bibliotecas, praticamente em qualquer lugar em que seja possível o uso de um computador (LIMA JUNIOR; SILVA, 2016).

Nos últimos anos temos presenciado uma mudança na forma de acesso a esse conteúdo. Antes restrito ao computador pessoal, agora está disponível nos mais variados tipos de dispositivos. O mais representativo na vida das pessoas é o *Smartphone*. Inicialmente restrito ao serviço de comunicação por voz, hoje passa a ser protagonista na

forma de acesso às informações utilizando as conexões de dados fornecidas pelas diversas operadoras de telefonia disponíveis no mercado (PELLANDA, 2009).

> O acesso à *internet* começa a ser o próximo canal de expansão da comunicação móvel no país à medida que as redes de telefonia vão se expandindo e os custos começam a baixar com a escala do aumento de usuários (PELLANDA, 2009, p 12).

A Comunicação Móvel vem se tornando uma das principais formas de acesso à *internet* por parte dos brasileiros. De acordo com (LEMOS; JOSGRILBERG, 2009), números oficiais apontavam que em junho de 2009 existiam quase 160 milhões de acesso ao serviço móvel pessoal, sendo que 81,82% estão presentes na modalidade pré-pago. De forma significativa, observa-se, atualmente, um aumento no número de usuários segundo dados publicados pela ANATEL, uma vez que em fevereiro de 2015 existiam 282,56 milhões de linhas ativas na telefonia móvel (ANATEL, 2015).

Todo esse cenário favorece uma mudança nos hábitos de consumo da informação por parte das pessoas, visto que esses acessos móveis podem ser realizados em qualquer lugar e a qualquer momento. O *smartphone* tornou-se um dispositivo quase essencial nas vidas das pessoas.

Como consequência tem-se também uma mudança perceptível nas atividades comerciais e sociais induzida pela comunicação móvel. O acesso à *internet*, às mídias sociais e todas as formas de relacionamento interpessoal proporcionado pelos dispositivos móveis, permite também que essa tecnologia seja aproveitada para fins comerciais e mercadológicos.

> Nesse contexto, a comunicação móvel está transformando atividades econômicas e sociais de maneira profunda. Desde um vendedor de cachorro quente ambulante que pode oferecer serviços de tele-entrega até profissionais *freelancers* que podem ter escritórios móveis. Com isso, várias funções da economia informal nasceram dessa possibilidade. Tais atividades representam uma importante parcela da economia brasileira (PELLANDA, 2009, p 16).

Toda essa base de usuários conectados às redes de comunicação móvel representa uma fatia significativa de mercado consumidor em potencial para o desenvolvimento de várias atividades comerciais, que podem ser aproveitadas não apenas por empresas, mas também por profissionais liberais prestadores de serviço que são o motivo para o desenvolvimento desse trabalho acadêmico.

Plataformas Móveis Mais Utilizadas

Nos primeiros anos da telefonia móvel, os *smartphones* ainda eram apenas aparelhos celulares com *hardware* bem simples e foco apenas nas comunicações por voz.

O *Software* necessário para o funcionamento desses aparelhos não era muito complicado e desenvolvido geralmente pelo próprio fabricante do dispositivo (ANDRADE; AGRA; MALHEIROS, 2012).

Com o tempo a plataforma foi se popularizando e os aparelhos foram evoluindo e adquirindo mais poder de processamento e memória, possibilitando novos usos que antes não seriam possíveis.

Motivada pela evolução do *hardware* desses aparelhos, o *software* acabou se tornando um verdadeiro Sistema Operacional, permitindo o gerenciamento de todos os recursos e integrando possibilidades de rede e conexões de dados chegando ao ambiente que possuímos hoje.

Esses Sistemas Operacionais deram origem a ambientes de desenvolvimento próprios, com suas linguagens de programação, ferramentas e regras que permitem que aplicativos sejam desenvolvidos e estendam as funcionalidades existentes nos aparelhos. Esses ambientes de desenvolvimento são chamados de IDE e permitem o desenvolvimento das chamadas aplicações nativas, que possibilitam o uso dos recursos disponíveis no sistema operacional e no aparelho em que serão instalados (SILVA; SANTOS, 2014).

Uma aplicação nativa exibe a característica de ser desenvolvida apenas para uma plataforma especifica, não sendo possível utiliza-la em um aparelho que utilize um Sistema Operacional diferente do para qual ele foi projetado. Isso tem como desvantagem o fato de que esse aplicativo tem que ser desenvolvido para todas as plataformas em que se deseje utilizá-lo, aumentando as dificuldades e os custos de desenvolvimento da aplicação. De acordo com (SILVA; SANTOS, 2014, p 02), “O desenvolvimento de um aplicativo nativo para aparelhos celulares exige conhecimentos específicos a respeito das tecnologias utilizadas pela plataforma na qual se deseja executá-lo”.

Como vantagem, os aplicativos nativos possuem a velocidade de execução, que em geral é muito boa e também à similaridade com o próprio sistema operacional para o qual foram desenvolvidos, pois em geral utilizam os mesmos padrões e componentes visuais tornando a experiência de uso agradável para os usuários da aplicação.

Existem também ferramentas de desenvolvimento multiplataforma que permitem o desenvolvimento de aplicativos compatíveis com vários sistemas operacionais diferentes. A grande vantagem dessas ferramentas é agilidade no desenvolvimento de aplicações compatíveis com várias plataformas, mas devido a velocidade em que mudanças ocorrem nos sistemas operacionais, muitas vezes não será possível acessar todos os recursos disponíveis na plataforma utilizando essas ferramentas (SILVA; SANTOS, 2014).

Atualmente entre as diversas plataformas de Sistemas Operacionais para dispositivos móveis existentes, duas se destacam pela quantidade de aparelhos em que são utilizadas. Temos o *Android* desenvolvido pela *Google* e o *iOS* desenvolvido pela *Apple*.

A plataforma *Android* tem como vantagem ser de código aberto, isso possibilita que muitos fabricantes de aparelhos celulares possam utilizá-lo em seus produtos. Isso reflete em uma maior oferta de produtos para os consumidores e com isso uma redução do custo desses aparelhos (DEITEL *et al*., 2015).

A escolha do uso da plataforma *Android* para o desenvolvimento desse trabalho é baseada na existência de uma grande base de usuários de dispositivos compatíveis, linguagem *Java*, aliada ao baixo custo da plataforma e a grande variedade de ferramentas e informações que auxiliam o desenvolvimento de aplicativos.

Características da Plataforma *Android*

A plataforma *Android* é composta por um conjunto de *softwares* desenvolvidos para dispositivos móveis, que incluem o sistema operacional, bibliotecas ferramentas de desenvolvimento e aplicativos chave que atendem as necessidades básicas de uso desses dispositivos.

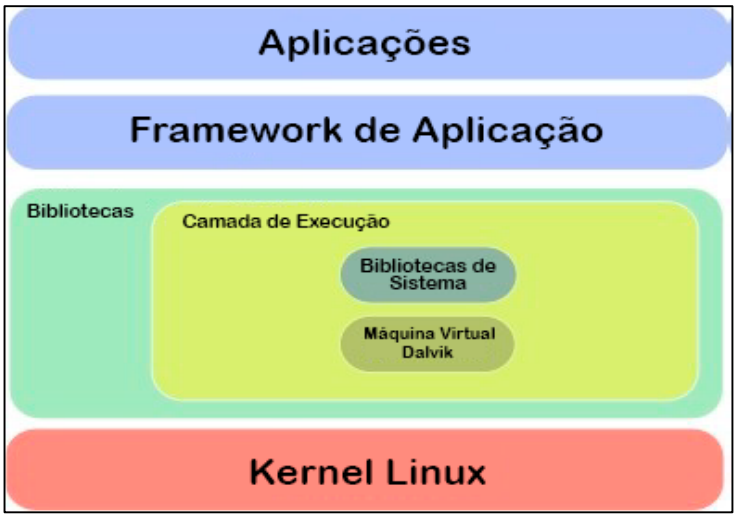
O sucesso da plataforma perante a indústria deve-se principalmente à sua arquitetura interna baseada em *Linux* e ao fato de ser *open-source*. A arquitetura interna do *Android* garante acesso a várias características existentes no sistema *Linux*, principalmente quanto a otimização de memória e uso do *hardware*, que são fundamentais em dispositivos móveis (GANDHEWAR; SHEIKH, 2010).

Ser *open-source*, ou de código livre, garante acesso ao código fonte do sistema permitindo que os fabricantes e desenvolvedores possam entender melhor como o sistema funciona e contribuir ativamente com melhorias, tanto de desempenho e principalmente segurança. Isso também facilita a integração do sistema ao *hardware* dos variados dispositivos existentes no mercado, melhorando a compatibilidade e facilitando futuros lançamentos dos diversos fabricantes.

A plataforma *Android* é baseada em *Java* e por isso se utiliza do conceito de Máquina Virtual ou *Java* VM. No caso do *Android* é utilizada uma VM própria chamada de *Dalvik*, que interpreta o código baseado em *Java* e o executa com melhorias e otimizações especificas para a plataforma móvel. Segundo (GANDHEWAR; SHEIKH, 2010) *Dalvik* executa aplicações *Java* que foram transformadas no executável *Dalvik* (.dex) que é otimizado para alocar o mínimo da memória.

Um exemplo da arquitetura interna da plataforma *Android* é mostrado na figura abaixo:

Figura 4 - Arquitetura interna do Sistema Operacional.



Fonte: Adaptado de Gandhewar; Sheikh (2010).

Na camada de Aplicações estão disponíveis para a utilização do usuário desde os aplicativos básicos que são para realizar chamadas de voz, envio e recebimento de SMS, navegador *web*, que já vem por padrão no sistema, tanto suportada para receber as diversas opções de aplicativos disponíveis no mercado. A camada de *Framework* garante a estrutura, serviços e funções destinadas as aplicações desenvolvidas para a plataforma. No *Android* as aplicações podem ser executadas simultaneamente sem interferência entre elas. Na próxima camada ficam as bibliotecas e a Máquina Virtual *Dalvik*. As bibliotecas são acessadas pelo código *Java* e disponibilizam suporte a *Codecs* de áudio e vídeo como MP3 e MP4, bancos de dados SQL com o *SQLite*, funções visuais 2D e 3D entre outras. No suporte a todas as camadas anteriores está o núcleo *Linux* que garante a separação entre o *software* e o *hardware*, bem como os serviços básicos como gerenciamento de memória, processamento, rede, processos e demais serviços (GANDHEWAR; SHEIKH, 2010).

Localização de Prestadores de Serviço

O intuito do aplicativo é facilitar a busca por prestadores de serviços mais próximos de sua localidade, pois isso facilita o usuário que precisa de um serviço com caráter emergencial, já que a sua busca será específica não tendo que perder tempo em pesquisar em sites que indicam prestadores de serviços de outras localidades e na maioria das vezes empresas que só atendem em horário comercial. Após o serviço ser realizado,

o usuário terá a opção de avaliar o profissional, gerando confiança para os outros usuários e também visando manter a excelência do serviço prestado.

Plataforma de Divulgação para Prestadores de Serviço

Geralmente nas cidades situadas no interior do estado, os prestadores de serviços costumam ser contratados da maneira tradicional, utiliza-se a publicidade boca-a-boca, pessoas quando precisam encontrar algum profissional, entram em contato com seus conhecidos para pedir indicações, é um método válido, mas o prestador de serviço fica limitado aquele público que está acostumado a chamá-lo. A vantagem de o prestador de serviço estar relacionado no aplicativo é poder atender os mais variados tipos de clientes, aumentando assim sua visibilidade na região atendida. Outra vantagem também é que com isso os profissionais liberais consigam atender as necessidades dos usuários de acordo com as suas competências em horários alternativos (AMORIM; SANTOS, 2015).

E-Commerce* Tipo *Online-to-Offline (O2O)

O aumento de dispositivos móveis e serviços baseados em localização impulsionaram a ascensão do mercado O2O, caracterizado por transações de serviços e produtos que têm início no meio virtual e terminam no meio físico, atendendo a necessidades específicas da população de determinada região. E esse hábito está transformando a maneira como as pessoas fazem compras.

A essência do O2O é utilizar ferramentas online, para impulsionar as empresas cuja natureza vem do *offline*, ampliando a oferta de serviços que, anteriormente, não eram acessíveis à distância, para o meio *online*. Por exemplo, o Uber une passageiros e motoristas que antes não conseguiam se conectar. Sites como *iFood, JustEat* e *Postmates* permitem que restaurantes locais ofereçam seus cardápios *online* e realizem entregas. O *Yellow List* possibilitará que as pessoas encontrem prestadores de serviços que possam solucionar seus problemas (CHENTAO; YONGLE, 2014).

O O2O reduz os obstáculos entre clientes e empresas, além de mobilizar uma potência de trabalho inexplorada, como motoristas, entregadores, domésticas, fazendo com que as pessoas complementem a renda, empresários ampliem seus negócios e certas operações eliminem suas lojas físicas.

Apesar de muitos desses produtos e serviços estarem presentes na web, a atividade está focada nos dispositivos móveis, o que amplia, ainda mais, a conveniência. Normalmente, os clientes utilizam as plataformas de O2O porque precisam de algo com urgência e, muitas vezes, estão na rua, no trabalho ou em festas com os amigos, e não à frente de um computador. Além disso, a tela pequena dos dispositivos móveis estimula os desenvolvedores a otimizarem a experiência dos usuários, minimizando os cliques até o fim do processo (CHENTAO; YONGLE, 2014).

Outro ponto relevante é o fato de aparelhos celulares e *tablets* oferecerem múltiplos graus de personalização, reunindo dados baseados em localização, bem como preferências. Isso gera maiores taxas de conversão, uma vez que as soluções para a busca de cada usuário passam a ser mais padronizadas, tornando a experiência única e movimentando o mercado de O2O.

Desenvolvimento do Aplicativo e *Web Service*

Desenvolvimento do Aplicativo

Os aplicativos da plataforma *Android* são desenvolvidos utilizando-se *JAVA*, que é a linguagem de programação nativa escolhida pelo *Google*, mas existem algumas peculiaridades que diferenciam o modo como a linguagem é utilizada durante o desenvolvimento para *Android* de como o *JAVA* é utilizado nas outras plataformas (DEITEL *et al*., 2015).

Aplicativos *Android* possuem uma separação muito bem definida entre a programação dos *Layouts* visuais, ou seja, das telas do aplicativo e a programação da lógica que implementa as funcionalidades esperadas.

Existem quatro tipos básicos de componentes que podem ser utilizados para construir uma aplicação *Android*:

- ***Activity*:** representa uma única tela que serve como ponto de entrada para o usuário. São basicamente as telas que formam a aplicação.
- ***Services*:** são componentes que rodam em segundo plano, executando tarefas de longa duração e não possuem *interface* visual.
- ***Content Providers*:** são utilizados para gerenciar dados da aplicação, como por exemplo acessar arquivos, bases de dados ou fazer requisições *web.*
- ***Broadcast Receivers*:** são componentes que servem para interceptar e exibir mensagens do sistema ou de outras aplicações, como avisos de bateria fraca por exemplo.

Cada *activity* é independente das outras e possuem seu próprio ciclo de vida no sistema, mas trabalham juntas para formar uma aplicação completa. É possível até utilizar *activities* de outas aplicações instaladas no dispositivo, como por exemplo para enviar um e-mail ou acessar a câmera fotográfica. As *activities* são a parte central de uma aplicação, com os outros componentes sendo utilizados como auxiliares para as funções mais especificas (RAYNALDO, 2012).

Antes da aplicação ser iniciada pelo sistema, o *Android* precisa saber quais os componentes fazem parte da aplicação, quantas activities são utilizadas e qual é a *activity* inicial a ser carregada. Para isso existe um arquivo chamado *AndroidManifest.xml*.

PROSPECTUS
ISSN 2674-8576 Gestão e Tecnologia
Artigo Original

O arquivo *Manifest* é do tipo XML e declara todos os componentes que fazem parte da aplicação, qual é o componente inicial a ser carregado e também todas as permissões necessárias para a sua execução. Nele também é informada a versão mínima da API do *Android* que o aplicativo exige para funcionar, isso é, qual a versão mínima do sistema *Android* é necessária. O *Manifest* também é o local em que se vincula as chaves de acesso necessárias para todas as API de terceiros que serão utilizadas no aplicativo, por exemplo as chaves da API do *Google Maps* ou do *Facebook* (RAYNALDO, 2012).

Uma *activity* é dividia em duas partes distintas. Um arquivo de *Layout* do tipo XML que é o responsável pela definição da interface visual da tela e um arquivo *Java* que é responsável pela funcionalidade associada a essa *activity*. Um arquivo de *Layout* tem a estrutura básica abaixo:

Figura 5 - Arquivo layout_activity.xml.

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:id="@+id/activity_main"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:paddingBottom="16dp"
    android:paddingLeft="16dp"
    android:paddingRight="16dp"
    android:paddingTop="16dp"
    android:background="@color/colorPrimary"
```

Fonte: Criado pelos autores em 2017.

Nele são definidas as características visuais, como as cores, os componentes visuais que fazem parte da tela, o posicionamento dos componentes como botões, caixas de texto e imagens, e tudo que for relativo a características da *interface* com o usuário.

O arquivo que define as funcionalidades da *activity* é um arquivo tipo Classe *Java* padrão. Um exemplo é mostrado abaixo:

Figura 6 - Arquivo Activity.java.

```
public class MainActivity extends AppCompatActivity {

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

        //inicialização dos componentes e listeners dos Botões

    }

}
```

Fonte: Criado pelos autores em 2017.

O método *onCreate* mostrado acima é o ponto de entrada da *activity*. Ele é executado na criação da *activity* e contém o código necessário para o carregamento dos componentes utilizados. Neste método o *layout* é carregado no sistema à partir no arquivo XML anterior, e os componentes como botões e caixas de texto são efetivamente implementados. Toda a configuração das APIs e componentes necessários será executada neste método.

Como em toda Classe *Java*, outros métodos podem ser declarados neste arquivo e todas as funcionalidades da *activity* são declaradas aqui.

Para o funcionamento do aplicativo *Yellow List*, será necessário a troca de informações com uma API *web*, que é o *web service* definido anteriormente neste tabalho. Para isso são utilizados recursos das bibliotecas HTTP disponíveis no *Android*.

A biblioteca *httpURLconnection* possui todas as funcionalidades necessárias para implementar as chamadas HTTP que trocarão informações com o *web service* da aplicação. Através dessa bliblioteca serão contruidas as requisições tipo POST que enviarão para o *web service* os dados para consulta dos prestadores de serviços, as informações do usuário, formas de pagamento escolhidas e todas as informações necessárias. Ela também recebe as respostas e permite que os devidos tratamentos sejam realizados nas informações retornadas pelo *web service*. Um exemplo de método utilizando essa biblioteca é mostrado abaixo:

PROSPECTUS
ISSN 2674-8576 Gestão e Tecnologia
Artigo Original

Figura 7. Método "ConsultarPrestador" que utiliza a biblioteca *httpURLconnection*.

```
public static String consultarPrestador(ParametrosConsulta param){
    try{
        URL apiEnd = new URL("http://192.168.10.14:8080/YellowListAPI/localizarPrestador");
        conexao = (HttpURLConnection) apiEnd.openConnection();
        conexao.setDoOutput(true);
        conexao.setRequestMethod("POST");
        conexao.setReadTimeout(15000);
        conexao.setConnectTimeout(15000);
        conexao.setRequestProperty("Content-Type", "application/json;charset=utf-8");

        outputStream = new BufferedOutputStream(conexao.getOutputStream());
        JSONObject json = new JSONObject();
        try {
            json.put("lat", param.getLat());
            json.put("lng", param.getLng());
            json.put("categoriaServico", param.getCategoriaServico());
            json.put("raioDaConsulta", param.getRange());
        } catch (JSONException e) {
            e.printStackTrace();
        }
        BufferedWriter writer = new BufferedWriter(
                new OutputStreamWriter(outputStream));
        writer.write(json.toString());
        writer.flush();
        writer.close();
        outputStream.close();

        conexao.connect();
```

Fonte: Criado pelos autores em 2017.

Uma caracteristica da plataforma *Android*, é a necessidade de que os métodos que fazem uso de I/O como acesso a arquivos ou bancos de dados ou chamadas HTTP que podem demorar a serem executadas devido a influências externas, não possam ser executados na *thread* principal. Esses métodos precisam ser executados em segundo plano de modo a não bloquear a *activity* que está sendo exibida. Para isso existe uma classe chamada *AsyncTask* que recebe os dados, executa as chamadas HTTP fora da *thread* principal e retorna os resultados para a *activity* de origem. Isso evita avisos de ANR, ou seja, avisos de que a aplicação parou de responder que podem incomodar e afetar a experiência de uso da aplicação (DEITEL *et al*., 2015).

Um exemplo de utilização de *AsyncTask* é mostrado a seguir:

Figura 8 - Classe AsyncTask que consulta prestadores de serviços.

```
public class ConsultarPrestador extends  AsyncTask<ParametrosConsulta, Object, String> {

    @Override
    protected String doInBackground(ParametrosConsulta... params) {
        String resul = HttpUtils.consultarPrestador(params[0]);
        return resul;
    }
    @Override
    protected void onPostExecute(String resul) { lblResultadoConsulta.append(resul); }
}
```

Fonte: Criado pelos autores em 2017.

Essas caracteristicas tornam o desenvolvimento de aplicaçõs para *Android* um pouco diferentes das aplicações *JAVA* tradicionais. A linguagem básica é a mesma, mas existem diferenças que precisam ser entendidas e seguidas pelos programadores. No final, o uso da IDE *Android Studio* facilita um pouco o trabalho, pois muitos dos processos são automatizados e a própria ferramenta executa alguns desses passos.

Desenvolvimento do *Web Service*

Um componente de extrema importância no funcionamento da aplicação *mobile* é o *Web Service*. Ele é o responsável por receber as requisições do aplicativo, executar as principais operações lógicas associadas ao negócio, realizar as operações com o banco de dados, construir uma resposta segundo uma regra previamente definida e então devolver essa resposta para a aplicação que roda no dispositivo do usuário.

Esse *Web Service* é uma aplicação do tipo *web* desenvolvida utilizando a linguagem Java. Para facilitar e aumentar a produtividade do desenvolvimento foi utilizado o *Spring Framework* com seus módulos de Web, DAO e ORM. O ambiente de desenvolvimento utilizado foi o *Eclipse*[9].

O *Web Service* é do tipo REST, e trabalha recebendo mensagens via protocolo HTTP do tipo POST. No corpo das mensagens devem estar os parâmetros específicos da consulta obedecendo o formato JSON. As respostas que o *Web Service* retorna seguem o mesmo padrão HTTP POST com conteúdo em formato JSON.

O *Spring Framework* já possui bibliotecas especificas que facilitam a construção dessa aplicação e permitem tratar as requisições, realizar a quebra do JSON em objetos Java facilmente utilizáveis e então construir o JSON de resposta e envia-lo ao aplicativo cliente.

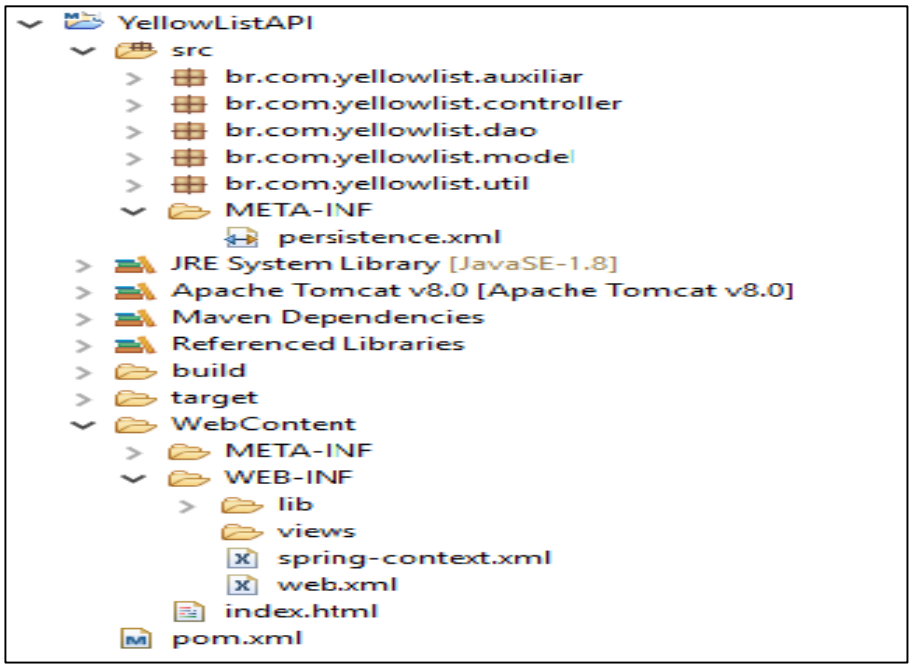
O primeiro passo é a criação do projeto dentro do *Eclipse* e a importação das bibliotecas necessárias, no caso o *Spring* e seus módulos auxiliares bem como as bibliotecas especificas para comunicação com banco de dados como JPA, *Hibernate*[10] e *MySQL connector*[11]. Como em qualquer projeto Java o uso da ferramenta *Maven* [12]auxilia no carregamento dessas bibliotecas.

A configuração do *Spring* e JPA são realizadas em arquivos tipo XML específicos. Basicamente esses arquivos preparam as bibliotecas para como trabalhar no contexto da aplicação, quais as classes de objetos Java representam entidades no Banco de Dados, as credenciais de acesso ao Banco e a URL de conexão, etc. (BRAGA, 2011).

[9] http://www.eclipse.org

[10] http://hibernate.org
[11] https://www.mysql.com/products/connector/
[12] https://maven.apache.org

Figura 9 - Estrutura do projeto Web Service no Eclipse.



Fonte: Criado pelos autores em 2017.

O arquivo *spring-context.xml* reside no diretório *WebContent/WEB-INF* do projeto e contém todas as configurações relativas ao *framework*. O arquivo *persistence.xml* no diretório *src/META-INF* possui as configurações relativas ao JPA/*Hibernate* responsável pelo mapeamento entro os objetos *Java* e as entidades no Banco de Dados Relacional.

A utilização do *Spring* na aplicação utiliza o padrão de anotações do *Java*. Cada anotação possui uma função especifica e auxilia o programador na execução das tarefas.

Na classe *ControllerAP*I, a anotação *@RestController* configura a função de controladora REST, responsável por receber as requisições, realizar os devidos processamentos e devolver a resposta. Dentro dessa classe, os métodos que recebem as requisições utilizam a anotação *@RequestMapping* que indica como o método será chamado e o tipo de requisição que ele atende, no nosso caso o POST.

Figura 10 - Exemplo da classe ControllerAPI.

```
38 @Transactional
39 @RestController
40 public class ControllerAPI {
41
42     @Autowired
43     PrestadorDao prestadorDao;
44
45     @RequestMapping(value = "/localizarPrestador", method = RequestMethod.POST)
46     public ResponseEntity<List<RespostaPrestador>> localizarPrestador
47     (@RequestBody() Coordenadas coordenadas){
48
49         List<RespostaPrestador> lista = new ArrayList<>();
50
51         /* bloco de codigo */
52
53         return new ResponseEntity<List<RespostaPrestador>>(lista, HttpStatus.OK);
54     }
55
56 }
```

Fonte: Criado pelos autores em 2017.

Com a anotação *@RequestBody* seguida de um objeto *Java*, o *Spring* é capaz de quebrar o JSON recebido como parâmetro na requisição e preencher esse objeto com as informações contidas no JSON. De posse desse objeto *Java* podemos executar nossa lógica de negócio e a resposta será devolvida pelo *ResponseEntity* no *return* do método.

A resposta contém um código HTTP 200 de confirmação e o objeto Java retornado pelo método, transformado em JSON automaticamente pelo *Spring*. Todos os métodos necessários na aplicação seguem o mesmo padrão, alterando apenas os parâmetros de entrada e os resultados devolvidos.

Para representar as entidades que são armazenadas no banco de dados relacional, são utilizadas classes *Java* que representam objetos utilizados em nossa aplicação. Esses objetos recebem anotações especificas que indicam ao S*pring* que a camada de persistência precisa monitorar seus estados e realizar as transações com o banco quando necessário.

Figura 11 - Exemplo da classe Contato.

```
 8 @Entity
 9 public class Contato {
10
11     @Id @GeneratedValue(strategy=GenerationType.IDENTITY)
12     private long id;
13     private String telefone;
14     private String celular;
15     private String email;
```

Fonte: Criado pelos autores em 2017.

A anotação *@Entity* indica que a classe representa uma entidade para persistencia no banco de dados. As anotações *@Id* e *@GeneratedValue* indicam que o atributo Id representa a chave primária que é controlada pelo SGBD. Existem outras anotações especificas para indicar os relacionamentos entre as tabelas e podem ser utilizadas quando necessário.

Para padronizar a comunicação entre a classe controladora e o banco de dados é utilizado o padrão DAO, com classes DAO especificas para cada classe de objetos modelados.

Figura 12 - Exemplo da classe ContatoDao.

```
12 @Repository
13 public class ContatoDao {
14
15     @PersistenceContext
16     private EntityManager manager;
17
```

Fonte: Criado pelos autores em 2017.

Neste caso a anotação *@PersistenceContext* indica ao *Spring* que essa classe realiza operações com o banco de dados e necessita de uma instância do *EntityManager*. O *EntityManager* faz parte da especificação JPA e controla o estado das entidades, garantindo a consistencia entre os objetos *Java* e as respectivas entidades no banco. A

anatoção *@Repository* liga a classe DAO a classe que a utiliza, neste caso a classe controladora.

Através desse padrão de anotações, o *Spring* mantém o controle da aplicação, garantindo que a classe controladora receba uma instância DAO quando for necessário. Por sua vez, a instancia DAO recebe o *EntityManager* e esse prove a conexão ao banco de dados. O desenvolvedor não precisa mais manter o controle sobre essas dependencias entre as classes e pode focar no código referente a lógica de negócio da aplicação.

Isso prove maior produtividade e maior reaproveitamente de código, visto que após configurado, a maior parte desse projeto pode ser aproveitado para outros desenvolvimentos.

Definição e Implantação da Infraestrutura

Para que os usuários possam utilizar o aplicativo, o *Web Service* tem que estar disponível sempre, sem interrupções ou o acesso ficará comprometido, assim como o seu uso. Para garantir essa disponibilidade, é preciso um planejamento cuidadoso sobre a infraestrutura que dará suporte a essa aplicação.

Esse planejamento envolve várias questões, por exemplo, escolher qual a capacidade de processamento será necessária, como a redundância será aplicada no quesito energia, conectividade, servidores, *backup* entre outras.

Atualmente, a computação em nuvem permite que essas questões sejam respondidas com muita facilidade. O modo como os serviços de nuvem trabalham, permite que a infraestrutura seja alocada sob demanda, assim os requisitos de computação podem acompanhar a necessidade do aplicativo. Outro fator importante é o custo da infraestrutura. Os provedores de nuvem possuem formatos de faturamento por consumo, em que os serviços são cobrados pelo que de fato consumiram dos recursos alocados, possuindo também alguns que são gratuitos e podem ser utilizados para avaliação e testes de viabilidade (POSSOBOM, 2010).

Devido a característica da aplicação ter que atender usuários espalhados por várias localidades e utilizando diversos provedores de acesso, seria muito difícil e de alto custo manter a infraestrutura localmente, com a aquisição de servidores e a preparação de um local adequado com energia e conectividade apropriados.

Neste caso, optou-se por utilizar um provedor de Computação em Nuvem. A escolha foi pela *Amazon Web Services* [13], visto que é uma das maiores empresas do ramo e conta com uma gama enorme de serviços disponíveis em seu catálogo (POSSOBOM, 2010).

Foi criada uma estrutura baseada nos serviços EC2, RDS e VPC disponíveis no catálogo da AWS.

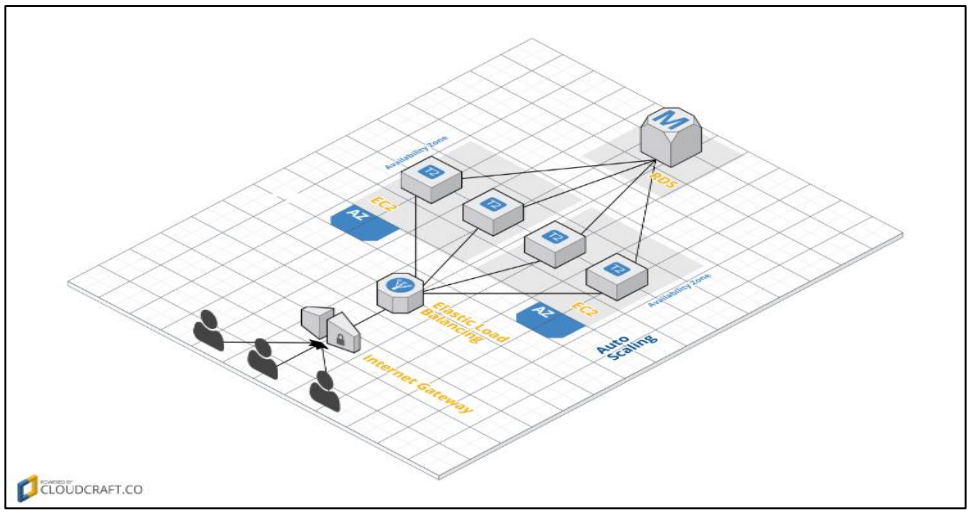
[13] https://aws.amazon.com/pt/

- **EC2:** São maquinas virtuais que funcionam como servidores e podem desempenhar diversas funções. Possuem diversos tamanhos em relação a CPU e memória e são utilizados como servidores web para executar nosso *Web Service*.
- **RDS:** É o serviço de banco de dados relacional gerenciado da AWS. Você escolhe o SGBD desejado e recebe uma instancia funcional do mesmo. Também possui diversos tamanhos de CPU e memória à disposição.
- **VPC:** É a rede virtual em que seus serviços serão executados. É possível configurar endereçamento e faixas de rede e subrede. Possui *firewall* e balanceadores de carga permitindo controle total sobre o que está disponível para a internet e o que deve ficar restrito internamente.

Utilizando esses três tipos de recursos, foi criada uma infraestrutura com servidores web para o Web Service e uma instancia do MySQL para receber a base de dados relacional utilizada. Para garantir alta disponibilidade aos usuários, foram criadas instancias EC2 em múltiplas zonas de disponibilidade, o que a AWS chama de *Availability Zone*. Isso significa que estão alocadas em *Datacenters* diferentes para garantir a disponibilidade em caso de problemas em um determinado ponto.

Com a VPC foi criado um balanceador de tráfego que distribui as chamadas para as instâncias EC2 de modo a garantir um desempenho equilibrado e com o *Firewall* criamos regras que isolam o banco de dados da rede externa. Assim apenas o *Web Service* tem acesso a instância RDS.

Figura 13 - Diagrama de Infraestrutura na AWS.



Fonte: Criado pelos autores no site https://cloudcraft.co/

Devido a flexibilidade da Computação em Nuvem, todos esses recursos podem ser escalados vertical ou horizontalmente, seja aumentando o tamanho das instâncias ou aumentando suas quantidades em paralelo. A infraestrutura não é rígida e os custos são controlados de acordo com a real necessidade da aplicação (POSSOBOM, 2010).

CONCLUSÕES

Durante a elaboração deste trabalho foi possível identificar que a forma de acesso a informação tem mudado atualmente. Os *smartphones* e as novas formas de conexão à *internet* como o 3G, 4G e *WiFi*, estão possibilitando que as pessoas tenham acesso a um vasto conteúdo de informações em praticamente qualquer lugar e a qualquer momento.

Hoje é possível consultar informações sobre qualquer pessoa ou empresa diretamente na tela do *smartphone*. Fazer compras *online* já é uma atividade corriqueira e o uso dos dispositivos móveis apenas vem acompanhando essa tendência.

O desenvolvimento do aplicativo *Yellow List* vem aproveitar uma oportunidade na área do *e-commerce* tipo O2O, possibilitando que as pessoas possam realizar a localização de prestadores de serviços, aproveitando das características que o acesso móvel e as novas tecnologias permitem.

Ao mesmo tempo, o trabalho apresenta um passo a passo demonstrando as etapas envolvidas no planejamento e desenvolvimento de uma aplicação para a plataforma *Android*. Foram apresentados os motivos para a utilização da plataforma móvel e o porquê da escolha do *Android* como sistema operacional suportado para a aplicação.

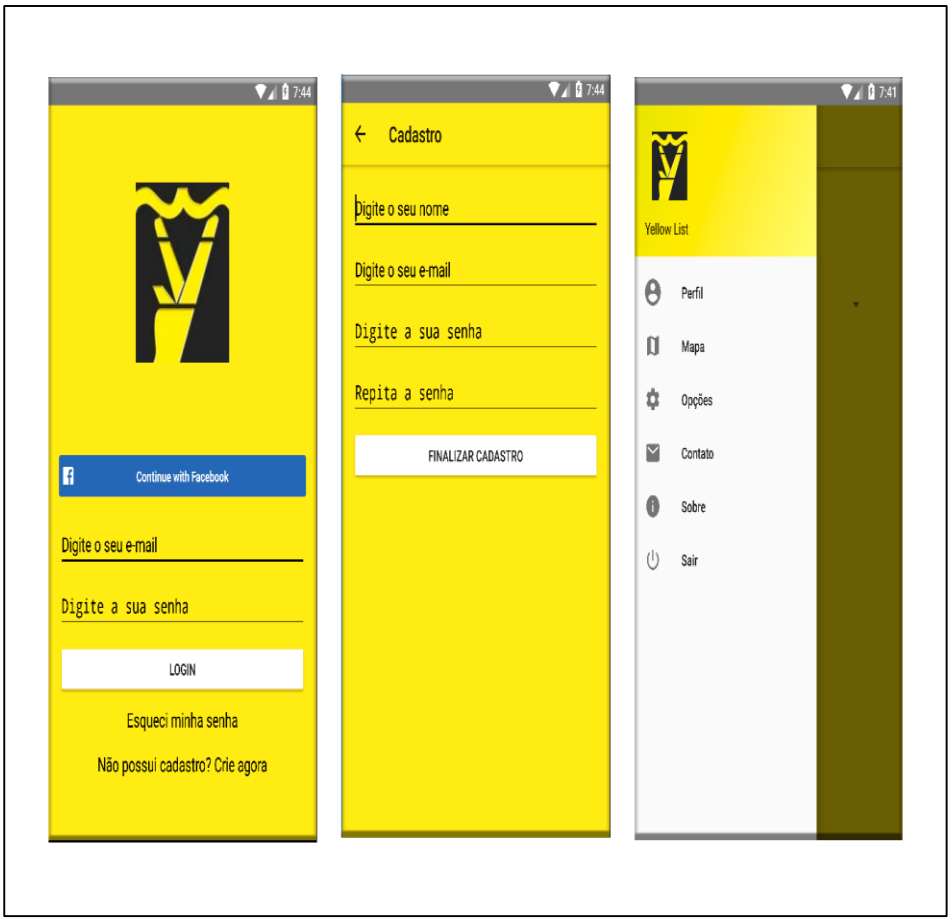
Evidenciou-se as vantagens da integração com a plataforma *Firebase* disponibilizada pelo *Google*, e o modo como ela facilita o desenvolvimento e a integração com as APIs mais utilizadas no momento, incluindo a API do *Facebook* utilizada no aplicativo.

Foi demonstrada a elaboração do ambiente de *back-end* que dá suporte a operação do aplicativo. Para isso utilizamos a topologia de *Web Service*, e mostramos as tecnologias envolvidas do desenvolvimento dessa aplicação que é independente, mas extremamente necessária ao funcionamento do aplicativo *Yellow List* como um todo.

Para finalizar é apresentado um protótipo com as funcionalidades básicas esperadas, incluindo o *Web Service* funcional para permitir as operações principais. O aplicativo conta com um *Layout* personalizado e de acordo com a identidade esperada para a marca *Yellow List*, incluindo o logotipo. As funcionalidades de *login* e gerenciamento de usuários já estão funcionais. É possível acessar a aplicação através de um cadastro básico fornecendo um *e-mail* valido e uma senha, ou então utilizar uma

conta do *Facebook* para ter a integração com a plataforma da rede social. A aplicação possui um menu lateral que permite ao usuário acessar as várias telas disponíveis, incluindo uma tela com as funcionalidades de mapas fornecida pela API do *Google Maps,* conforme figura 14.

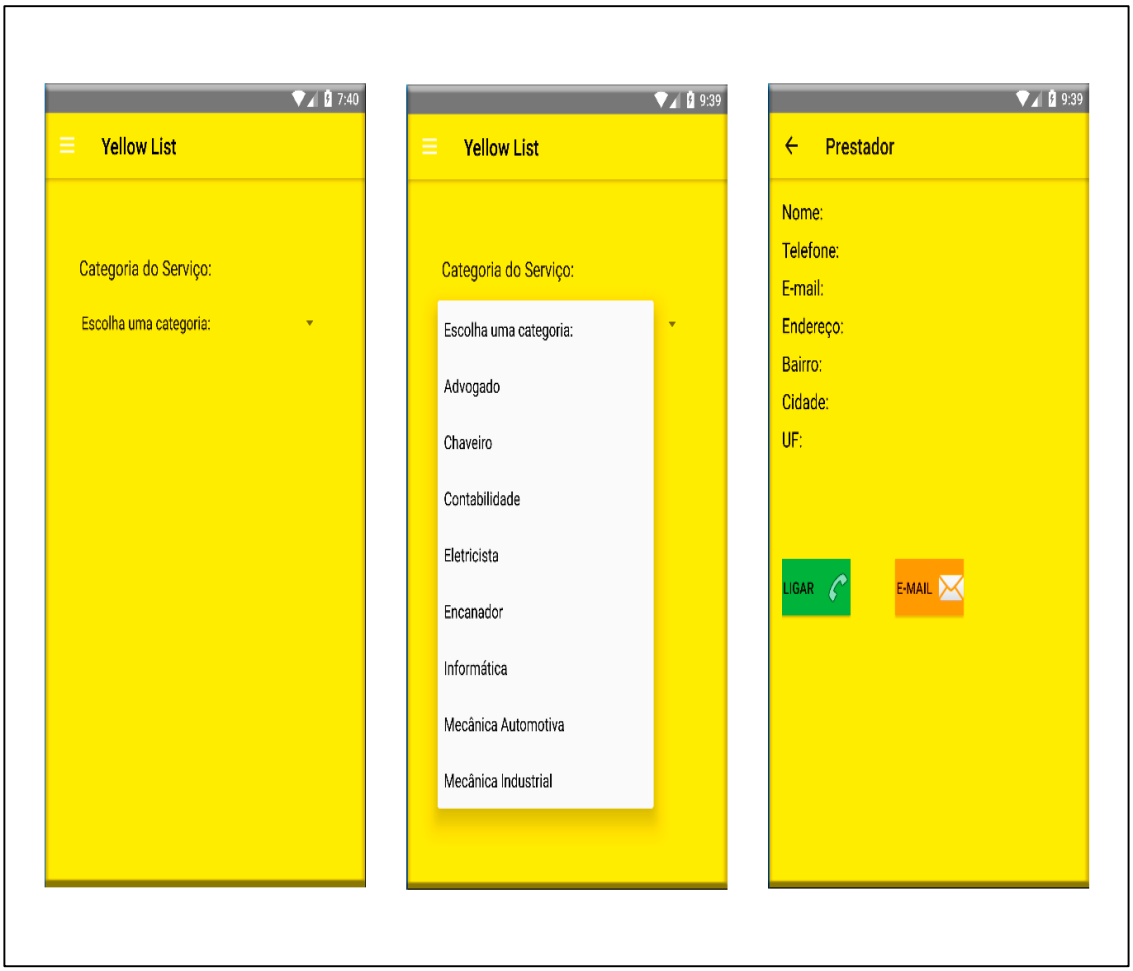
Figura 14 - Protótipo das telas de *login*, cadastro de usuário e menu lateral.



Fonte: Criado pelos autores em 2017.

A consulta a prestadores de serviço está funcional e retorna os registros de um banco de dados de teste preparado para auxiliar no desenvolvimento. Uma vez selecionado um prestador, é mostrada uma tela contendo as informações de contato e endereço, e é permitida a realização de uma chamada telefônica ou o envio de um e-mail ao prestador, conforme figura 15.

Figura 15 - Protótipo das telas de consulta a prestadores de serviço.



Fonte: Criado pelos autores em 2017.

Como proposta futura, continuarão os trabalhos no desenvolvimento da aplicação, refinando as funcionalidades já existentes no protótipo e adicionando novas para tornar a aplicação cada vez mais relevante. Será providenciado a elaboração do suporte comercial necessário, como o registro do nome e da marca, registro de domínio na *internet* e abertura da empresa para o permitir o início das atividades comerciais.

O *Yellow List* atende a um ramo do *e-commerce* que está em alta no momento e por isso muitas oportunidades estão disponíveis. Esperamos poder continuar e posicionar a marca como referência no mercado da região e além.

REFERÊNCIAS

AGÊNCIA NACIONAL DE TELECOMUNICAÇÕES (ANATEL). **Brasil registra 282,56 milhões de linhas móveis em operação em fevereiro de 2015.** Disponível em:http://www.anatel.gov.br/Portal/exibirPortalNoticias.do?acao=carregaNoticia&codigo=37095. Acesso em: 05/10/2016.

AMORIM, Maurício Alves; SANTOS, Eric de Oliveira. **Comunicação Móbile:** um estudo de caso sobre a Bigod.es software, a primeira agência a trabalhar com Marketing Móbile em Vitória da Conquista – BA, 2015. 25 f. Trabalho de Conclusão de Curso. Universidade Federal da Paraíba, João Pessoa. 2015

ANDRADE, Alisson Wilker; AGRA, Ronaldo; MALHEIROS, Viviane. **Estudos de caso de aplicativos móveis no governo brasileiro.** Brasília: SERPRO, p 780-791, 2012.

BRAGA, Jackson. **Padrão "inversão de controle com injeção de dependência":** Aplicações EJB "versus" spring framework. 2011. 48 f. Trabalho de diplomação (Tecnólogo em Desenvolvimento de Sistemas de Informação) – Universidade Tecnológica Federal do Paraná, Medianeira. 2011.

CERVO, Amado Luiz; BERVIAN, Pedro Alcino. **Metodologia Científica**: para uso dos estudantes universitários. São Paulo: McGraw-Hill do Brasil, 1983.

CHENTAO, Shen; YONGLE, Wang. **Online to offline business model:** Comparative study off chinese O2O companies. 2014. 94 f. Master Thesis – Halmstad University, Halmstad, Sweden. 2014.

DEITEL, Paul; DEITEL, Harvey; DEITEL, Abbey. **Android:** How to Program.2nd. edition. New Jersey: Pearson Education, 2015.

FIGUEIREDO, Andre Trindade. **Suporte a notas fiscais eletrônicas e integração com Facebook em aplicativo Android para gerenciamento de listas de compras colaborativas.** 2015. 56 f. Trabalho de Conclusão de Curso (Bacharelado em Ciência da Computação) – Universidade Federal do Rio Grande do Sul, Porto Alegre. 2015.

GANDHEWAR, Nisarg; SHEIKH, Rahila. Google android: an emerging software platform for mobile devices. **International Journal on Computer Science and Engineering**, edição especial, p. 12-17, 2010.

GERHARDT, Tatiana Engel; SILVEIRA, Denise Tolfo. **Métodos de Pesquisa**. Porto Alegre: Editora da UFRGS, 2009.

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.** 2016. 61 f. Trabalho de Conclusão de Curso (Bacharelado em Ciência da Computação) - Universidade Federal do Rio Grande do Sul, Porto Alegre. 2016.

INSTITUTO BRASILEIRO DE INFORMAÇÃO EM CIÊNCIA E TECNOLOGIA. **OJS em uma hora:** uma introdução ao sistema eletrônico de editoração de revistas SEER/OJS versão 2.1.1. Stanford, California: Public Knowledge Project, 2006.

KALIN, Martin. **Java Web Services:** Up and Running. 2nd Edition. Cambridge: O'Reilly, 2013.

LEMOS, André; JOSGRILBERG, Fabio. **Introdução**. In: Comunicação e Mobilidade: Aspectos Socioculturais das Tecnologias Móveis de Comunicação no Brasil. LEMOS, André; JOSGRILBERG, Fabio (organizadores). Salvador: EDUFBA, p. 7-11, 2009. 156 p.

LIMA JUNIOR, Guaratã Alencar Ferreira; SILVA, Rodrigo Cezario da. Guia de boas práticas para desenvolvimento de interface e interação para desenvolvedores da plataforma Android. **III Workshop de Iniciação Científica em Sistemas de Informação.** Universidade Federal de Santa Catarina: Simpósio Brasileiro de Sistemas de Informação, p. 31-34, 2016.

PELLANDA, Eduardo Campos. **Comunicação móvel no contexto brasileiro**. In: Comunicação e Mobilidade: Aspectos Socioculturais das Tecnologias Móveis de Comunicação no Brasil. LEMOS, André; JOSGRILBERG, Fabio (organizadores). Salvador: EDUFBA, p. 11-18, 2009. 156 p.

POSSOBOM, Camila Cerezer. **Estudo de caso:** *cloud computing* – computação em nuvem. 2010. 82 f. Trabalho de Conclusão de Curso (Bacharelado em Informática – Sistemas de Informação) - Universidade Regional do Noroeste do Estado do Rio Grande do Sul, Ijuí. 2010.

RAYNALDO, Camilus. **Android Programming Painless.** Kindle Edition. Tutorial Book, 2012.

RECH, Willian Rodrigues da Fonseca. **Comercialização de software em plataforma mobile:** um estudo de caso aplicado ao Android. 2013. 64 f. Trabalho de Conclusão de Curso (Especialização) – Universidade Federal de Santa Maria, Frederico Westphalen. 2013.

SILVA, Marcelo Moro; SANTOS, Marilde Terezinha Prado. Os paradigmas de desenvolvimento de aplicativos para aparelhos celulares. **Tecnologias, Infraestrutura e Software**, v. 03, n. 02, p. 162-170, 2014.

SOARES, Bruno Cesarino, **Requisitos para utilização de prototipagem evolutiva nos processos de desenvolvimento de software para Web.** Belo Horizonte: UFMG, 2007.

SOUZA, Dalva Inês; MÜLLER, Deise Margô; FRACASSI, Maria Angélica Thiele; ROMEIRO, Solange Bianco Borges. **Manual de orientações para projetos de pesquisa**. Novo Hamburgo: FESLSVC, 2013. 55 p.

VALERIANO, Dalton L. **Gerência em Projetos**. São Paulo: Makron Books, 1998.

WALLS, Craig; BREIDENBACH, Ryan. **Spring em Ação**. 2. ed. Rio de Janeiro: Alta Books, 2008.

Os autores declararam não haver qualquer potencial conflito de interesses referente a este artigo.