Francisco Jefferson Ferreira de Lima

WorkBook: Uma plataforma de contratação de prestação de serviços

Pau dos Ferros

2018

Francisco Jefferson Ferreira de Lima

WorkBook: Uma plataforma de contratação de prestação de serviços

Trabalho de conclusão de curso submetido ao Curso de Tecnologia em Análise e Desenvolvimento de Sistemas do Instituto Federal de Educação, Ciência e Tecnologia do Rio Grande do Norte, em cumprimento às exigências legais como requisito parcial à obtenção do título de Tecnólogo em Análise e Desenvolvimento de Sistemas.

Instituto Federal de Educação, Ciência e Tecnologia do Rio Grande do Norte – IFRN

Tecnólogo em Análise e Desenvolvimento de sistemas

Orientador: Me. Demetrios Araujo Magalhaes Coutinho

Pau dos Ferros

2018

Francisco Jefferson Ferreira de Lima

WorkBook: Uma plataforma de contratação de prestação de serviços

Trabalho de conclusão de curso submetido ao Curso de Tecnologia em Análise e Desenvolvimento de Sistemas do Instituto Federal de Educação, Ciência e Tecnologia do Rio Grande do Norte, em cumprimento às exigências legais como requisito parcial à obtenção do título de Tecnólogo em análise e desenvolvimento de sistemas.

Trabalho aprovado

Me. Demetrios Araujo Magalhaes Coutinho
Orientador

Prof. Lucas Hiago de Azevedo Dantas
Coorientador

Ieferson Queiroga Pereira
Examinador interno

Prof. Me. Francisco Sergio de Almeida Neto
Examinador interno

Pau dos Ferros Abril/2018

Agradecimentos

Por trás de toda conquista, existem aqueles que serviram como apoio, direção e motivação. À estes deve-se o devido reconhecimento. Começo agradecendo Àquele que me disse: "Eis que estou contigo todos os dias até a consumação dos séculos"(Bíblia sagrada, Mateus 28:20b). É nessa certeza, debaixo desse cuidado e banhado por esse amor que procuro viver minha vida, sabendo que Ele está no controle de tudo.

Agradeço imensamente à minha família, minha base. À meu pai por ser exemplo de dedicação e cuidado com os seus, à minha mãe por ser o abraço apertado nos tempos difíceis e ao meu irmão pelo companheirismo sempre presentes. Sei que sem estes não seria quem sou hoje e tenho orgulho em dizer que são tudo pra mim.

Aos meus amigos e colegas de curso que são parte importante dessa trajetória. Há um provérbio africano que diz: “Se quer ir rápido, vá sozinho. Se quer ir longe, vá em grupo”. Agradeço cada conselho ou palavra de consolo e até palavras duras proferidas por estes, sei que me trouxeram força e motivação para continuar à cada minuto.

Ao IFRN, instituição que aprendi à amar e tenho orgulho em servir. Aos diretores, professores, técnicos e terceirizados que mostram dia a dia que trabalhar com amor e dedicação traz resultados acima do esperado. Agradeço e presto honra à este lugar que posso chamar de lar.

Por fim, agradeço ao meu amigo e orientador, Demetrios Araujo Magalhães Coutinho. Existem pessoas que estão dispostas à fazer a diferença onde estão e ter mostrado isso desde a primeira aula no curso de TADS me fez admirar sua postura e índole. Sei que dei muito trabalho ao longo desse tempo e agradeço pela paciência e por ter estado do meu lado até aqui. Fico feliz em saber que a relação de aluno e professor atravessou as paredes da universidade e se tornou uma parceria para a vida.

Felicidade?
Disse o mais tolo: "Felicidade não existe."
O intelectual: "Não no sentido lato."
O empresário: "Desde que haja lucro."
O operário: "Sem emprego, nem pensar!"
O cientista: "Ainda será descoberta."
O místico: "Está escrito nas estrelas."
O político: "Poder"
A igreja: "Sem tristeza? Impossível.... (Amém)"

O poeta riu de todos,
E por alguns minutos...
(Felicidade - O Teatro mágico)

Resumo

O setor de prestação de serviços é o setor com maior participação na economia brasileira. Porém, por vezes torna-se difícil encontrar um prestador de serviço através dos meios convencionais de propaganda e, quando encontrado, muitas vezes não se encontram informações sobre a qualidade dos serviços prestados. Faz-se necessária então uma ferramenta que se apresente como solução para estas dificuldades. Diante deste cenário, este trabalho descreve o desenvolvimento de um sistema composto por um aplicativo *mobile* e um servidor REST que tem como premissa servir de interface de comunicação entre prestadores de serviço e seus clientes: o Workbook. Através do aplicativo, os usuários poderão pesquisar serviços, ver avaliações de outros usuários sobre cada prestador, solicitar orçamentos e conversar diretamente com prestadores. Esta ferramenta demonstra-se capaz de melhorar a interação entre clientes e prestadores de serviços, através de uma comunicação transparente e orçamentos bem definidos, estabelecendo uma confiança entre o contratante e o prestador.

Palavras-chaves: Prestação de serviços. Aplicativo Mobile. Servidor REST.

Abstract

The service sector is the sector with largest participation in Brazilian economy. However, sometimes it's difficult to find a service provider through conventional ways of advertising and, when it's found, many times the client can't find information about the quality of the services provided. The population needs a solution to these difficulties. Considering this scenario, this work describes the development of a system composed of a mobile app and a server that has a premisse to act as a communication interface between service providers and their clients: the Workbook. Through the application, users will be able to search services, view other users' reviews of each provider, request budgets, and talk directly to providers. This app is capable of improving the client-provider interaction by transparent comunication and well defineds budgets, establishing trust between customer and provider.

Key-words: Services providing. Mobile App. REST Server

Lista de ilustrações

Sumário

1 Introdução

Ao longo dos anos, o setor da prestação de serviços vem crescendo no Brasil. A prestação de serviços é o setor com maior participação na economia brasileira, no 4º trimestre de 2017, o setor de serviços representou 75,2% do valor adicionado do PIB brasileiro (DATASEBRE, 2016). Porém, ainda existem deficiências nesse setor. Em algumas cidades e regiões, usuários podem encontrar dificuldades para contratar serviços terceirizados de boa qualidade. Por mais que exista divulgação como *outdoors*, carros de som ou por estações de rádio e televisão, não há como saber se o serviço será realizado com um bom custo benefício. Esse é um setor que exige confiança entre o cliente o prestador do serviço. Por isso, o objetivo desse trabalho é desenvolver um aplicativo móvel, nomeado de WorkBook, que sirva de meio de comunicação entre prestadores e consumidores, centralizando a divulgação para que os moradores da região possam buscar qualquer tipo de serviço em um único lugar de forma confiável e eficiente.

Não é difícil notar que o mundo está cada vez mais *mobile*. A quantidade e a diversidade de aplicativos existentes para dispositivos móveis e a facilidade de utilizar esses recursos em qualquer lugar a partir de *smartphones*, traz um grande número de adeptos à essa tecnologia. O mercado não fica de fora dessa realidade. Existem inúmeros aplicativos voltados para o comécio, seja na área de compra e venda (Mercado Livre, OLX), alimentação (iFood) ou de prestação de serviços (Uber, App 99). O crescimento de aplicativos como esses impulsiona o crescimento do M-commerce, nome dado ao E-commerce mobile, no Brasil. Alguns dados importantes podem ser visto em pesquisa da Sociomantic (2016) que mostra que o valor gasto em M-commerce no Brasil subiu de R$13.2bi em 2014 para R$27.bi em 2015 e que em 2015 12% das compras no e-commerce foram feitas por celular. Outro dado relevante desta pesquisa é que 67% dos usuários já fizeram compras pelo celular. Dessa forma, demonstra o quanto que é importante desenvolver aplicativos da área de comércio para *mobile*.

No mercado, existem aplicativos que prestam tais serviços, 3 podem ser citados como referência: Achou App, Já liguei e GetNinjas. Os dois primeiros não permitem avaliação de clientes. Esta limitação faz com que o usuário não tenha uma referência real sobre o trabalho do prestador, tornando menos confiável a contratação do serviço. No GetNinjas, o usuário coloca o serviço que deseja realizar, então os prestadores são notificados e os interessados entram em contato com o usuário. O aplicativo promete enviar o pedido para 5 prestadores de serviço. Essa forma de trabalhar possui duas limitações. Se o algoritmo do aplicativo não escolher um determinado prestador, ele não terá a oportunidade de entrar em contato com o usuário e realizar o serviço. Do outro lado, limita a escolha do usuário, pois o mesmo não pode escolher diretamente um prestador

para contratar caso este não tenha recebido a solicitação do usuário.

No workbook, o usuário ao pesquisar um serviço tem acesso às informações do prestador, aos demais serviços que ele presta e às avaliações de outros usuários. O próprio usuário decide qual prestador solicitará o orçamento e, pode contatá-lo para trocar informações necessárias sobre o serviço. O foco do workbook é ser simples de utilizar e permitir liberdade ao usuário de escolher o prestador que mais lhe convém, além de que as avaliações e opiniões de outros usuários trazem confiabilidade tanto aos prestadores como aos usuários.

A princípio, um projeto piloto será aplicado na cidade de Pau dos Ferros, pois não há plataformas com o mesmo objetivo que atuam na cidade e isso deixa o mercado aberto para a chegada de uma solução para a prestação de serviços. Além disso, a cidade recebe um fluxo constante de pessoas oriundas de outras cidades circunvizinhas por ser um polo comercial da região. Acrescentando, Pau dos Ferros é uma cidade universitária, possui três instituições públicas de nível superior, IFRN[1], UERN[2] e UFERSA[3] e, também duas faculdades particulares Anhanguera e FACEP[4] o que aumenta ainda mais o fluxo de pessoas que precisam morar na cidade e consequentemente necessitarão de alguma prestação de serviço.

Esse trabalho está dividido como se segue. Este capítulo da introdução, seguido pelo capítulo sobre pesquisa de mercado realizado em Pau dos Ferros. O terceiro capítulo descreve as tecnologias utilizadas no desenvolvimento do workbook. O quarto capítulo mostra a arquitetura da aplicação e descreve os diagramas UML utilizados no desenvolvimento do sistema. No quinto capítulo há uma apresentação do sistema, mostrando as suas principais funcionalidades. Por fim, a conclusão aborda as considerações finais do desenvolvimento do workbook.

1.1 Objetivos gerais

Oferecer uma solução para a busca de serviço que facilite oferta e busca de serviços, além de possibilitar a interação entre usuários e prestadores de serviços. A solução deverá indicar a confiabilidade e qualidade no serviço dos prestadores, afim de auxiliar o usuário na escolha do melhor profissional.

[1] Instituto Federal de Educação, Ciência e Tecnologia do Rio Grande do Norte
[2] Universidade Estadual do Rio Grande do Norte
[3] Universidade Federal Rural do Semi-Árido
[4] Faculdade Evolução Alto Oeste Potiguar

1.2 Objetivos específicos

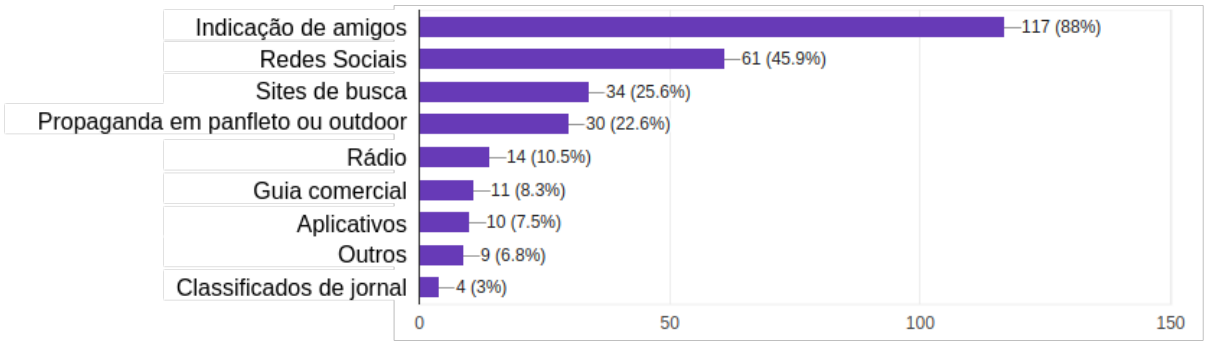
Desenvolver um aplicativo para auxiliar a busca de prestadores de serviços e prover a interação entre cliente e prestador. A solução deve possibilitar o envio de orçamentos ao cliente e a troca de mensagens entre cliente e prestador. O aplicativo deverá também possibilitar a avaliação do prestador, através de nota e comentário que ficarão visíveis aos demais usuários e utilizar essa avaliação para dar mais visibilidade aos prestadores mais bem avaliados.

2 Pesquisa de mercado

Antes de se iniciar o desenvolvimento do workbook, fez-se necessário buscar informações detalhadas sobre como as pessoas buscam e consomem serviços. Realizou-se então uma pesquisa, via internet e com 12 questões de múltipla escolha com 133 moradores de Pau dos Ferros e região. A pesquisa contou com questões sobre a busca e qualidade dos serviços e sobre o uso de aplicativos para celular de propósito geral e focado na área financeira.

O primeiro dado a ser destacado em relação à pesquisa foi que 82% dos entrevistados afirmaram que já tiveram dificuldade de encontrar prestadores de serviços. A pesquisa também revelou a forma de encontrar prestadores mais utilizada é a indicação de amigos, que obteve 88% dos votos, em segundo lugar, com 45%, redes sociais, em terceiro ficaram os sites de busca, com 34%. O Gráfico da figura 1 mostra a proporção das respostas.

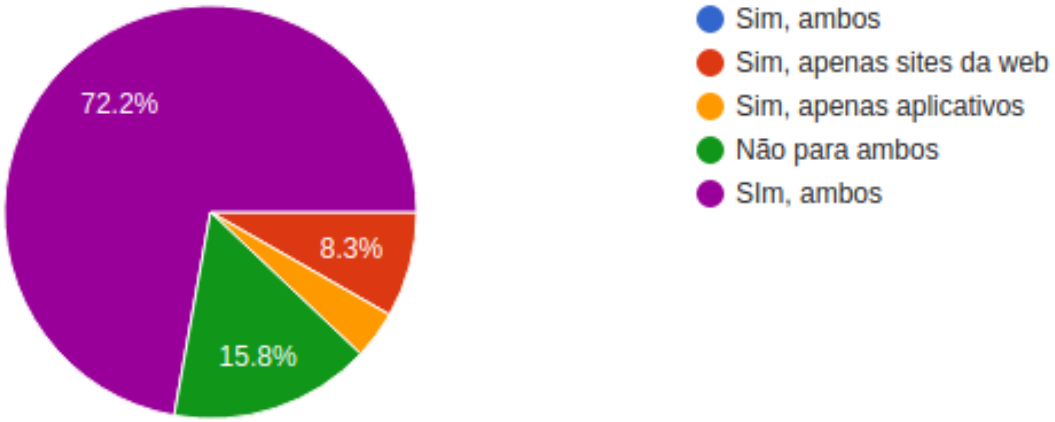
Figura 1 – Meios utilizados para encontrar serviços



Fonte: Pesquisa de mercado

Considerando que quase metade dos entrevistados utilizam redes sociais e um terço utilizam aplicativos para buscar prestadores de serviços, pode-se afirmar que boa parte dos moradores da região costuma utilizar a internet para tal atividade. Essa constatação é reafirmada pela figura 2, que mostra quantos usuários utilizariam aplicativos ou sites para buscar serviços.

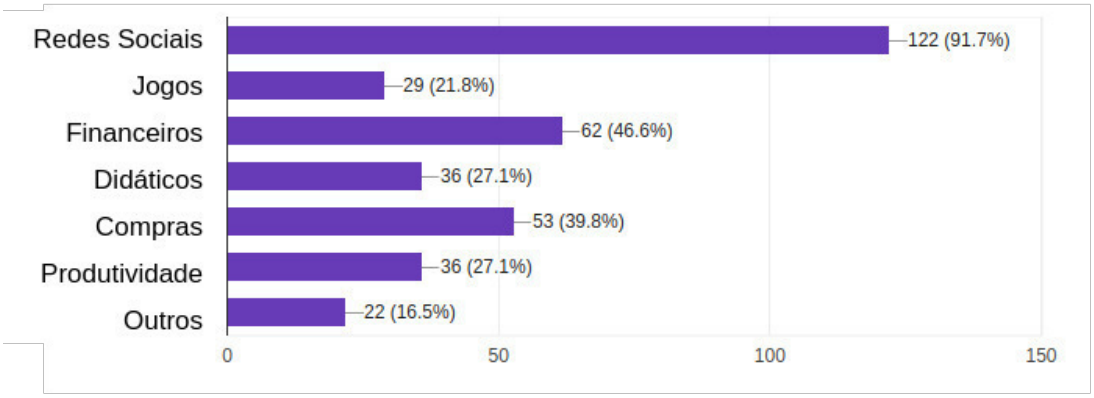
Figura 2 – Entrevistados que utilizariam ou não sites e aplicativos para busca de serviços



Fonte: Pesquisa de mercado

Pode-se notar, através da figura 3, que uma parcela significativa da população de Pau dos Ferros e região utiliza aplicativos para fins comerciais e aplicativos financeiros. 91,7% utilizam de redes sociais. Estes dados embasam a ideia de que um aplicativo para busca de serviços, onde os usuários podem interagir com os prestadores e ler opiniões e comentários de outros usuários pode ser bem recebido pela população.

Figura 3 – Aplicativos utilizados pelos entrevistados



Fonte: Pesquisa de mercado

A figura 1 mostra que indicações de outras pessoas são levadas em consideração por 88% dos entrevistados. Quando perguntados se gostariam de ver opiniões de pessoas sobre um determinado serviço, 99% responderam que sim. Quando perguntados se recomendariam um serviço de qualidade, a resposta positiva foi unanimidade.

Essas informações ajudam a traçar o perfil do consumidor. Nota-se que são habituados ao uso de aplicativos de celular, que são receptivos à uso de aplicativos voltados

para a área financeira e que estão dispostos à consumir e partilhar informações sobre os serviços utilizados.

Esta pesquisa também serviu de auxílio para a elaboração dos requisitos do workbook. Pesquisas de mercado são estratégias comuns em levantamento de requisitos de software.

> Em projetos onde é desejável obter informações de um grande número de pessoas, um questionário pode ser uma ferramenta valiosa. Um questionário, também chamado de pesquisa, é um documento contendo um número de questões padrão que podem ser enviadas para muitos indivíduos. Os questionários podem ser usados para obter informações sobre uma ampla gama de tópicos, incluindo cargas de trabalho, relatórios recebidos, volumes de transações tratadas, dificuldades e opiniões de como o trabalho poderia ser realizado melhor ou mais eficientemente.(SHELLY; ROSENBLATT, 2011)

Embora a pesquisa seja voltada para a necessidade e aceitação da ferramenta, os dados levantados foram base para definir necessidades dos potenciais usuários do workbook.

3 Fundamentação Teórica

Neste capítulo serão abordado conceitos fundamentais das tecnologias utilizadas no desenvolvimento do projeto. Os principais aspectos tratados são relacionados ao desenvolvimento para dispositivos móveis, comunicação cliente servidor e autenticação.

3.1 Aplicativos Móveis

Um dos fatores determinantes quando se desenvolve software é a plataforma onde ele irá ser executado, Windows, Linux, Web e sistemas embarcados são exemplos conhecidos de plataformas de desenvolvimento. Após a popularização dos smartphones, surgem então novas plataformas, como Android, iOS e Windows Phone. Dessa forma, torna-se necessário o desenvolvimento de softwares para dispositivos móveis

Existem duas formas de desenvolver aplicativos móveis: aplicações nativas e híbridas. Aplicações nativas são construídas utilizando linguagens específicas de cada plataforma. Lopes (2016) afirma que as plataformas nativamente oferecem a possibilidade de criar aplicativos. Usando o *Android SDK* e a linguagem *Java*, pode-se desenvolver para o sistema do Google. A Apple oferece ferramentas para iOS e permite usar *Objective-C* ou *Swift*. No Windows Phone, usamos C# e toda suíte de desenvolvimento *Microsoft*. Cada plataforma tem sua combinação de linguagem e, principalmente, APIs específicas.

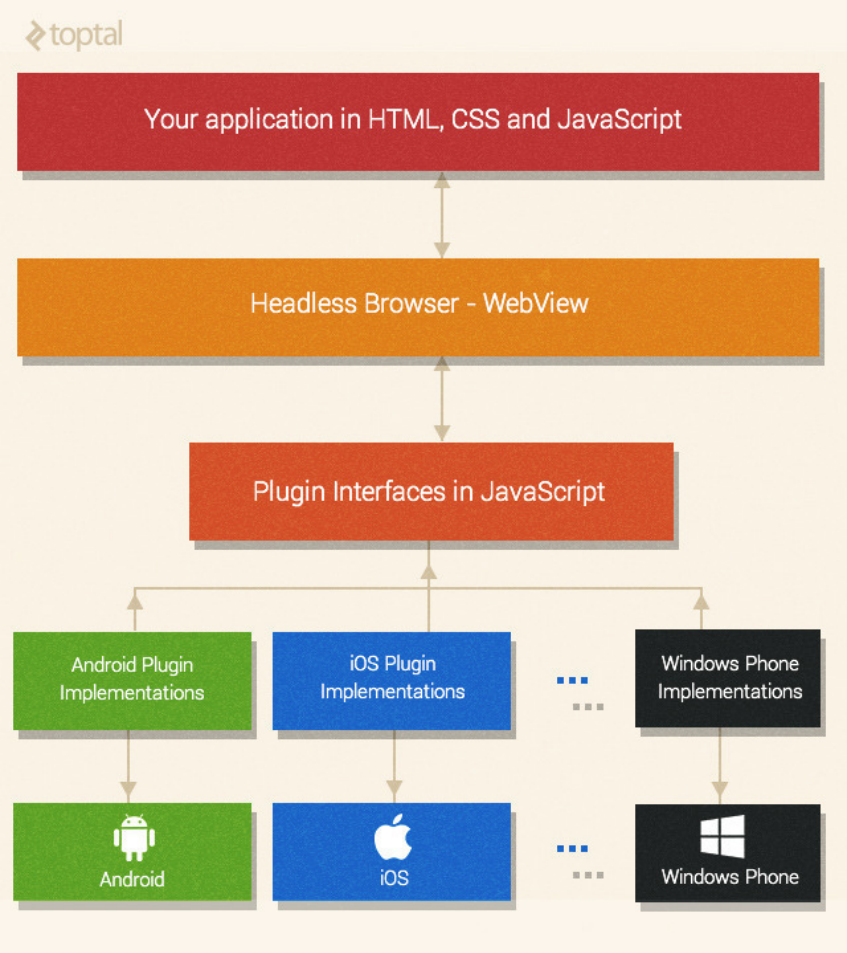
Aplicações híbridas são aplicações feitas em código web que rodam sobre uma *webview* dentro da plataforma móvel.

> A WebView permite a exibição de páginas da web que utiliza o mesmo engine do navegador disponível no Android. Ou seja, quando incluímos uma WebView em nosso aplicativo, temos praticamente todos os recursos do navegador padrão.(MONTEIRO, 2014)

No desenvolvimento híbrido, ocorre uma chamada nativa dessa webview para que o código web possa ser executado sobre a mesma. Na figura 4 podemos ver a arquitetura de uma aplicação híbrida.

Na primeira camada, está o código web desenvolvido pelo programador. Esse código é executado pela *webview*, que é a segunda camada da arquitetura. A camada seguinte é um conjunto de *plug-ins* que proveem acesso à funcionalidades do sistema operacional nativo, como GPS, câmera, armazenamento interno e etc. Na quarta camada encontram-se implementações específicas de cada plataforma para que os plugins da terceira camada possam ser executados. Por fim, está o sistema operacional.

Figura 4 – Arquitetura de uma aplicação híbrida



Fonte: (TOPTOTAL, 2015)

A vantagem em se desenvolver de forma híbrida está na portabilidade e no baixo custo tanto de desenvolvimento quanto de manutenção. Como as webviews em diferentes plataforma apresentam características muito semelhantes, o mesmo código web pode ser executado em diversos dispositivos. Para isso, é necessário apenas uma aplicação nativa que crie a webview. Existem diversas ferramentas responsáveis por compilar esse código nativo, sem a necessidade de que o programador o faça. Um exemplo dessa ferramenta é o cordova.

3.2 Apache Cordova

O Apache Cordova é um um *framework*[1] para desenvolvimento de aplicações multi-plataforma. Sua função é encapsular o código web em uma aplicação nativa. Quando um projeto é compilado no cordova para uma plataforma específica, ele se torna o responsável pelo código nativo que cria uma instância da *webview*. No fim, ele reúne tudo em um

[1] Framework é abstração que une códigos comuns entre diferentes softwares e aplicativos. Eles trabalham provendo funcionalidades genéricas, assim o desenvolvedor não terá que escrever código repetitivo

único aplicativo, contendo tanto o código que executa a webview quanto o código web a ser executado.

É possível descartar o uso do cordova ou qualquer outra plataforma do tipo. Para isso, basta criar de forma nativa a chamada da webview na qual o código web será executado. Porém, dessa forma deve-se escrever um código nativo solicitando uma *webview* na qual o código web será executado. Ou seja, é necessário ter diferentes aplicativos para diversas plataformas devido ao código nativo específico inerente. Com o cordova não há sequer a necessidade de conhecer a plataforma onde o aplicativo irá ser executado. Esse trabalho é de responsabilidade do framework, deixando o programador se preocupar apenas com a aplicação web, suas funcionalidades e tecnologias.

3.3 AngularJS

AngularJS é um *framework* Javascript que provê recursos e facilidades para o desenvolvimento Web.

> AngularJS é um framework Javascript que simplifica o desenvolvimento de aplicações web dinâmicas robustas, viabilizando a implementação do conceituado modelo MVC (Model-View-Controller). Ele apresenta características satisfatórias tais como: produtividade, desempenho, fácil customização, é testável e plugável, implementa o fantástico conceito de diretivas, ampliando o vocabulário HTML, suporta o desenvolvimento de módulos, implementa o revolucionário Two-Way Data Binding(PEREIRA, 2014)

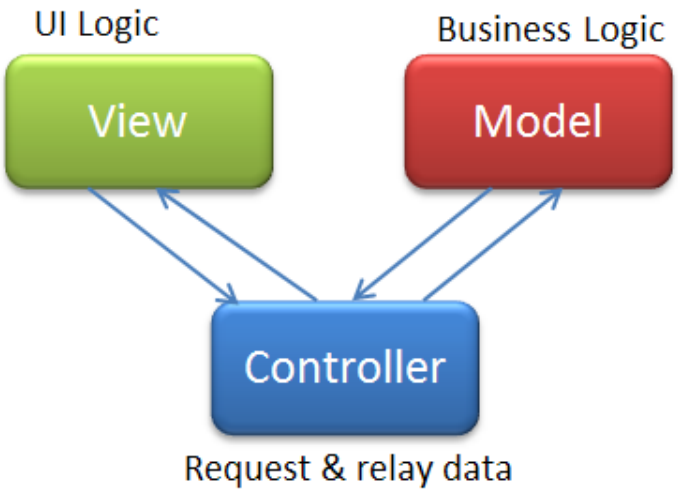
3.3.1 MVC

O padrão MVC divide o software em três camadas: Model, View e Controller. Cada qual é responsável apenas por executar o que lhe é definido.

- View - Quando o usuário acessa um sistema, é a camada View que ele está vendo. É a camada responsável pela interação com o usuário e define como a interface sistema irá se comportar de acordo com as ações do usuário;
- Model - A camada de model é responsável pela definição e manipulação de dados. Nessa camada se define as tabelas e relacionamentos do banco de dados de um sistema.
- Controller - É responsável pela comunicação entre as camadas, fazendo com que os dados sejam acessíveis pela camada de apresentação.

O conceito de MVC do lado cliente[2] muda a forma como as coisas funcionam na web. Em vez de haver Model e Controller rodando no servidor e a View executando no cliente, todas as camadas estão no lado cliente. A figura 5 define a arquitetura deste padrão.

Figura 5 – Arquitetura MVC



Fonte: (CODEPROJECT, 2018)

Controllers no AngularJS, assim como em todos os frameworks MVC, são responsáveis por fazer a interação entre a View e o Model. Segundo Pereira (2014), um controller também é parte do MVC. Seu principal propósito é intermediar a view e o model - visão e modelo - de tal forma que consiga alterar o estado do model e intermediar a propagação deste estado para a view. No angularJS, pode-se definir, na View, qual controller será responsável por cada trecho de código. Pode-se assim, por exemplo ter um Controller específico para uma listagem de usuários e um Controller diferente para o menu da página.

As Views dentro do AngularJS são criadas em HTML. Porém, o AngularJS apresenta o conceito de diretivas. Diretivas atuam como uma extensão do HTML e permite criar novos comportamentos, componentes reutilizáveis. Existem diretivas que simulam estruturas de repetição dentro da página, definem o Controller que interage com a View. Dessa forma, torna-se fácil definir como a página HTML irá se comportar. Porém, o ponto forte do AngularJS está na interação entre a View e o Model: o *Two-Way Data Binding*.

O Two-Way Data Binding define a forma como a interação entre View e Model acontece.

> Two-Way Data Binding possibilita implementar um binding de uma variável a um elemento da view de tal forma que, ao alterar o elemento na memória, por meio de JavaScript, a view seja alterada automaticamente, sem interferência alguma. Da mesma forma, ao alterar o elemento na view,

[2] Em aplicações cliente-servidor, o lado cliente (*client-side*, em inglês é aquele pelo qual o usuário tem acesso ao sistema. Por exemplo, o cliente de um site é o navegador)

> o valor da variável na memória também é alterado automaticamente. É uma alteração automática de 'duas vias'"(PEREIRA, 2014)

3.3.2 AngularJS Material

Pode-se definir AngularJS Material tanto como um framework de interface gráfica ou como uma implementação do Google Material Design. O Google Material Design é um framework gráfico criado com a intenção de simplificar e design e implementação. Ele provê uma união fácil entre estilo, interação e movimento. O AngularJS Material possui um conjunto de componentes gráficos reutilizáveis baseado no Google Material Design. Seus componentes são responsivos, ou seja, adaptam-se à tamanhos diferentes de telas e janelas. Por ser implementado em AngularJS, sua configuração, seu comportamento e seus eventos são definidos através de controllers AngularJS.

3.4 REST

REST é um acrônimo para *Representational State Transfer* (Transferência de Estado Representativo). REST é uma forma de se desenvolver web services para prover recursos para aplicações clientes. Web services são aplicações que tem como única finalidade prover dados para outras aplicações. Saudate (2014) define que o REST é guiado pelo que seriam boas práticas de uso de HTTP:

- Uso adequado de métodos HTTP;
- Uso adequado de URL's;
- Uso de códigos de status padronizados para representação de sucessos ou falhas;
- Uso adequado de cabeçalhos HTTP;
- Integração entre vários recursos diferentes;

Qualquer conjunto de dados que trafega pelo protocolo é um recurso. O REST provê uma interface de acesso à esses recursos, através de URL's[3] e utilizando o protocolo HTTP para essa comunicação.

O REST utiliza métodos HTTP para prover a comunicação entre cliente e servidor, tendo como meta criar, recuperar, alterar ou apagar recurso. Entre estes métodos, podem se destacar: o método POST do HTTP é utilizado na criação de um recurso; O método GET recupera um ou mais recursos; O método PUT altera um recurso e o método DELETE apaga um recurso.

[3] *Uniform Resource Locator*. É o endereço pelo qual se localizam recursos em uma rede, tais como sites e servidores

3.5 Python

Python é uma linguagem interpretada, de alto nível e de propósito geral. É compilado automaticamente para o *bytecode* e executado, o que torna o Python adequado para uso como linguagem de script, implementação de aplicativos da Web, etc. O Python permite escrever aplicativos claros e lógicos para pequenas e grandes tarefas, por sua sintaxe simples e seu uso consistente de objetos e programação orientada para objetos.

Devido à sua sintaxe de simples entendimento, tipagem forte e dinâmica, e o uso de identação para definição de blocos, o Python se tornou uma das linguagens mais utilizadas no mundo. O site PYPL colocou, em fevereiro de 2018, o Python como a segunda linguagem de programação mais utilizada no mundo, atrás apenas do java.

Python também é conhecida por sua portabilidade, pois é executada sobre um interpretador. Interpretadores são softwares responsáveis por transformar o código de uma linguagem específica em código de máquina, que é executado pelo processador. Por isso, o Python pode ser executado em diferentes plataformas e ambientes, sejam PCs com Windows ou Linux ou microcontroladores como o *Orange Pi*. Além disso, há um poderoso framework python voltado para a web: O *Django*.

3.6 Django

Django é um framework para desenvolvimento web focado em produtividade. Holovaty e Moss (2009) afirmam que Adrianand Simon desenvolveu uma estrutura de desenvolvimento web que economizava tempo por necessidade. O Django se encarrega de encapsular rotinas repetitivas dentro do desenvolvimento para que o programador preocupe-se apenas com as especificidades do software. Supondo que o programador esteja desenvolvendo uma página que acessa um banco de dados. Ele precisará abrir conexão com o banco, realizar a query, aplicar a lógica de negócios e fechar a conexão. Caso precise de outra página, precisará repetir todo o processo, gerando código similar. O Django torna essa, entre outras atividades, menos repetitiva e mais produtiva. Através de padrões, o Django toma para si tais atividades repetitivas. Sendo assim, após o programador definir sua estrutura de dados e configurar suas URL's, ele necessitará apenas trabalhar sua lógica de negócio, enquanto o django se encarregará de abrir conexão com o banco, realizar a query e fechar a conexão. Além de aumentar a produtividade, este comportamento do django torna o desenvolvimento menos suscetível à erros.

Frameworks web atuais tem como premissa a separação entre apresentação, lógica de negócio e modelo de dados. O padrão MVC é um exemplo disto. O Django utiliza o padão MTV (Model-Template-View) para tal separação.

Model, dentro do MTV, é a camada de acesso aos dados. Nessa camada são definidos

o modelo de dados, acesso, validação e os relacionamentos entre os dados. O django se encarrega de criar tabelas e relacionamentos no banco de dados através do que é definido nas classes do Model. Sendo que as classes Python criadas se transformam em tabelas e os atributos são campos do banco. Esse tipo de comportamento é chamado de *Object Relational Mapping* (ORM).

Além de realizar o mapeamento do banco de dados, a camada model ainda fornece funções de busca, filtro, ordenação, alteração, inserção e validação de dados. Esse tipo de abstração diminui probabilidade de erros e principalmente torna desnecessário qualquer código de conexão, alteração ou busca no banco. Em sistemas grandes construídos sem esse tipo de framework, esse código costuma ser extenso e repetitivo.

Template é a camada de apresentação do MTV. É essa camada que o usuário enxerga quando utiliza o sistema. Ela define a forma como os dados serão exibidos e como o usuário vai interagir com o programa.

View é a camada responsável pela lógica de negócio. É responsável por acessar o Model e enviar dados ao Template ou, em caminho inverso, receber dados inseridos pelo usuário através do Template e enviar para o Model realizar a persistência dos dados. Outro papel fundamental da view está na segurança dos dados. Nela são definidas quais permissões serão necessárias para leitura, inserção e alteração de dados. A View pode ser considerada uma ponte entre Template e Model.

Views no Django são acessadas através de URLs. O django possui um arquivo de configuração de URLs que define qual View será chamada para cada URL e define quais parâmetros serão enviados para a View. A view então, renderiza o template, passando para ele, quando necessário, dados oriundos do banco de dados ou qualquer outra informação necessária.

Uma ferramenta poderosíssima presente no django é o Django Admin. Holovaty e Moss (2009) afirmam que essa ferramenta trabalha lendo os metadados do model para prover uma poderosa interface que os administradores do site podem utilizar imediatamente. A partir de uma simples configuração, as tabelas criadas estão disponíveis no Django Admin. Essa ferramenta é útil para testes, iniciação de dados e possui uma interface simples de usar.

3.6.1 Django REST framework

O Django REST Framework traz a possibilidade de utilizar REST dentro do framework Django. Em um site Django que não utiliza REST, o servidor recebe a requisição e acessa a View referente à URL acessada. A view, por sua vez, acessa o model para recuperar os dados buscados e os entrega para o Template. O servidor então retorna uma página HTML, definida pelo template e com os dados recuperados pela view.

O Django Rest framework pode aproveitar o máximo uma das maiores características do REST: o trafego somente de dados. Sendo assim, o software cliente tem a liberdade de exibir e manipular os dados de forma mais correta, sem se prender à uma interface gráfica já existente. Além disso, o Django Rest Framework especifica os dados em formato de texto no padrão chave-valor, o que reduz o tamanho dos dadosa serem enviados

O que o django REST framework trabalha com o formato JSON. JSON (JavaScript Object Notation) é um formato de dados específico para tráfego em rede. É simples de se entender, pois trabalha na forma de chave:valor e leve de se transportar pela rede, pois é apenas texto puro.

3.7 OAuth2

OAuth2 é um protocolo de segurança utilizado para conectar entre diferentes sites ou aplicativos entre si ou com serviços na nuvem. É utilizado por grandes companhias como Google e Facebook para disponibilizar suas API's para aplicativos de terceiros.

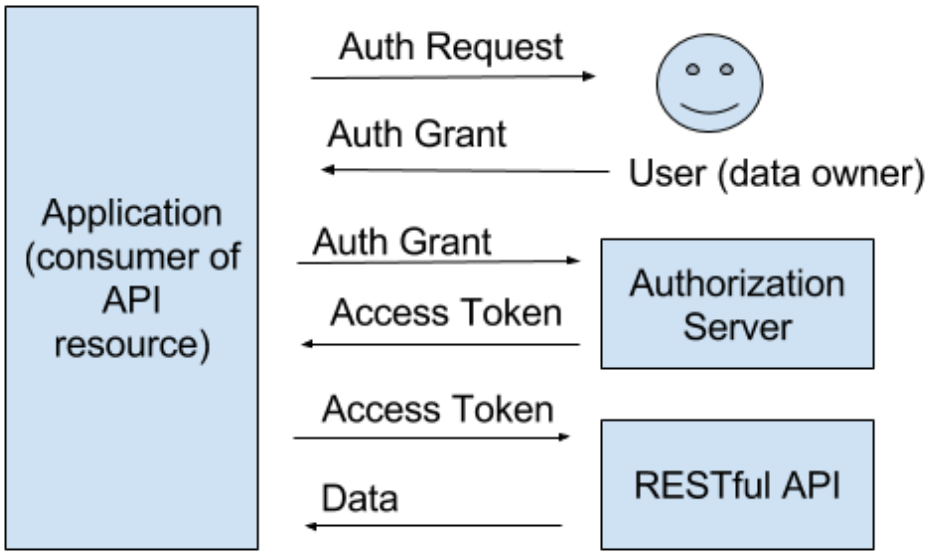
Antes que uma aplicação possa acessar um serviço protegido por OAuth, é necessário que sejam criadas credenciais de autenticação, o *client id* e o *client secret*. Estes dados são responsáveis por identificar a aplicação cliente que está acessando o serviço. Tais dados serão utilizados para gerar sequências de caracteres responsáveis que identificaram usuários da aplicação. Essas sequências de caracteres recebem o nome de token de autenticação são sequências de caracteres. Um token possui um tempo limite para ser utilizado, que é definido na hora de sua criação. Após esse tempo, é necessário a renovação do token.

Além de identificar a aplicação que está acessando o servidor, também é necessário identificar o usuário e aplicar-lhe as devidas permissões. Para isso, o OAuth cria tokens de autenticação cada vez que um usuário efetua login e atribui esse token ao usuário.

A figura 6 ilustra o funcionamento do OAuth2. Primeiro, o cliente faz uma requisição das informações necessárias ao login: client secret e client ID (Na figura, representados pelo Auth Grant). Logo após, ele envia suas informações de login (usuário e senha) e recebe um token caso as informações sejam válidas. Esse token identifica o usuário. A partir desse momento, todas as requisições feitas ao servidor deverão conter este token de autenticação em seu cabeçalho HTTP. Requisições sem esse token apresentaram erro 401, indicando que não há autorização para acessar tal dado.

Interessante notar que o OAuth2 não trabalha com o conceito de sessão, como é comumente utilizado em progamação web. O OAuth2 trata cada requisição individualmente, utilizando o token como forma de identificação do usuário e aplicando-lhe as permissões devidas.

Figura 6 – Funcionamento do protocolo OAuth2



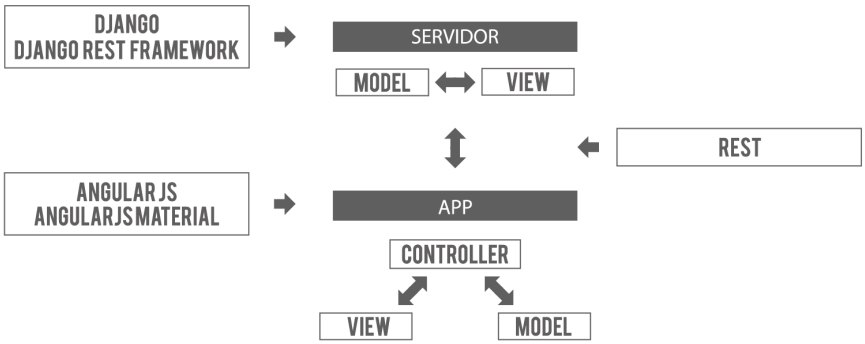
Fonte: (PARTNERS, 2016)

4 Desenvolvimento

Este capítulo descreverá como foi desenvolvido o projeto, abordando arquitetura, comunicação e as especificidades do app mobile e do servidor REST. Também serão mostrados os diagramas e a modelagem do sistema.

4.1 Arquitetura

Figura 7 – Arquitetura do Workbook



Fonte: Elaborado pelo autor

A figura 7 mostra a arquitetura do Workbook. O projeto se divide em servidor e Aplicativo móvel híbrido. Aplicações híbridas desenvolvidas com o *Apache Cordova* tem uma característica em comum, apenas código HTML, CSS e *Javascript* pode ser utilizado. Nesse cenário, é necessário que haja um servidor web externo para armazenar e prover dados. Sendo assim, a arquitetura do workbook foi desenvolvida para se trabalhar com o *Apache Cordova*. Para facilitar o desenvolvimento do aplicativo e a sua comunicação com o servidor, foi escolhido o *framework AngularJS*.

O fator crucial da arquitetura é a comunicação entre o aplicativo e o servidor. Para tal comunicação é utilizado protocolo REST, atuando entre o AngularJS no aplicativo cliente e o Django Rest Framework no servidor

Sendo assim, a comunicação dentro do workbook é feita unicamente em formato JSON, sobre o protocolo REST. Quando é feito uma requisição, seja uma listagem ou detalhamento de algum prestador, o servidor envia apenas os dados requisitados, sem se preocupar com a apresentação de tais dados. O aplicativo então recebe esses dados e os incorpora ao seu HTML e então exibe ao usuário. Importante notar, ainda na figura 7 que a camada Template do MTV foi omitida. Isso se dá pelo fato de que, nessa arquitetura, o aplicativo móvel é responsável pela camada de apresentação.

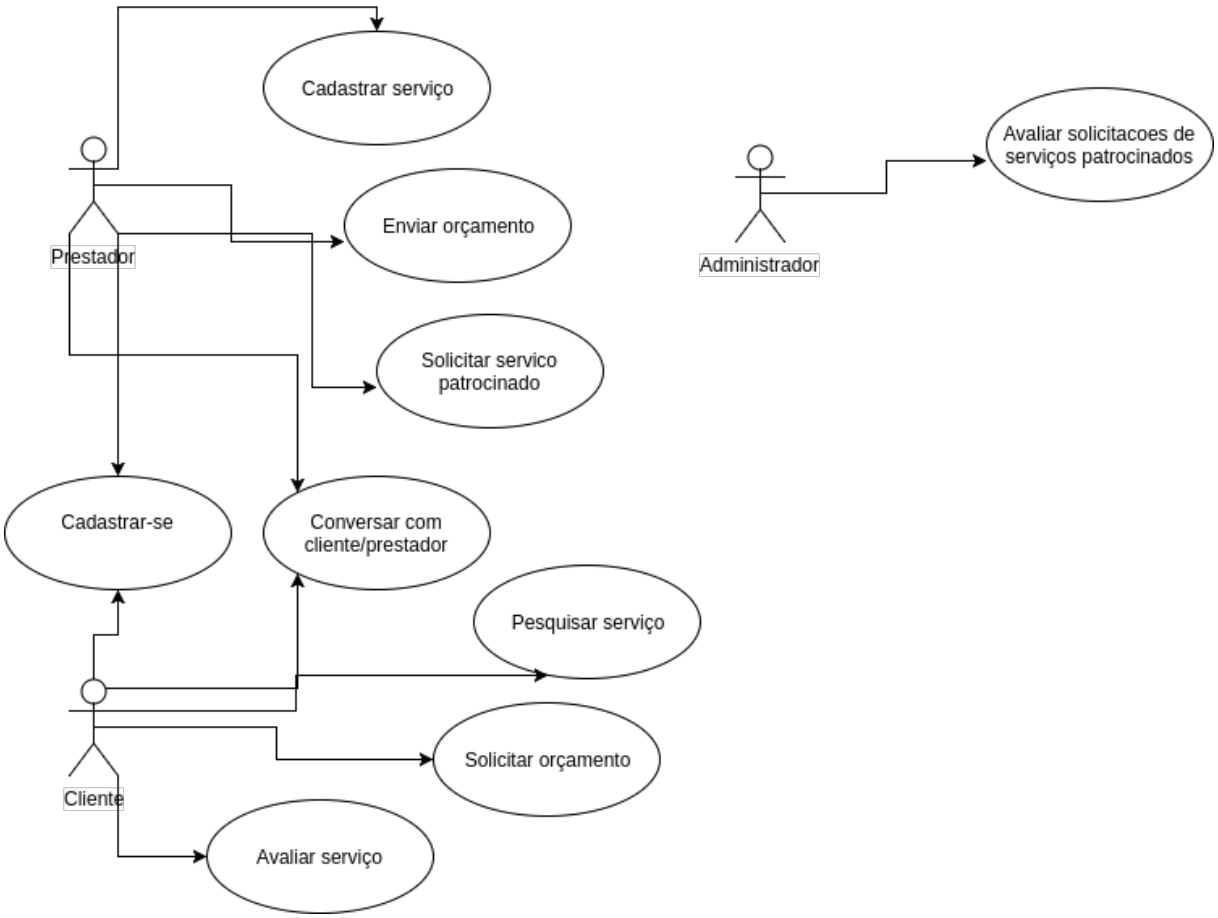
4.2 Diagramas e modelagem

Parte fundamental do desenvolvimento de softwares, diagramas UML (Unified Modeling Language) são representações do sistema à ser desenvolvidos e servem como guias para os desenvolvedores. A seguir serão mostrados os diagramas UML utilizados durante o desenvolvimento do Workbook.

4.2.1 Diagrama de casos de uso

A figura 8 mostra o diagrama de casos de uso do Workbook. Nele existem três atores: usuário, prestador e Administrador.

Figura 8 – Diagrama de casos de uso



Fonte: Elaborado pelo autor

O caso de uso *cadastrar-se* é comum aos dois atores, consistindo em preencher um formulário simples que será salvo no banco.

Através do caso de uso *cadastrar serviço*, o usuário poderá inserir os serviços que realiza. É necessário apenas preencher os dados do serviço. Usuários que não são prestadores podem se tornar prestadores e ter acesso à esse caso de uso. Para isso, basta apenas informar as informações solicitadas antes de cadastrar o primeiro serviço

O caso de uso *pesquisar serviço* é utilizado pelo usuário para encontrar o serviço que deseja contratar. O usuário poderá fazer a pesquisa pelo nome do prestador ou pelo serviço desejado.

É através do caso de uso *solicitar orçamento* que o usuário tem o primeiro contato direto com o prestador. Após a pesquisa, ele seleciona o serviço desejado e vai visualiza informações do prestador. Após isso, basta selecionar os serviços desejados e solicitar o orçamento.

Após a solicitação do orçamento, o prestador e o usuário tem a possibilidade de enviar mensagens entre si, através do caso de uso *enviar mensagem*. Há lista de todas as solicitações relacionadas àquele usuário. Ao abrir a solicitação, o usuário ou prestador tem acesso ao chat do sistema.

Em enviar orçamento o prestador define o valor à ser cobrado pelos serviços que realizará. O prestador tem uma listagem das solicitações de orçamento e dos orçamentos já enviados. O prestador deverá selecionar a solicitação, preencher o valor do serviço, marcar quais serviços serão atendidos e então enviar os dados.

Após a realização do serviço, o usuário o avalia atribuindo uma nota ao prestador e fazendo um comentário sobre o mesmo. Essa avaliação poderá ser vista por outros usuários ao acessarem o perfil daquele prestador.

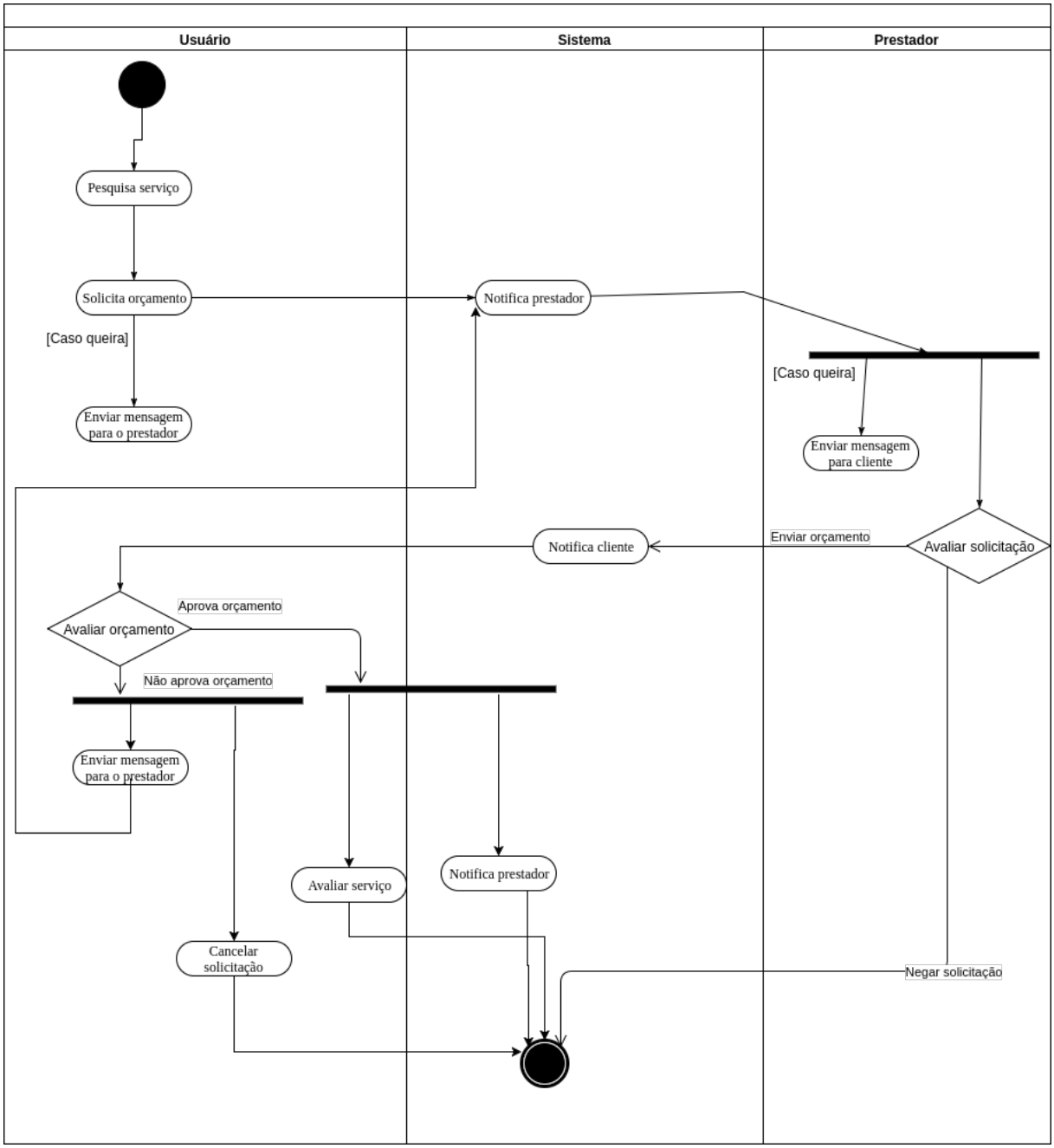
O prestador poderá solicitar que seu serviço apareça em destaque no aplicativo, a partir do caso de uso *Solicitar serviço patrocinado*. Serviço patrocinados aparecem primeiro nas pesquisas e aparecem na tela inicial do aplicativo.

O administrador tem a função de avaliar solicitações de serviços patrocinados, confirmando ou negando uma requisição de anúncio patrocinado

4.2.2 Diagrama de atividades

O diagrama de atividades representado na figura 9 mostra o fluxo de execução de uma solicitação de orçamento no workbook. O fluxo se inicia na pesquisa pelo serviço, após isso o usuário, ao escolher o prestador, solicita o orçamento e pode ou não enviar uma mensagem para o prestador. O Prestador recebe a solicitação e pode também enviar uma mensagem ao cliente. O fluxo segue com o envio do orçamento ao cliente ou a negação do mesmo. O cliente então aprova ou rejeita o orçamento, caso recebido. Em caso de orçamento não aprovado, o cliente e o prestador podem voltar à se comunicar para buscar um novo orçamento. Após o orçamento aprovado, o usuário avalia o serviço. Caso não haja aprovação do orçamento, o usuário cancela a solicitação e o fluxo se encerra.

Figura 9 – Diagrama de atividades



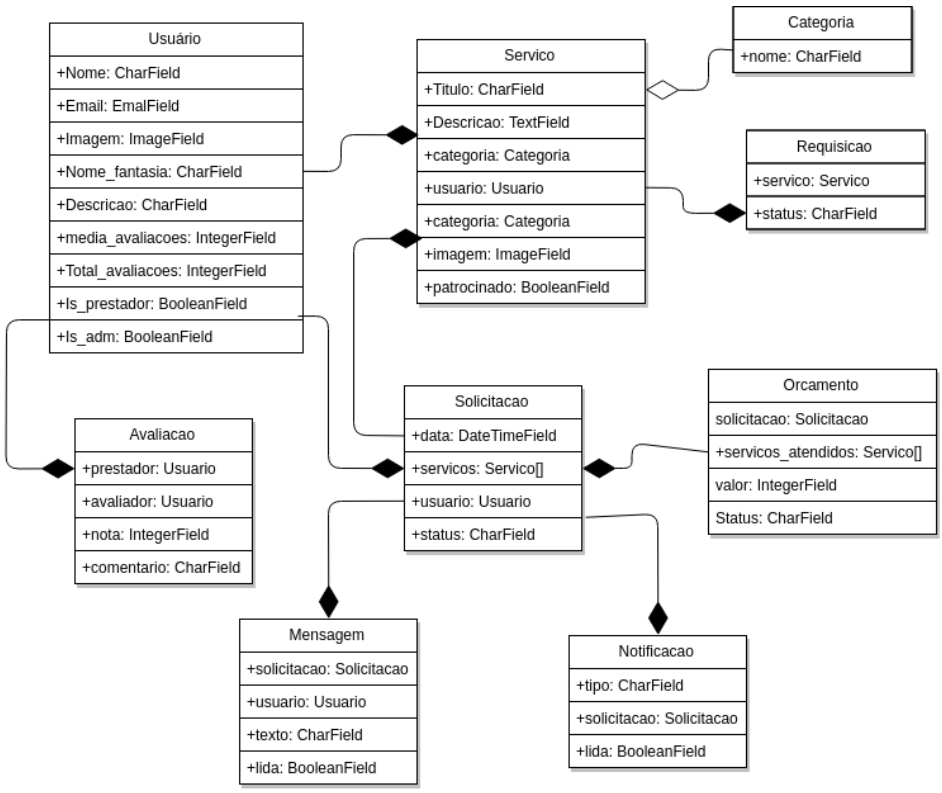
Fonte: Elaborado pelo autor

4.2.3 Diagrama de classes

O Workbook foi desenvolvido sob o conceito de ORM, ou seja, cada classe do domínio representa uma tabela no banco de dados. Por esse motivo, não se faz necessário um diagrama entidade-relacionamento.

A figura 10 representa o diagrama de classes do workbook. Como se pode ver, a separação entre usuário e prestador se dá através do atributo *is_prestador* da classe usuário. Dessa forma, todo prestador pode utilizar as funcionalidades do usuário, sem

Figura 10 – Diagrama de classes



Fonte: Elaborado pelo autor

necessidade de possuir um perfil de usuário. Assim, torna-se simples a mudança de perfil de um usuário comum para prestador.

Notificação e mensagem estão ligadas à solicitação. Dessa forma, o chat do sistema só está disponível após uma solicitação de serviços e todas as alterações ocorridas no status ou mensagens relacionadas àquela solicitação geram uma notificação para usuário ou prestador.

A classe chave do Workbook é solicitação. Ela contem um usuário, um prestador e um array de serviços que armazenará os serviços que o usuário deseja daquele prestador. A classe orçamento é ligada à solicitação e serve como resposta a solicitação. Ela possui o valor do serviço e um array que indica quais serviços fazem parte daquele orçamento.

A classe serviço possui informações do serviço prestado e um relacionamento com a classe usuário, que identificará qual usuário é o prestador de cada serviço. Além disso, possui uma categoria. A classe requisição é utilizada quando um prestador deseja patrocinar seu serviço.

Por fim, a classe avaliação é ligada ao usuário e definem os valores dos campos "total_avaliacoes"e "media_avaliacoes". Os comentários e notas de cada avaliação são exibidos na tela de cada prestador

5 Apresentação do sistema

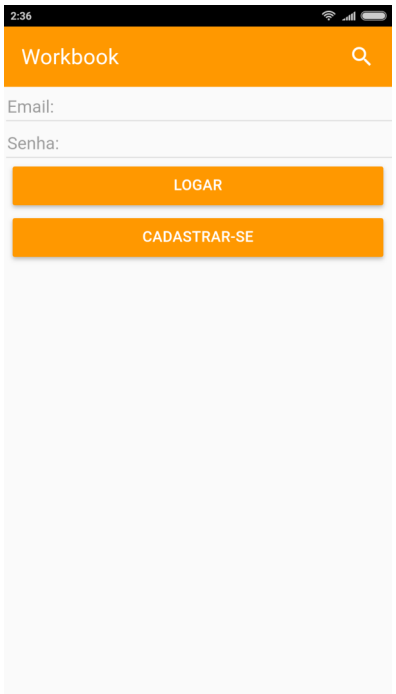
Neste capítulo será apresentado o aplicativo móvel workbook.

A figura 11a mostra a tela de login. Além da possibilidade de login, há um link para a tela de cadastro, mostrada na figura 11b, onde novos usuários se cadastram no sistema. Para o cadastro, é necessário apenas e-mail, nome senha e imagem para o perfil. Após o cadastro, o usuário poderá se tornar prestador informando o campos nome fantasia e a descrição do seu trabalho, como pode ser visto na figura 12

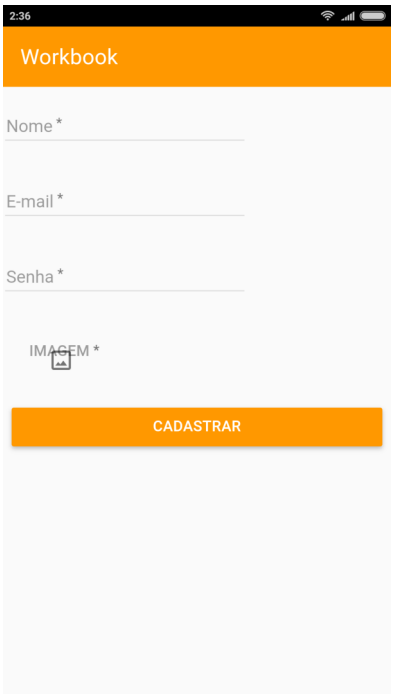
A figura 13 mostra a tela inicial do workbook. Além do ícone do menu e do campo de pesquisa, são exibidos anúncios patrocinados.

Figura 11 – Telas de login e cadastro

(a) Tela de Login

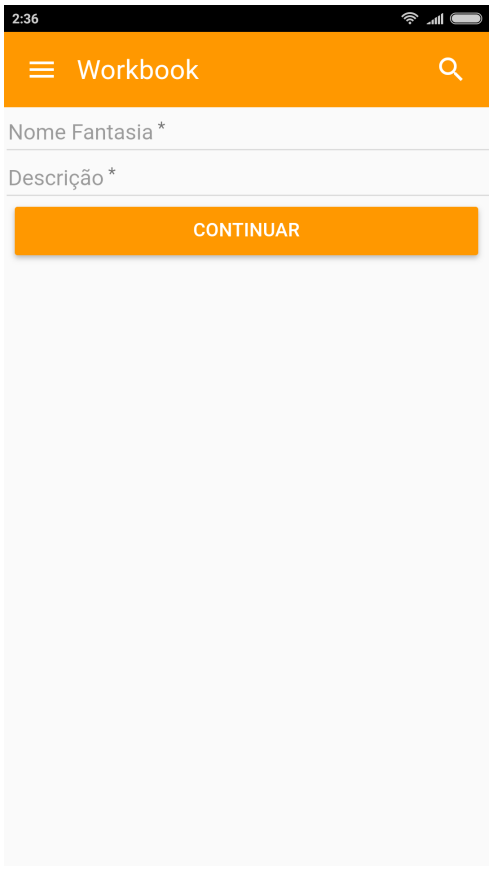


(b) Cadastrar-se



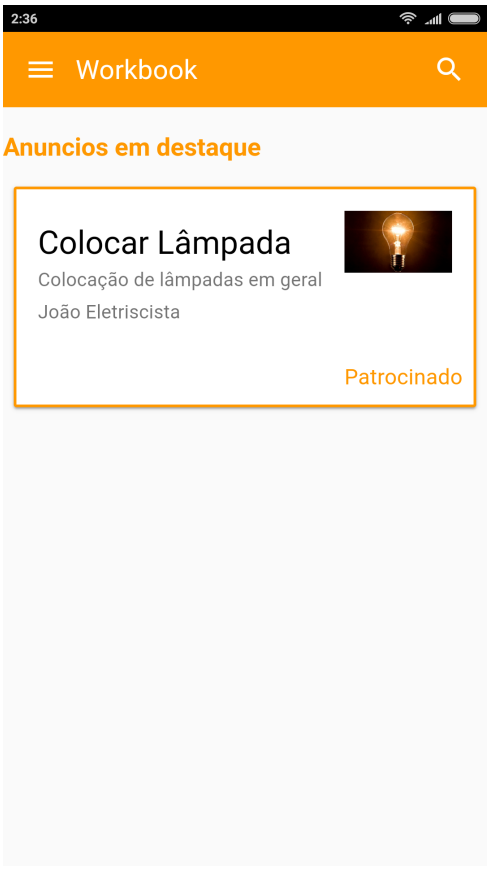
Fonte: Autoria própria

Figura 12 – Formulário para prestadores



Fonte: Autoria própria

Figura 13 – Tela inicial



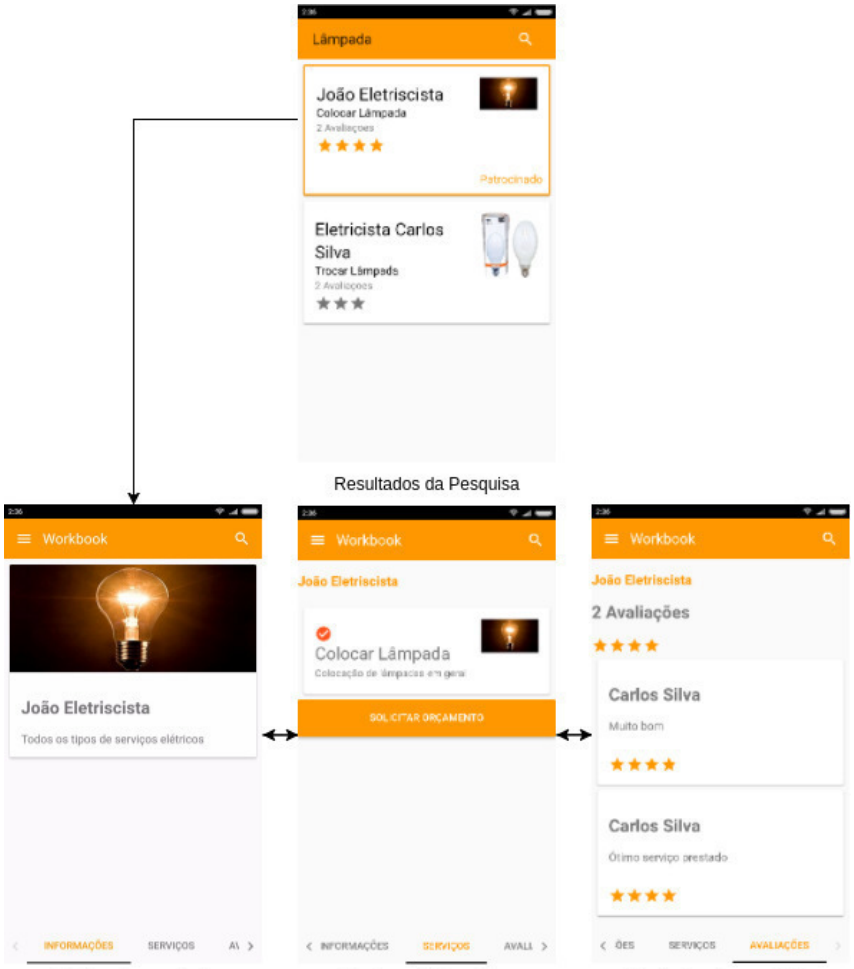
Fonte: Autoria própria

A figura 14 mostra fluxo de pesquisa e solicitação de orçamento. Os resultados da pesquisa são ordenados da seguinte forma: Serviços patrocinados, ordenados por média de avaliações do prestador seguidos pelos demais serviços, também ordenados por avaliações. Ao selecionar um prestador na lista de resultados da pesquisa, o usuário é levado á página do perfil do prestador. Nesta tela dividida em abas, é possível ver as informações gerais, avaliações e solicitar um orçamento ao prestador.

A figura 15 mostra o fluxo de envio de orçamento. O prestador tem acesso aos orçamentos e às solicitações não respondidas. Ao selecionar uma solicitação, o prestador é direcionado para uma página onde ele pode selecionar os serviços à serem atendidos, definir o valor e enviar o orçamento; acessar um chat com o cliente ou negar a solicitação.

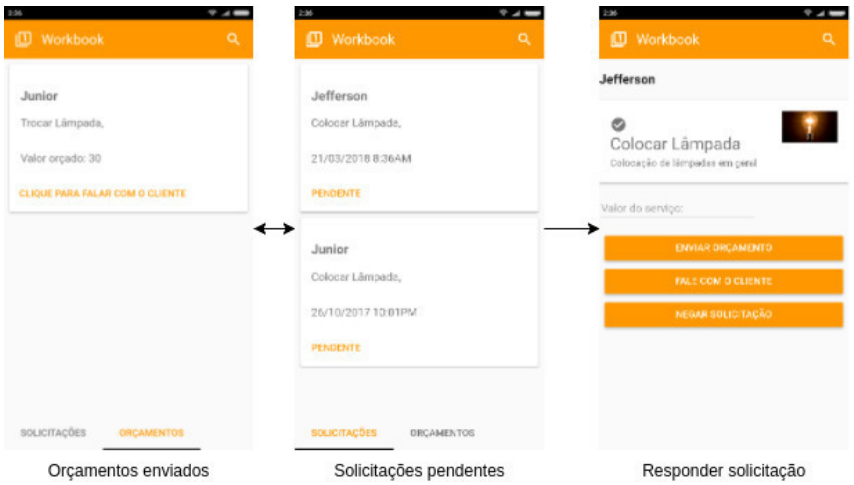
A figura 16 mostra o fluxo de aprovação de um orçamento e a avaliação do prestador. Ao selecionar um orçamento recebido, o usuário é direcionado para uma tela onde ele pode ver os detalhes do orçamento e aprovar o orçamento. Caso não esteja de acordo com o valor orçado, ele pode entrar em contato com o prestador para solicitar um novo orçamento. Após a aprovação do orçamento, o usuário pode avaliar o prestador. A avaliação é composta por uma nota que varia de 1 até 5 estrelas e um comentário.

Figura 14 – Fluxo de solicitação de orçamento



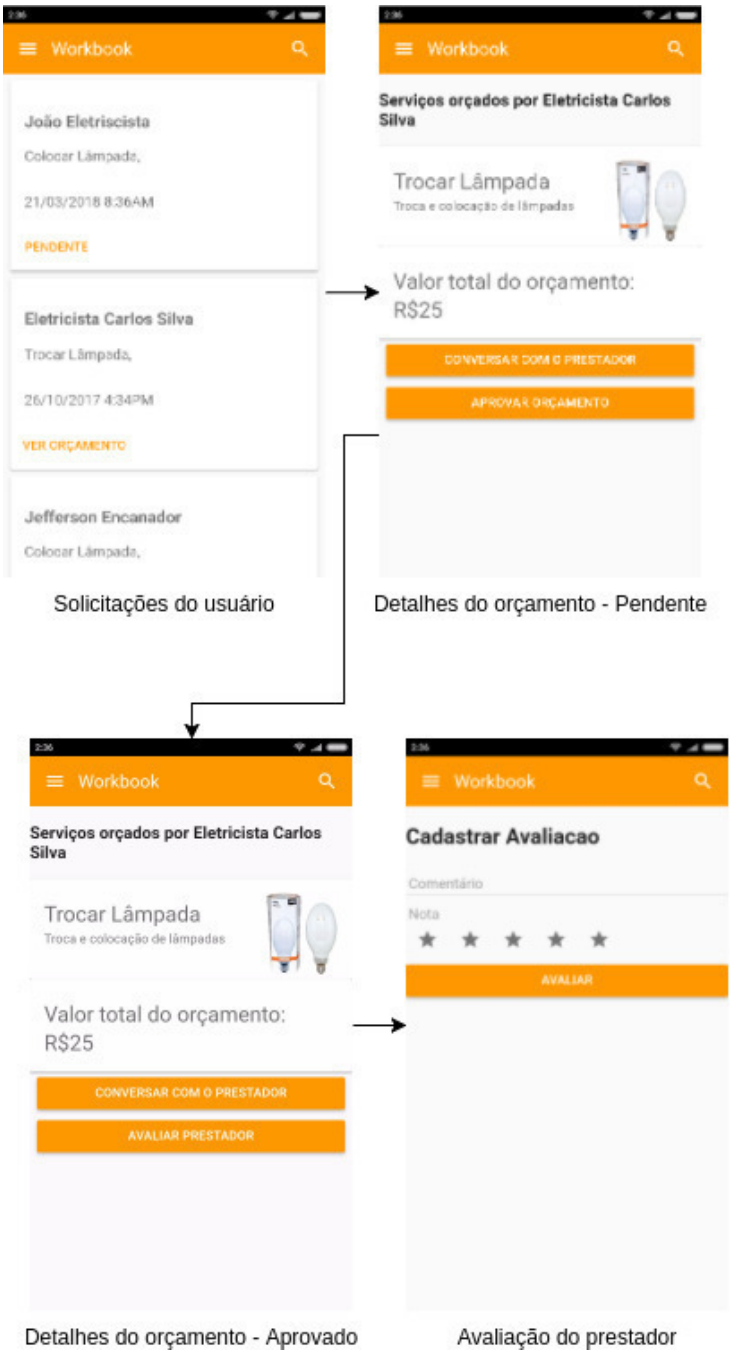
Fonte: Autoria própria

Figura 15 – Fluxo de envio de orçamento



Fonte: Autoria própria

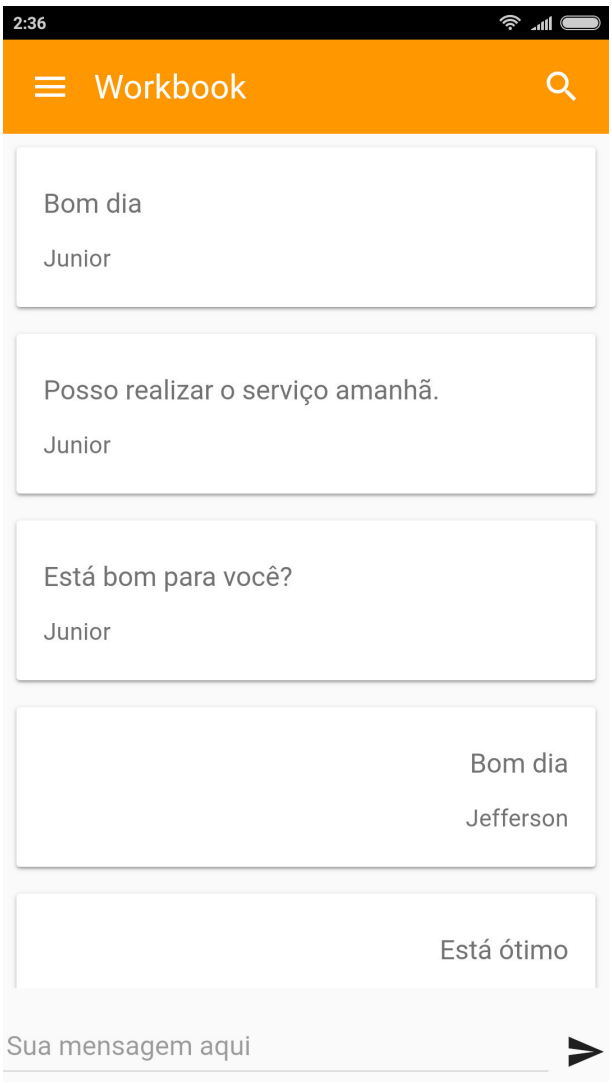
Figura 16 – Fluxo de aprovação de orçamento



Fonte: Autoria própria

A figura 17 mostra a tela de troca de mensagens entre prestador e usuário.

Figura 17 – Chat



Fonte: Autoria própria

6 Conclusão

O presente trabalho apresentou as etapas do desenvolvimento de uma solução para as dificuldades encontradas na prestação de serviços, capaz de prover a interação desejada entre cliente e prestador, além de prover confiabilidade na contratação de prestadores. Foram utilizados durante o desenvolvimento tecnologias web, frameworks de comunicação e segurança, e tecnologias para desenvolvimento híbrido. à esta solução, foi dado o nome de workbook

O workbook é um aplicativo mobile híbrique que oferece aos seus usuários a possibilidade de solicitar orçamentos, troca de mensagens com prestadores de serviço e, ao contratar e ter o serviço atendido, avaliar o prestador dando uma nota e deixando um comentário.

6.1 Trabalhos futuros

Todo software deve estar em constante aperfeiçoamento. Para o workbook, algumas funcionalidades devem ser adicionadas futuramente para melhorar sua utilização, tais como:

- Inclusão de pagamentos pelo aplicativo, mediante confirmação de realização do serviço por ambas as partes
- Geração de gráficos de faturamento do prestador
- Inclusão de prestadores favoritos para o cliente
- Inclusão de estatísticas de uso na área do administrador, tais como prestadores mais acessados afim de melhorar o algoritmo de busca
- Inclusão de avaliação de clientes
- Interação com redes sociais, onde os comentários e avaliações dos amigos apareçam em destaque

Referências

CODEPROJECT. *Code Project*. 2018. Disponível em: <https://www.codeproject.com/Articles/552846/Why-s-How-s-of-Asp-Net-MVC-Part>. Acesso em: 13/02/2018. Citado na página 18.

DATASEBRE. *DataSebrae*. 2016. Disponível em: <http://datasebrae.com.br/pib/>. Acesso em: 23/03/2018. Citado na página 9.

HOLOVATY, A.; MOSS, J. K. *The definitive guide to django: Web development done right*. 2. ed. [S.l.]: Apress, 2009. ISBN 9781590597257. Citado 2 vezes nas páginas 20 e 21.

LOPES, S. *Aplicações mobile híbridas com Cordova e PhoneGap*. Casa do Código, 2016. ISBN 9788555191572. Disponível em: <https://books.google.com.br/books?id=uSCjCwAAQBAJ>. Citado na página 15.

MONTEIRO, J. *Google Android: Crie aplicações para celulares e tablets*. Casa do Código, 2014. ISBN 9788566250916. Disponível em: <https://books.google.com.br/books?id=34-CCwAAQBAJ>. Citado na página 15.

PARTNERS, O. *Using OAuth 2.0 With Google APIs*. 2016. Disponível em: <https://objectpartners.com/2016/01/21/using-oauth-2-0-with-google-apis/>. Acesso em: 13/02/2018. Citado na página 23.

PEREIRA, M. *AngularJS: Uma abordagem prática e objetiva*. Novatec Editora, 2014. ISBN 9788575224113. Disponível em: <https://books.google.com.br/books?id=MUelBQAAQBAJ>. Citado 3 vezes nas páginas 17, 18 e 19.

PYPL. *PYPL PopularitY of Programming Language*. 2018. Disponível em: <https://pypl.github.io/PYPL.html>. Acesso em: 13/02/2018. Citado na página 20.

SAUDATE, A. *REST: Construa API's inteligentes de maneira simples*. Casa do Código, 2014. ISBN 9788566250985. Disponível em: <https://books.google.com.br/books?id=uo-CCwAAQBAJ>. Citado na página 19.

SHELLY, G.; ROSENBLATT, H. *Systems Analysis and Design*. Cengage Learning, 2011. (SAM 2010 Compatible Products Series). ISBN 9780538481618. Disponível em: <https://books.google.com.br/books?id=XiJTWMRPZi4C>. Citado na página 14.

SOCIOMANTIC. *Infográfico aponta dados do m-commerce no Brasil*. 2016. Disponível em: <http://adnews.com.br/internet/infografico-aponta-dados-do-m-commerce-no-brasil.html>. Acesso em: 23/03/2018. Citado na página 9.

TOPTOTAL. *Apache Cordova Tutorial: Developing Mobile Applications with Cordova*. 2015. Disponível em: <https://www.toptal.com/mobile/developing-mobile-applications-with-apache-cordova>. Acesso em: 13/02/2018. Citado na página 16.