Sistema *Online* de Solicitações de Orçamentos

Evandro B. Sangiovo[1], **Alexandre O. Zamberlan**[1], **Ricardo F. da Silva**[1]

[1]Centro Universitário Fransciscano - UNIFRA
Rua dos Andradas, 1614, Santa Maria – RS

esangiovo@gmail.com, {alexz, ricardo.frohlich}@unifra.br

Abstract. *This work is inserted in the Service Delivery area and is based on concepts of Management Information Systems and Decision Support Systems. We aim to design, implement and evaluate a computerized solution that manages requests and receipts of budgets to service providers. The system would approach clients and service providers, in addition to streamlining the budgets processes. The customer would receive multiple budgets from different providers with quality assessments from other customers. The provider has the possibility to access a greater number of clients and to be recognized by the quality of the previously executed services. The research methodology of this investigation is exploratory, with bibliographic review and case study application.*

Resumo. *Neste trabalho, inserido na área de Prestação de Serviços, tendo como alicerce conceitos de Sistemas de Informação Gerencial e de Sistemas de Apoio à Decisão, objetiva-se projetar, implementar e avaliar uma solução informatizada que gerencie solicitações e recebimentos de orçamentos a prestadores de serviços. O sistema aproximaria clientes e prestadores de serviço, além de agilizar os processos de orçamentos. O cliente receberia vários orçamentos de diferentes prestadores com as avaliações de qualidade de outros clientes. Já o prestador tem a possibilidade de acessar um número maior de clientes e ser reconhecido pela qualidade dos serviços executados anteriormente. A metodologia de pesquisa é exploratória, com revisão bibliográfica e aplicação de estudo de caso.*

1. Introdução

Na medida em que correria do dia a-dia aumenta, a Tecnologia da Informação (TI) apresenta-se como uma protagonista fundamental e indispensável, pois é por meio de suas metodologias, ferramentas e conceitos dinâmicos que se busca facilitar (ou senão amenizar) rotinas e processos de pessoas e organizações.

Conforme dados do Centro Regional para o Desenvolvimento da Sociedade da Informação [CETIC 2017], no ano de 2017 mais de 51% da população brasileira encontra-se conectada (*online*) por meio da Internet. Por essa razão, as ferramentas que são utilizadas nesse meio também devem evoluir e oferecer novas alternativas para favorecer novas perspectivas de uso, criando novos paradigmas de interação.

A área de Sistemas de Informação está em constante atenção, a fim de identificar mudanças, tendências e propor recursos para a adaptação.

No contexto de negócios via Internet (interações *online*), consumidores e prestadoras de serviço cada vez mais se utilizam da TI para agilizar, facilitar, reduzir custos, en-

tre outros. É possível visualizar inúmeros exemplos, como os aplicativos Uber[1], 99Taxi[2], Garupa[3], etc. Contudo, toda essa interação deve garantir a segurança dos usuários.

Na área da Saúde há prestadoras de serviços virtuais [Harrop 2001], conhecidas como *Virtual Healthcare*, em que diferentes profissionais da saúde estão cadastrados e à disposição dos usuários (médicos, enfermeiros, fisioterapeutas, psicólogos, etc). Esses usuários podem acessar um portal virtual, descrever seu problema ou demanda, identificar o possível profissional (ou profissionais) que o ajudarão, receber orçamentos, prazos e, finalmente, solicitar o serviço ou atendimento (presencial ou até mesmo virtual).

Portanto, este trabalho tem como objetivo geral projetar, implementar e avaliar uma solução informatizada que gerencie de forma descentralizada informações de prestadores de serviço, criando uma rede de interação *online* entre consumidores e prestadores.

Um propósito do trabalho é agilizar a solicitação e recebimento de orçamentos. Usuários poderão escolher um profissional que mais lhe convém, via análise de perfil, preço, prazos e opinião de outros usuários (visualizar as avaliações de consumidores e as de fotos de serviços já executados).

Para isso, assume-se como objetivos específicos:

- definir e descrever o estudo de caso;
- pesquisar e escrever sobre Sistemas de Informação, Software como Serviço, Arquiteturas para Software como Serviço e Trabalhos Relacionados;
- identificar e testar tecnologias para projetar e implementar o estudo de caso;
- aplicar a metodologia FDD;
- desenhar e descrever a arquitetura do sistema;
- implementar parte do sistema no Firebase;
- implementar parte do sistema para Android;
- realizar testes e avaliar o sistema;

1.1. Justificativa

Acredita-se que a relevância do projeto se dá porque para solicitar orçamentos a prestadores de serviços, é necessário recorrer a meios ditos tradicionais, como *sites*, classificados em jornais, propagandas ou a conhecida ”indicação de amigo”. Além disso, após a escolha, deve-se entrar em contato com o prestador para efetuar a solicitação do orçamento, avaliar os orçamentos, investigar a qualidade do prestador, escolher o orçamento e, finalmente, realizar o contrato formal.

Conforme a 27º Pesquisa Anual, em 2016, realizada pela Fundação Getúlio Vargas de São Paulo (FGV-SP) [Meirelles 2016], o Brasil alcançou um total de 168 milhões de *smartphones* em uso. E isso significa que mais de 81% da população brasileira com acesso a dispositivo móvel. Dessa forma, com base nessa pesquisa, fica evidente que aplicativos, que facilitem o cotidiano, são cada vez mais necessários.

[1]https://www.uber.com
[2]https://99app.com
[3]http://garupa.co

2. Referencial Teórico

Esta seção apresenta o contexto de Sistemas de Informação. Em seguida, é abordado o conceito de "*Backend*" para que se possa entender o porquê de usar um serviço terceirizado, denominado *Backend as a Service* (BaaS).

2.1. Sistemas de Informação

Sistema de Informação (SI) é a coleção estruturada de elementos úteis à coleta, processamento, armazenamento e entrega de informações. Essa coleção deve garantir suporte à tomada de decisões e no controle da organização pelos profissionais de Tecnologia da Informação (TI) [Laudon and Laudon 2014].

Há diferentes classificações para SI, de acordo com a sua utilização e o tipo de resultado no processo de tomada de decisões. Entretanto, pode-se afirmar que os tipos usuais são: Sistemas de Processamento de Transações (SPTs); Sistemas de Informação Gerenciais (SIG); Sistemas de Apoio à Decisão (SAD); Sistemas de Apoio ao Executivo (SAE); Sistemas Integrados (*Enterprise Resource Planning* – ERP); Sistemas de Gestão da Cadeia de Suprimentos (*Supply Chain Management* – SCM); Sistemas de Gerenciamento do Relacionamento com o Cliente (*Customer Relationship Management* – CRM) [Laudon and Laudon 2014].

2.2. *Cloud Computing*

Para [Veras 2012], o conceito de *Cloud computing* (computação em nuvem) refere-se a todo o processamento computacional e armazenamento de dados que são executados fora da infraestrutura local, ou seja, através de uma conexão com a Internet é possível utilizar serviços computacionais, armazenar informações de modo seguro, sem a necessidade da utilização de processamento e armazenamento das informações na estrutura física local.

2.3. *Software as a service*

Software as a Service (SaaS) é o modelo que disponibiliza sistemas, via Internet, com o intuito específico para usuários consumidores. Ou seja, são softwares disponibilizados em *Cloud Computing*.

Os projetos que utilizam esse modelo focam apenas com configurações da aplicação. Dessa forma, o cliente da aplicação ou serviço (usuário consumidor) paga somente pelo uso, manutenção e suporte técnico prestados. Serviços como Google Docs[4], Gmail[5] e Dropbox[6] são serviços dentro desse modelo - SaaS.

Uma vez que um sistema é pensado para Internet (com infraestrutura própria ou terceirizada), há outro conceito fundamental a discutir: *Backend as a Service* (BaaS).

2.4. *Backend*

No desenvolvimento de software, conforme [VisualWebz.com 2017, Rymsha 2017], *Backend* refere-se a todo o código que é executado no lado do servidor, tais como validação, regras de negócio, persistência, autenticação, armazenamento, envio de mensagens,

[4]https://docs.google.com
[5]https://www.gmail.com
[6]https://www.dropbox.com

notificações, entre outras, que possam ou não se comunicarem com o *Frontend*. Pode-se citar algumas tecnologias que são utilizadas para a execução dessas tarefas: .NET, JAVA, Python, PHP, RUBY. A Figura 1 ilustra as tecnologias e as pessoas envolvidas em cada camada.

Figura 1. *Frontend versus Backend* [VisualWebz.com 2017, Rymsha 2017]

O desenvolvimento do *Backend*, geralmente, é a parte mais complexa e custosa de um projeto. E é nesse ponto que se deve avaliar os benefícios da utilização de serviços de BaaS.

Backend as a Service é um serviço que implementa as funcionalidades básicas ou complexas de um *Backend* comum mas com toda a sua infraestrutura disponibilizada na Nuvem. É possível citar alguns serviços que utilizam esse conceito: Google Firebase [7], Parse Server [8] e Kinvey. [9]

Esses serviços podem ser utilizados por desenvolvedores por meio da utilização de Interfaces de Programa de Aplicações (*Application Program Interface* - API) e Kits de Desenvolvimento de Software (*Software Development kit* - SDK). Uma API é um conjunto de rotinas, protocolos e ferramentas para a construção de aplicativos de software. Basicamente, API especifica como os componentes de software devem interagir. Além disso, as APIs são usadas na programação dos componentes da Interface Gráfica do Usuário (GUI). Um SDK é um conjunto de ferramentas de desenvolvimento de software que permitem a criação de aplicativos para um determinado pacote de software, estrutura de software, plataforma de hardware, sistema informatizado, etc. [Borges 2014].

2.5. Trabalhos relacionados

Software como serviço é uma realidade nos mais diferentes contextos, saúde, lazer, transporte, entre outros muitos.

Em 2001, já existia uma preocupação com a prestação de serviços na área da saúde para a Internet [Harrop 2001]. Essa preocupação motivou um estudo [Harrop 2001] que

[7]https://firebase.google.com
[8]https://www.back4app.com
[9]http://kinvey.com

identificou que organizações provedores de serviços na saúde não tinham: i) definição de prestação de serviços ”virtual”em relação aos produtos, serviços e processos oferecidos pelas *ponto.coms*; ii) um modelo para integrar a entrega de cuidados de saúde reais e virtuais. Dessa forma, o artigo propôs um modelo de entrega de cuidados de saúde virtual e assíncrono e terceirizado. Esse modelo teve como intuito romper barreiras existentes naquele ano: resistência às redes virtuais integradas de distribuição de terceirização; confusão sobre requisitos de infraestrutura virtual para portais na saúde e serviços, e o impacto de redes de prestação de serviços integradas e terceirização sobre normas existentes e práticas geradoras de receita.

O trabalho realizado por [dos Santos et al. 2015], propôs e implantou uma arquitetura de *Cloud Computing* baseada nas características propostas pelo NIST (*National Institute of Standards and Technology*), por meio de software livre, que garantiu alta disponibilidade e escalabilidade horizontal necessários para a disponibilização de um sistema *online* para gestão de consultórios e clínicas através do modelo SaaS. Nesse trabalho, foram utilizadas e integradas diversas tecnologias, destacando Linguagem de programação Python com o *framework* Django, Node.js (interpretador de código Javascript no lado do servidor), Sistema gerenciador de banco de dados Postgresql e *framework* de *frontend* Bootstrap. Esses dois trabalhos fornecem alicerce para esta pesquisa. O trabalho de [Harrop 2001] discute e aponta conceitos e processos da terceirização de serviços de forma virtual e a sua relação entre clientes e prestadores de serviços. Além disso, apresenta um modelo bastante interessante de contrato e cobrança de valores. Já no trabalho de [dos Santos et al. 2015], a contribuição se dá pela revisão bibliográfica e por todas as tecnologias utilizadas para colocar em operacionalização um SaaS.

3. Materias e Métodos

A metodologia de pesquisa deste estudo é exploratória, com revisão bibliográfica e aplicação de estudo de caso. Assume-se que estudo de caso é um método qualitativo que serve para responder questões que o investigador não tem controle sobre o fenômeno estudado (sistema *online* de solicitação e atendimento à prestadora de serviços). Esse método é útil quando o fenômeno a ser investigado é amplo e complexo e não pode ser estudado fora do contexto onde ocorre naturalmente. Considerado estudo empírico, ele busca determinar ou testar uma teoria, e tem como fonte de informações as entrevistas. Por meio delas o entrevistado expressa sua opinião sobre determinado assunto [Yin 2015].

3.1. Tecnologias

Está sessão descreve as tecnologias que serão utilizadas para o desenvolvimento do projeto. O sistema será desenvolvido utilizando as seguintes Tecnologias: IDE Android Studio para a escrita de códigos e construção das telas (*Views*);Linguagem de programação Java que será utilizada para a codificação do aplicativo; Plataforma de desenvolvimento do Google chamada Firebase que será utilizada para a abstração do *Backend* da aplicação. E o sistema operacional Android, pois o aplicativo será executado em dispositivos móveis que utilizam este sistema operacional.

1. Android - sistema operacional desenvolvido pela empresa Google, sendo baseado no *kernel* 2.6 do Linux. Tem como responsabilidade gerenciar memória, processos, *threads*, arquivos, redes e *drivers* [Lecheta 2010]. Conforme

[StatCounter 2017], o sistema operacional Android é o líder do mercado mundial com 39,81% em setembro de 2017, quando comparado com outros sistemas operacionais para dispositivos móveis.

2. Java - plataforma de desenvolvimento composta por três pilares: Máquina Virtual Java (JVM), conjunto de API's e a linguagem de programação Java[Silveira et al. 2012]. A linguagem de programação Java é uma das mais utilizadas no mundo. Conforme pesquisa da empresa TIOBE, ela está listada em primeiro lugar[TIOBE 2017].
3. Android Studio - ambiente de desenvolvimento integrado (IDE) utilizado para o desenvolvimento de aplicativos para Android. A IDE permite a integração com vários softwares de versionamento de códigos tais como: Git, GitHub, CVS, Mercurial, subversion e Google Cloud Source Repositories[Google 2017a]
4. Google Firebase - plataforma de desenvolvimento de aplicativos que abstrai vários processos do *Backend* da aplicação. É possível citar alguns serviços que serão utilizados neste projeto: *Authentication* para autenticar usuários; *Cloud Storage* para armazenamento de arquivos na nuvem; *Realime Database* para o armazenamento e sincronização de informações. Para utilizar os serviços do Firebase, os desenvolvedores devem utilizar as bibliotecas disponibilizadas [Google 2017b].

3.2. Estudo de Caso

Assume-se que a prestação de serviços contempla profissionais como pedreiros, carpinteiros, pintores, eletricistas, domésticas, cuidadores, entre outros. O sistema aproximaria clientes (pessoas físicas e/ou sociais) e prestadores de serviço, além de agilizar os processos de orçamento e de contrato. O cliente receberia vários orçamentos de diferentes prestadores com as avaliações de qualidade de outros clientes. Já o prestador tem a possibilidade de acessar um número maior de clientes e ser reconhecido pela qualidade dos serviços executados anteriormente.

3.2.1. Funcionamento do sistema

O sistema proposto será um aplicativo para solicitação de orçamentos em dispositivos móveis que utilizam o sistema operacional Android. O processo envolve dois atores : consumidores e prestadores de serviços. O consumidor efetua solicitações de orçamentos e o sistema faz a notificação para todos os prestadores que se enquadram na solicitação. Após receber a notificação o prestador poderá avaliar a solicitação e enviar proposta de valores para a execução do serviço. O consumidor após receber a proposta, poderá visualizar o perfil do prestador, visualizar as notas dos serviços já prestados e, por fim, autorizar a proposta recebida. Em seguida, contrato é firmado e a autorização será efetuada na agenda de execução.

3.3. Metodologia

A metodologia escolhida para a construção do sistema, mais especificamente do estudo de caso, é *Feature Driven Development* (FDD). Considerada uma metodologia ágil e customizável para projetos orientados a objetos, com desenvolvimento iterativo e incremental, promovendo a colaboração constante entre desenvolvedores e interessados (participantes), do planejamento até os testes do sistema [Pressman 2010].

FDD é dividida em cinco processos, separada por duas fases: concepção e planejamento; e construção. A primeira fase contém três processos: desenvolver modelo abrangente; construir lista de funcionalidades; e planejar por funcionalidades. Já na segunda fase, há dois processos: desenhar por funcionalidades e construir por funcionalidades [Pressman 2010].

Além disso, o processo de desenvolvimento obedece o padrão arquitetural *Model-View-Controller* (MVC). Esse padrão sugere a divisão do software em três partes interligadas: i) *model* que trata dos dados, lógica e regras do sistema; ii) *view* que fornece a representação da informação, ou seja, a saída do processamento por meio de texto, gráficos ou animações; iii) *controller* que recebe a entrada dos dados do sistema e faz a ligação entre *model* e *view* [Borges 2014].

3.3.1. Desenvolver modelo abrangente

De acordo com a metodologia FDD, este processo marca o início projeto. Esse modelo deve ser criado a partir da análise e do entendimento do escopo do sistema, tendo como resultado os diagramas necessários [Pressman 2010]. Para ilustrar o modelo foram desenvolvido o diagrama de domínio (Figura 2). Pela metodologia FDD ser customizável, é possível adaptar e utilizar diferentes artefatos [Pressman 2010].

Diagrama de domínio: ilustrado com diagrama de classes, as definições de operações, responsabilidades e cardinalidades são dispensadas. Este modelo de domínio é composto por: classes conceituais, relações entre elas e atributos [McLaughlin et al. 2007].

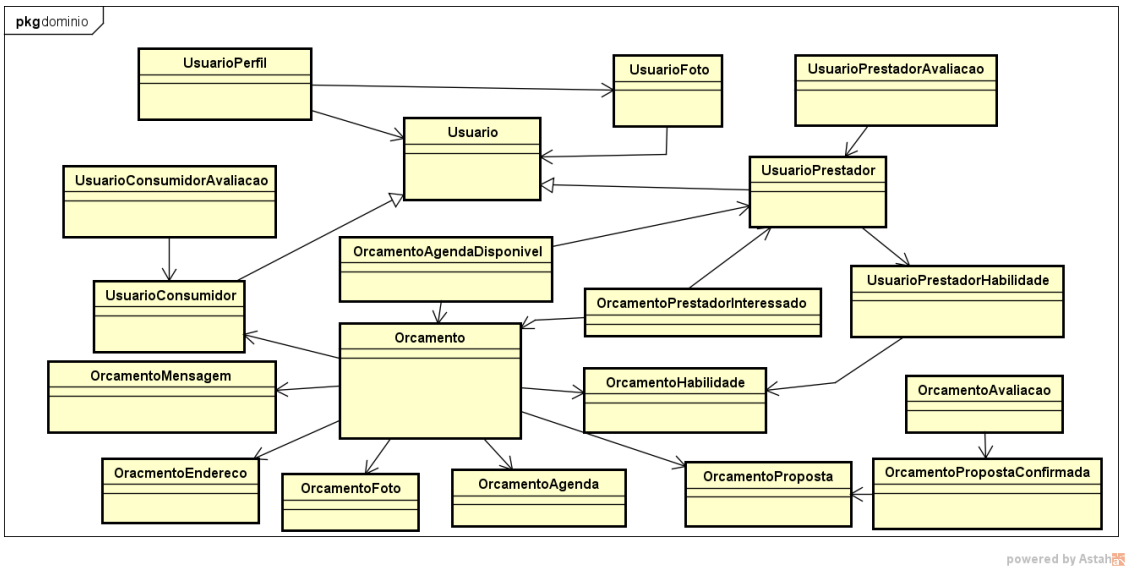

Figura 2. Diagrama de domínio

3.3.2. Construir lista de funcionalidades

Conforme a metodologia FDD, a lista de funcionalidades é extraída a partir da análise do modelo do domínio, que pode ser chamada de *backlog* ou lista de espera [Pressman 2010]. O diagrama de casos de uso também tem um papel fundamental na identificação das funcionalidades do sistema.

Diagrama de caso de uso: a figura 3 ilustra as funcionalidades que o sistema deve implementar. Ele pode ser utilizado para facilitar a identificação dos requisitos [McLaughlin et al. 2007].

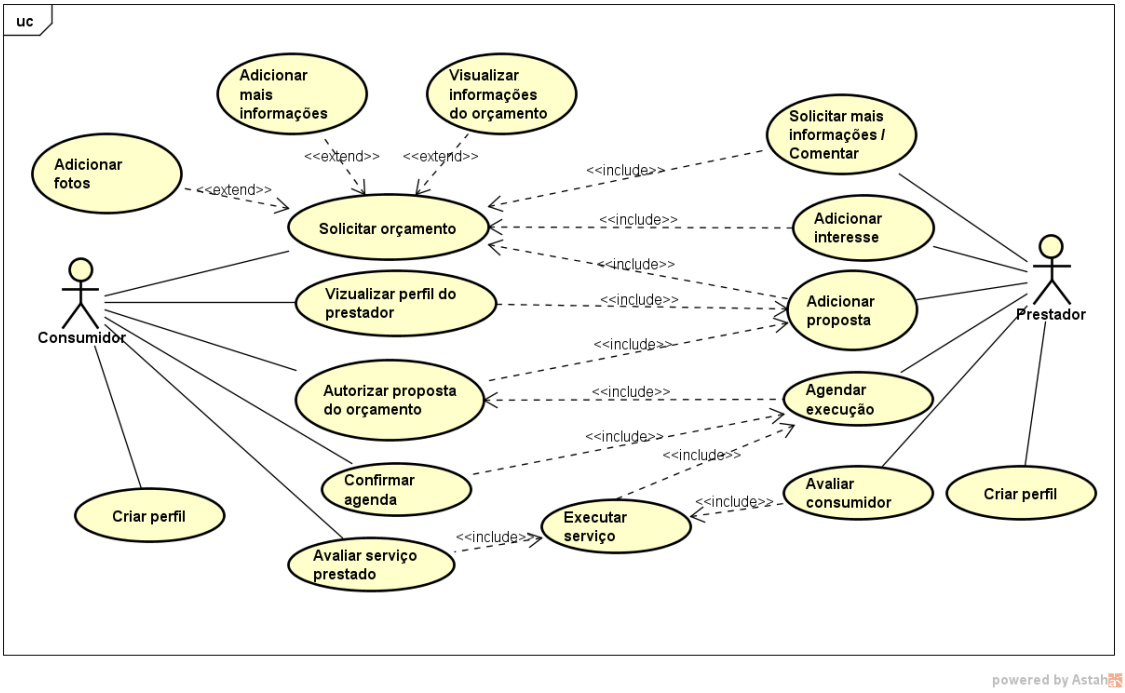

Figura 3. Diagrama de caso de uso

Dentro desse processo, há a fase de levantamento de requisitos, que tem o objetivo de identificar e compreender as necessidades do negócio. Após a compreensão do problema, deve-se documentar e transferir tais necessidades para o software.

Requisitos funcionais: são funções que o sistema deve ter. Esses requisitos são extraídos mediante a análise do problema e refinados com o desenvolvimento dos diagramas [McLaughlin et al. 2007]. Seguem os requisitos funcionais do sistema:

- RF01 - O sistema deverá permitir o cadastro de novos usuários consumidores e prestadores de serviços;
- RF02 - O sistema deverá permitir o preenchimento do perfil dos usuários;
- RF03 - O sistema deverá permitir o cadastro das habilidades para os perfis de prestadores;
- RF04 - O sistema deverá permitir que os consumidores realizem solicitações de orçamentos para um grupo de prestadores de serviços;
- RF05 - O sistema deverá permitir que os consumidores adicionem novas informações aos orçamentos existentes;

- RF06 - O sistema deverá permitir que os prestadores realizem lançamentos de interesse no orçamento;
- RF07 - O sistema deverá permitir que os prestadores realizem lançamentos de propostas;
- RF08 - O sistema deverá permitir que os prestadores realizem lançamentos de mensagens;
- RF09 - O sistema deverá permitir que os prestadores realizem a solicitação de mais informações nos orçamentos;
- RF10 - O sistema deve permitir que os consumidores visualizem os perfis dos prestadores que enviaram propostas;
- RF11 - O sistema deverá permitir que os consumidores aprovem propostas recebidas dos prestadores;
- RF12 - O sistema deverá permitir o agendamento da execução do serviço;
- RF13 - O sistema deverá permitir que os consumidores avaliem os perfis dos prestadores quando os mesmos já prestaram o serviço;
- RF14 - O sistema deverá permitir que os prestadores avaliem os perfis dos consumidores quando os mesmos já foram requisitados para efetuar o serviço;
- RF15 - O sistema deverá permitir que os consumidores avaliem os serviços prestados pelos prestadores.

Requisitos não funcionais: descrevem regras e características de usabilidade, disponibilidade, tecnologias, segurança, desempenho que devem ser seguidas no desenvolvimento do projeto [McLaughlin et al. 2007]. Seguem os requisitos não funcionais do sistema:

- RNF01 - O sistema deverá ser desenvolvido utilizando a linguagem de programação Java;
- RNF02 - O sistema deverá ser desenvolvimento para executar em dispositivos móveis que utilizam o sistema operacional Android;
- RNF03 - O sistema deverá utilizar o *framework* Firebase para abstrair o *backend* da aplicação;
- RNF04 - O sistema deverá utilizar o banco de dados Firebase - Realtime Database;
- RNF05 - O sistema deverá notificar os usuários com o Firebase - Cloud Messaging;
- RNF06 - O sistema deverá funcionar *online* e *offline*;
- RNF07 - O sistema deverá autenticar usuários utilizando o Firebase - Authentication.

3.3.3. Planejar por funcionalidades

Este processo contempla a estimativa de tempo (prazos de entrega) e de dependência para cada funcionalidade e requisitos listados [Silveira et al. 2012]. Dessa forma, a Tabela 3.3.3 ilustra as estimativas de entrega para cada funcionalidade elencadas no diagrama de casos de uso da Figura 3.

Tabela 1. Estimativas de desenvolvimento

Nome	Descrição	Estimativa
RF01	Cadastro básico dos perfis.	5 horas
RF02	Preenchimento das demais informações dos perfis.	3 horas
RF03	Preenchimento das habilidades nos perfis dos prestadores.	4 horas
RF04	Solicitações de orçamentos de serviços.	8 horas
RF05	Adicionar mais informações no orçamentos existentes.	3 horas
RF06	Lançamentos de interesse no orçamento.	2 horas
RF07	Lançamentos de propostas.	2 horas
RF08	Lançamentos de mensagens.	2 horas
RF09	Solicitação de mais informações nos orçamentos.	3 horas
RF10	Visualização dos perfis dos prestadores.	6 horas
RF11	Aprovar proposta recebida.	2 horas
RF12	Agendamento da execução do serviço.	6 horas
RF13	Avaliação de perfil de prestadores.	2 horas
RF14	Avaliação de perfil de consumidores.	2 horas
RF15	Avaliação do serviço prestado.	3 horas
	Total	51 horas

3.3.4. Desenhar por funcionalidades

Este processo também é conhecido por detalhar por funcionalidade. Por FDD ser iterativo e incremental, este processo já abrange a construção, contudo com dependência dos diagramas de atividades e de classe. Neste ponto, o diagrama de classe precisa ser detalhado, com atributos e métodos.

Diagrama de classes: ilustra a estrutura do sistema, expondo o conjunto de classes, atributos, operações e seus relacionamentos. Ele permite uma visualização mais precisa do escopo do software, detalhando as propriedades, operações e relações entre as classes [McLaughlin et al. 2007]. A figura 4 ilustra a estrutura do sistema com suas relações, abaixo serão listadas as classes:

- Usuario: classe responsável pelo gerenciamento dos usuários do sistema, realiza a configuração dos perfis, valida e-mail, adiciona fotos e seleciona o avatar;
- UsuarioConsumidor: estendida da classe ”Usuario”, com a funcionalidade para solicitar novos orçamentos e avaliar usuários do tipo Prestador;
- UsuarioPrestador: estendida da classe ”Usuario”, possui a funcionalidade de avaliar usuários do tipo Consumidor;

- Orcamento: classe principal do sistema, possui as principais funcionalidades, que são: Adicionar uma nova mensagem, adicionar fotos, adicionar proposta de valores, registrar interesse no orçamento solicitado, adicionar agenda disponível para a execução do serviço, selecionar uma agenda disponível, efetuar a avaliação do serviço já executado, aprovar propostas recebidas e adicionar informações de endereço da solicitação.
- Outras: As classes UsuarioPerfil, UsuarioFoto, UsuarioConsumidorAvaliacao, UsuarioPrestadorAvaliacao, UsuarioPrestadorHabilidade, OrcamentoHabilidade, OrcamentoFoto, OrcamentoPrestadorInteressado, OrcamentoAvaliacao, OrcamentoPropostaConfirmada, OrcamentoProposta, OrcamentoEndereco, OrcamentoMensagem, OrcamentoAgenda, OrcamentoAgendaDisponivel são denominadas do tipo *Value Object* e são utilizadas para trafegar, calcular e armazenar dados no fluxo interno do aplicativo[Freeman and Freeman 2009].

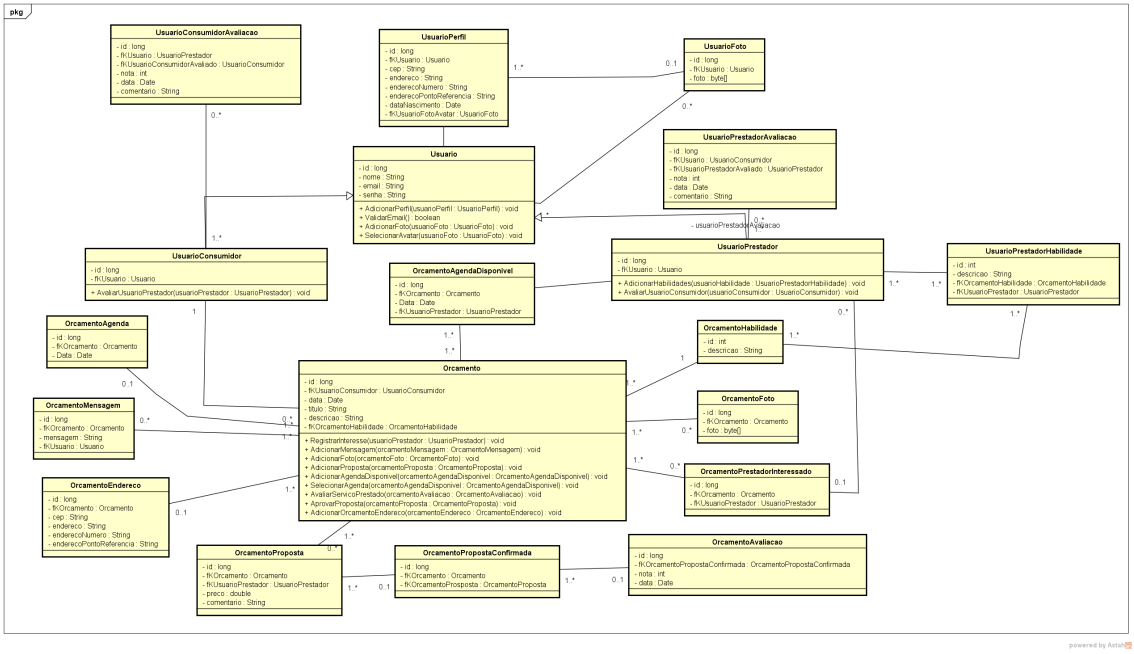

Figura 4. Diagrama de classes

3.3.5. Construir por funcionalidade

Para cada classe do diagrama, são feitos: implementação, testes e inspeções (foco do Trabalho Final de Graduação (II).

Diagrama de atividades: tem por objetivo detalhar o fluxo das atividades do sistema [Silveira et al. 2012] (Figura 5)

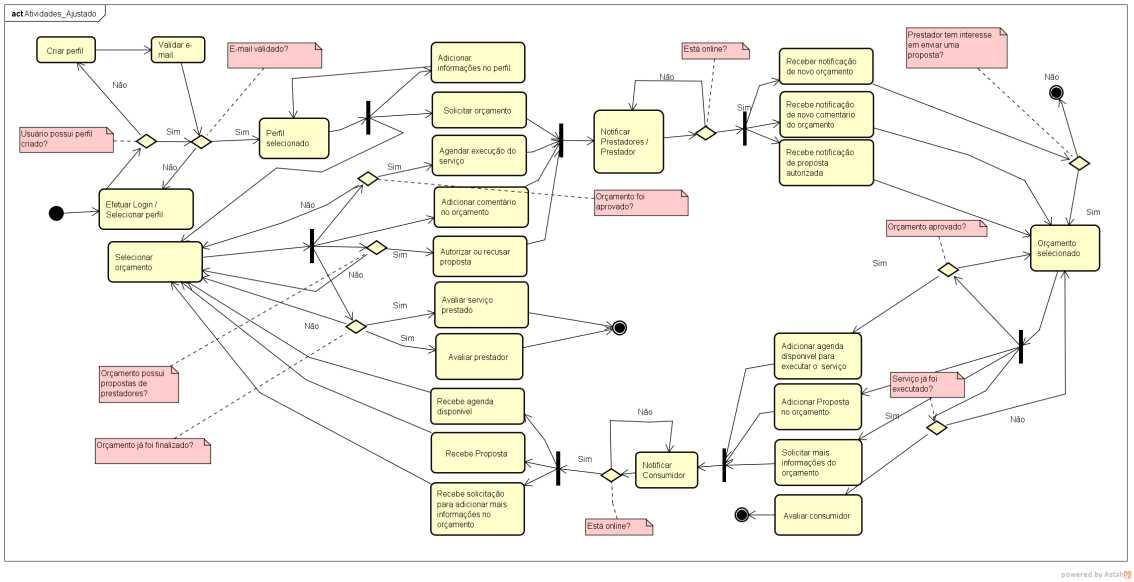

Figura 5. Diagrama de atividades

4. Considerações Finais

O texto apresentou um referencial teórico fundamental para a compreensão da proposta, e detalhou aspectos funcionais e estruturais do sistema, tendo como base a metodologia FDD. Dos conceitos importantes, destacam-se *Backend*, *Backend as a Service*, *Software as a Service*.

Este trabalho tem como proposta o desenvolvimento de um sistema para dispositivos móveis (arquitetura distribuída e descentralizada), com o objetivo de facilitar a relação cliente-prestador. Dessa forma, agilizando as solicitações de orçamentos para prestadores de serviços, e garantindo agilidade, qualidade e segurança aos clientes.

O conjunto de tecnologias escolhido para a construção da proposta tende a reduzir o tempo de desenvolvimento, pois as ferramentas utilizadas são consolidadas e fornecem funcionalidades prontas. Por exemplo, os processos de autenticação (usuários) e de envio de mensagens. Além disso, esse conjunto promove a segurança, tanto para a empresa desenvolvedora, quanto para os usuários do sistema.

Finalmente, pode-se afirmar que as primeiras fases da metodologia FDD estão concluídas, faltando o desenvolvimento das funcionalidades mapeadas e listadas (Seção 3.3.4). O cronograma da Tabela 2 apresenta as atividades planejadas, executadas e a serem realizadas no Trabalho Final de Graduação II (implementação das funcionalidades, testes e ajustes finais).

Tabela 2. Cronograma

	Jul	Ago	Set	Out	Nov	Dez	Fev	Mar	Abr	Mai
Redação e entrega da Proposta	X									
Revisão bibliográfica	X	X								
Pesquisa de Trabalhos Relacionados	X	X								
Estudo de ferramentas de ***Backend***		X								
Criação da estrutura no ***Backend***		X								
Levantamento de requisitos		X	X							
Desenvolvimento dos diagramas		X	X							
Redação e entrega do TFG I		X	X	X	X					
Apresentação TFG I					X					
Término do desenvolvimento						X	X	X		
Testes finais									X	
Redação e entrega do TFG II								X	X	X
Apresentação TFG II										X

4.1. Cronograma

Referências

[Borges 2014] Borges, L. E. (2014). *Python para Desenvolvedores: Aborda Python 3.3*. Novatec Editora.

[CETIC 2017] CETIC (2017). TIC domicílios - Centro Regional para o Desenvolvimento da Sociedade da Informação. Disponível em http://cetic.br. Acessado em setembro de 2017.

[dos Santos et al. 2015] dos Santos, R. E., da Silva, R. F., and de Oliveira Zamberlan, A. (2015). Proposta de uma plataforma de cloud computing para disponibilização de um sistema online para consultórios e clínicas por meio do modelo saas. In *XIII Simpósio de Informática - SIRC*, page 90. Centro Universitário Franciscano.

[Freeman and Freeman 2009] Freeman, E. and Freeman, E. (2009). *Use a Cabeça - Padrões de Projetos*. Alta Books Editora, 2nd edition.

[Google 2017a] Google (2017a). Conheça o android studio. Disponível em https://developer.android.com/studio/intro/index.html. Acessado em outubro de 2017.

[Google 2017b] Google (2017b). Documentation. Disponível em https://firebase.google.com/docs. Acessado em setembro de 2017.

[Harrop 2001] Harrop, V. M. (2001). Virtual healthcare delivery: defined, modeled, and predictive barriers to implementation identified. In *Proceedings of the AMIA Symposium*, page 244. American Medical Informatics Association.

[Laudon and Laudon 2014] Laudon, J. P. and Laudon, K. C. (2014). *Sistemas de Informação Gerenciais*. Pearson, 11st edition.

[Lecheta 2010] Lecheta, R. R. (2010). *Google Android*. Novatec Editora, 2nd edition.

[McLaughlin et al. 2007] McLaughlin, B., Pollice, G., and West, D. (2007). *Use a Cabeça - Análise e Projeto Orientado ao Objeto*. Alta Books Editora.

[Meirelles 2016] Meirelles, F. (2016). 27ª pesquisa anual do uso de tecnologia da informação. Disponível em http://eaesp.fgvsp.br/sites/eaesp.fgvsp.br/files/pesti2016gvciappt.pdf. Acessado em setembro de 2017.

[Pressman 2010] Pressman, R. S. (2010). *Software Engineering: a Practitioner's Approach*. McGraw-Hill Education, New York, 7th edition.

[Rymsha 2017] Rymsha, R. (2017). Who is the Boss in da House? Frontend vs Backend vs Fullstack. Disponível em https://www.linkedin.com/pulse/who-boss-da-house-frontend-vs-backend-fullstack-roma-rimsha. Acessado em outubro de 2017.

[Silveira et al. 2012] Silveira, P., Silveira, G., Lopes, S., Moreira, G., Steppat, N., and Kung, F. (2012). *Introdução à arquitetura e Design de Software*. Elsevier Editora.

[StatCounter 2017] StatCounter (2017). Operating system market share worldwide. Disponível em http://gs.statcounter.com/os-market-share. Acessado em outubro de 2017.

[TIOBE 2017] TIOBE (2017). Tiobe index for october 2017. Disponível em https://www.tiobe.com/tiobe-index/. Acessado em outubro de 2017.

[Veras 2012] Veras, M. (2012). *Cloud Computing: nova arquitetura da TI*. Brasport Editora.

[VisualWebz.com 2017] VisualWebz.com (2017). Front End vs Back End developers. Disponível em https://visualwebz.com/front-end-vs-back-end-developers/. Acessado em outubro de 2017.

[Yin 2015] Yin, R. K. (2015). *Estudo de Caso: Planejamento e Métodos*. Bookman editora.