PONTIFÍCIA UNIVERSIDADE CATÓLICA DO RIO DE JANEIRO

Pau pra toda obra

Aplicativo de contratação de serviço de trabalhadores autônomos

Árthus James Serrano Erthal

PROJETO FINAL DE GRADUAÇÃO

CENTRO TÉCNICO CIENTÍFICO - CTC

DEPARTAMENTO DE INFORMÁTICA

Curso de graduação em Ciência da Computação

Rio de Janeiro, Junho de 2023

Árthus James Serrano Erthal

Pau pra toda obra

Aplicativo de contratação de serviço de trabalhadores autônomos

Relatório de Projeto Final, apresentado ao programa de Projeto Final da PUC-Rio como requisito parcial para a obtenção do título de Bacharel em Ciência da Computação.

Orientador: Roberto Ierusalimschy

Rio de Janeiro, Junho de 2023

Resumo

Erthal, Árthus James. Ierusalimschy, Roberto. Pau pra toda obra: Aplicativo de contratação de serviço de trabalhadores autônomos. Rio de Janeiro, 2023. 44 p. Relatório final de Projeto Final de Graduação - Departamento de Informática. Pontifícia Universidade Católica do Rio de Janeiro.

Este projeto consiste no desenvolvimento do aplicativo '*Pau pra toda obra*', desenvolvido com *Unity Real Time Development Platform* em integração com bibliotecas do *Google Firebase*. O aplicativo foi desenvolvido para sistemas Android com o intuito de disponibilizar uma plataforma em que trabalhadores autônomos possam publicar anúncios de seus serviços e seus potenciais clientes possam publicar suas demandas, permitindo unir pares de trabalhador e cliente em uma plataforma onde podem conversar um com o outro sem comprometerem informações pessoais, e trocar informações de contato caso seja desejado.

Palavras-chave:

Google Firebase, Unity Engine, Aplicativo, Dispositivos móveis, Android, C#

Abstract

Erthal, Árthus James. Ierusalimschy, Roberto. *Pau pra toda obra*: An app for hiring self-employed workers. Rio de Janeiro, 2023. 44 p. Relatório final de Projeto Final de Graduação - Departamento de Informática. Pontifícia Universidade Católica do Rio de Janeiro.

This project consists of the development of the '*Pau pra toda obra'* app, developed using Unity Real Time Development Platform, in integration with the Google Firebase libraries. The app was developed for Android systems with the goal of providing a platform where self-employed workers can post announcements of their services while their potential clients can share their demands, allowing for the union of client-worker pairs in a platform where they can text each other without compromising their personal data, and exchange contact information if they so desire.

Keywords:

Google Firebase, Unity Engine, Application, Mobile, Android, C#

Sumário

1. Introdução

Devido à pandemia do COVID-19, o mundo inteiro entrou em crise. Com a quarentena imposta para combater o vírus, muitos negócios vieram à falência, tornando muitos trabalhadores desempregados. Perdendo suas fontes de renda, muitos brasileiros recorreram a formas de trabalho informal: pedreiros, pintores, babás, cuidadores, faxineiros, diaristas, entre vários outros serviços. Essa classe de trabalhadores depende muito da divulgação de seus serviços em suas comunidades para serem contratados e, com baixa fonte de renda, essa divulgação quase sempre é feita na conversa e no conhecimento da pessoa no dia a dia. Mas, em cidades maiores, onde as pessoas têm menos contato com a vizinhança, é difícil fazer essa troca sem ter que recorrer ao marketing digital, que muitas vezes está acima do orçamento de muitos trabalhadores deste grupo.

Apesar de existirem aplicativos no mercado que disponibilizam formas de divulgar estes serviços, muitos cobram taxas mensais para serem usados, reduzindo o lucro deste trabalhador que já tem renda baixa. E os que não têm taxas comumente têm diversos outros tipos de limitações, como serem especializados demais em apenas um tipo de serviço, ou só atender a usuários em regiões limitadas, deixando boa parte da população sem acesso ao aplicativo.

Enquanto isso, do outro lado da transação, há clientes em potencial buscando esses serviços que não encontram aplicativos no mercado para auxiliar na procura. Quantas pessoas poderiam estar procurando um profissional para realizar um trabalho em casa, mas que não conhecem em sua região ninguém que possa realizá-lo? Poucas das opções do mercado permitem que o contratante publique suas necessidades para trabalhadores se apresentarem para o serviço.

O objetivo deste projeto, portanto, foi criar o aplicativo Pau Pra Toda Obra, um aplicativo único que atenda a ambos os públicos para reuni-los e facilitar a formação da conexão entre trabalhador e empregador, permitindo que os dois grupos publiquem o que cobram e o que oferecem e que troquem mensagens para estabelecer um contato e chegarem a um acordo.

O aplicativo Pau Pra Toda Obra foi desenvolvido para dispositivos Android, por ser um sistema mais acessível no dia a dia, facilitando não apenas uma

disponibilidade do aplicativo para um maior número de usuários em potencial como os testes durante seu desenvolvimento.

Devido a uma maior experiência com a ferramenta, será utilizada a Unity Real Time Development Platform[9], que, apesar de comumente ser aplicada no desenvolvimento de jogos, não é limitada a essa funcionalidade, além de incluir ferramentas e bibliotecas úteis para agilizar o desenvolvimento e permitir futuramente expandir a aplicação para versões disponíveis em outros sistemas como iOS, Windows e Linux.

A linguagem de programação escolhida para desenvolver o aplicativo foi C#, por ser a linguagem aceita pelo Unity. Para atender às necessidades de acesso, transferência e hospedagem de dados em nuvem que esse tipo de aplicativo demanda, foi usada a plataforma Google FireBase[10], que possui serviços de banco de dados na nuvem, de hosting de servidores e de autenticação, além de ter bibliotecas dedicadas ao desenvolvimento em Unity. Para conectar esses serviços, também foram usados os pacotes Android SDK (*Software Development Kit*) e Firebase Unity SDK.

O desenvolvimento do aplicativo descrito requer conhecimentos de diversos assuntos estudados ao longo do curso, como lógica para programação e programação orientada a objetos (para produzir o código fonte do aplicativo); e estruturação de dados e redes de computadores (para organizar e permitir compartilhamento de conteúdo entre usuários). Portanto, é justificável sua elegibilidade como de projeto final.

2. Situação Atual

Existem alguns aplicativos no mercado, atualmente, com funcionalidades semelhantes. Porém, cada um destes traz consigo algumas limitações:

Uber[1] e UberEats[2]: Aplicativos de contratação de serviço de transporte e de encomenda de comida, respectivamente. Esses aplicativos populares empoderam o trabalhador desempregado com uma opção de renda mais confiável que o trabalho autônomo, contudo muitos brasileiros não têm carro ou moto para prestar tais serviços. Diferentemente dos demais aplicativos dessa lista, é o único que funciona no modelo que estamos chamando de "por comissão": é o cliente quem publica que deseja o serviço e, só então, o trabalhador o aceita, ao invés de o trabalhador anunciar para que o cliente o contrate. Outros aplicativos de entrega também seguem este modelo.

Craigslist[3]: Aplicativo de anúncios de empregos bem similar a esta proposta para o Pau Para Toda Obra. Porém o aplicativo não é muito popular, especialmente no Brasil, e a versão web[4] tem interface antiga e robusta, além de o serviço ser limitado a poucas cidades brasileiras e de nenhuma das duas versões apresentar a opção ao empregador para postar comissões buscando trabalhadores.

SOS Costura[5]: aplicativo de contratação de serviços de costura. Permite ambas as direções de contato nas postagens, mas é limitado ao nicho de serviços de costura. Resenhas na página do aplicativo também indicam que o serviço do aplicativo é lento.

Sitly[6]: aplicativo de contratação de babás. Parte das funcionalidades do aplicativo requer uma assinatura mensal, tornando-o menos acessível.

Parafuzo[7]: aplicativo de contratação de diaristas e faxinas. Abrange uma variedade maior de serviços prestados, porém também é limitado a uma assinatura mensal.

Mãos à Obra[8]: aplicativo de contratação de diversos serviços voltados à obra, tanto para reformas quanto para construção. Também tem o defeito de necessitar que o trabalhador pague para trabalhar, além de burocratizar os pagamentos através de um sistema de compra de créditos e limitar o serviço apenas às cidades do Rio de Janeiro e São Paulo.

Pode-se observar que dentre as opções disponíveis atualmente, poucas disponibilizam as opções de ambos (empregador e trabalhador) anunciarem no aplicativo, simultaneamente, e várias cobram taxas para o uso do serviço, que é indesejável para trabalhadores autônomos que muitas vezes já sofrem com baixa renda. O Pau Pra Toda Obra busca resolver essas duas pendências, sendo de uso gratuito e permitindo que ambos os grupos de usuários anunciem comissões e serviços na mesma plataforma.

3. Objetivos

3.1 Objetivos Gerais

O objetivo deste projeto, portanto, é desenvolver um aplicativo móvel para sistemas Android que permita aos usuários anunciar em sua cidade que tipo de serviços eles buscam e que outros serviços eles estão dispostos a prestar, possibilitando aos interessados estabelecerem um contato inicial para serem

contratados. Para tratar esse objetivo como atendido, o projeto tem como meta, conforme definido na proposta inicial, implementar as seguintes funcionalidades neste aplicativo:

3.2 Registro e acesso de usuários no sistema

Cada usuário deve se registrar no sistema para utilizar os serviços, facilitando o contato entre ele e demais usuários. Para se registrar inicialmente, um usuário deve inserir um endereço de email, seu nome e a senha desejada para acessos futuros. Uma vez registrado, o usuário pode usar o email e a senha inseridos no registro para acessar seus dados no aplicativo em sessões futuras.

3.3 Anúncios de serviço

Usuários trabalhadores podem publicar seus serviços no aplicativo, que serão exibidos para os demais usuários e listados junto com outros serviços publicados, em uma página dedicada para essa exibição. Um anúncio terá descrição, categoria e preço cobrado, e o usuário pode optar por exibir uma lista de horários e dias da semana disponíveis. Os anúncios podem ter imagens em anexo, para exibir evidências da qualidade do serviço, e podem ser filtrados pelo cliente na busca de um contrato dentro de seu orçamento.

3.4 Anúncios de comissão

Os anúncios de comissão são similares aos anúncios de serviços, porém, neste caso, é o cliente quem descreve sua necessidade, enquanto o trabalhador entra em contato para aceitar o serviço. Uma comissão terá descrição, categoria e orçamento, e o usuário pode optar por exibir uma data limite ou dias da semana disponíveis para o serviço. As comissões podem ter imagens em anexo, para mostrar detalhes do serviço desejado, e podem ser filtradas pelo cliente na busca de um contrato que atenda às suas capacidades e necessidades.

3.5 Salas de conversa

Para que trabalhadores e clientes possam se comunicar dentro do aplicativo sem divulgar seus dados pessoais, o sistema deve disponibilizar uma funcionalidade de conversa que permita dois usuários estabelecerem este primeiro contato, para então trocarem informações caso desejem se comunicar por outros meios. Uma sala de conversa será, portanto, uma tela no aplicativo onde dois usuários podem trocar mensagens instantâneas entre si.

3.6 Avaliação de usuários

Ao confirmar um contrato, os dois usuários envolvidos devem ter a opção de avaliar um ao outro, tornando público seu desempenho e incentivando organicamente o tratamento adequado dos trabalhadores e o cumprimento correto dos contratos.

3.7 Perfil do usuário

Cada usuário terá sua página pessoal, onde pode preencher informações básicas de contato por fora do aplicativo e onde outros usuários podem ver seus anúncios ativos e avaliações de contratos anteriores.

4. Estudos Realizados

4.1 Estudos preliminares

Conforme descrito anteriormente, a Engine Unity foi escolhida para o projeto devido a uma familiaridade maior com o ambiente de desenvolvimento[11] e por ser acompanhada de ferramentas, bibliotecas e um ambiente de edição muito versáteis para desenvolver a parte visual do aplicativo de forma ágil e acompanhada de pré-visualização. Como já houve experiência anterior no desenvolvimento nesta plataforma, já foram estudadas a linguagem C# e algumas bibliotecas principais do Unity antes do início deste projeto, inclusive as ferramentas necessárias para o desenvolvimento para Android, como no caso do Pau Pra Toda Obra.

4.2 Estudos conceituais e de tecnologia

Para apoiar as necessidades de acesso a dados em nuvem e em banco de dados do projeto atual, foi necessário estudar a plataforma Firebase e suas bibliotecas para integração com Unity. Três destas, em especial, foram utilizadas para desenvolver as funcionalidades propostas: o pacote Firebase Auth, que disponibiliza métodos para registrar e autenticar usuários no sistema; Firebase Database, que permite salvar, remover, atualizar e ler dados estruturados no servidor Firebase do projeto; e por fim o Firebase Storage, que torna possível guardar arquivos na nuvem para serem recuperados mais tarde.

4.3 Testes e protótipos para aprendizado e demonstração

Para estudar o funcionamento dessas bibliotecas, foram produzidos protótipos com as funcionalidades básicas desejadas para o aplicativo, buscando na documentação oficial[12] as classes e métodos apropriados para a implementação do que foi proposto e a forma adequada de utilizá-los.

Os protótipos produzidos foram reunidos por uma interface básica desenvolvida no Unity para serem testados, amarrando os protótipos num menu principal único para navegar entre eles (conforme exibido na Fig 1, a seguir). Os botões exibidos ao centro desabilitam o menu principal temporariamente para deixar de exibi-lo e habilitam as telas associadas a seus respectivos protótipos, identificados por seus rótulos. No canto inferior direito, o botão rotulado "Logout" permite desconectar o usuário do sistema caso esteja atualmente autenticado.

Os campos de texto "<exemplo@teste.com>"; e "<exemplo mensagem de erro>" são substituídos em tempo real por, respectivamente, o email do usuário atualmente autenticado no sistema e quaisquer mensagens de erro que possam surgir em tentativas falhas de usar os protótipos. Ambos os textos permanecem visíveis, independentemente do menu de protótipo que está sendo acessado, facilitando o diagnóstico dos erros no desenvolvimento e o tratamento de casos de teste, porém só se fazem visíveis quando preenchidos, exceto pelo caso dado como exemplo. Todos os protótipos possuem em comum um botão rotulado "Voltar", que traz o usuário de volta para este menu inicial.

Fig. 1 Menu principal dos Protótipos Pau pra toda obra.

Os protótipos do sistema de registro de novo usuário e autenticação de usuário existente foram separados em dois menus acessíveis, respectivamente, pelos botões rotulados "Registrar" e "Login", demonstrados na figura 1. Para desenvolver o protótipo de ambas as funcionalidades, foi estudado o pacote Firebase Authentication[13]. Como o projeto desenvolvido não tem nenhuma demanda especial sobre a autenticação e registro de novos usuários, não foi necessário fazer uma implementação muito diferente do exemplo proposto na documentação para atender às necessidades do aplicativo.

No protótipo de registro (Fig. 2, a seguir), duas caixas de texto recebem como entrada o email e a senha desejados para o novo usuário, e o botão rotulado "Registrar" envia os valores inseridos para fazer uma tentativa de registrar um novo usuário com esses valores, com a função CreateUserWithEmailAndPasswordAsync. O método testa se os valores inseridos possuem problemas, como campos não preenchidos, email inválido ou já utilizado por outro usuário, ou senha fraca. E, caso as credenciais inseridas sejam válidas, registra o novo usuário e já autentica seu acesso ao sistema na sessão atual.

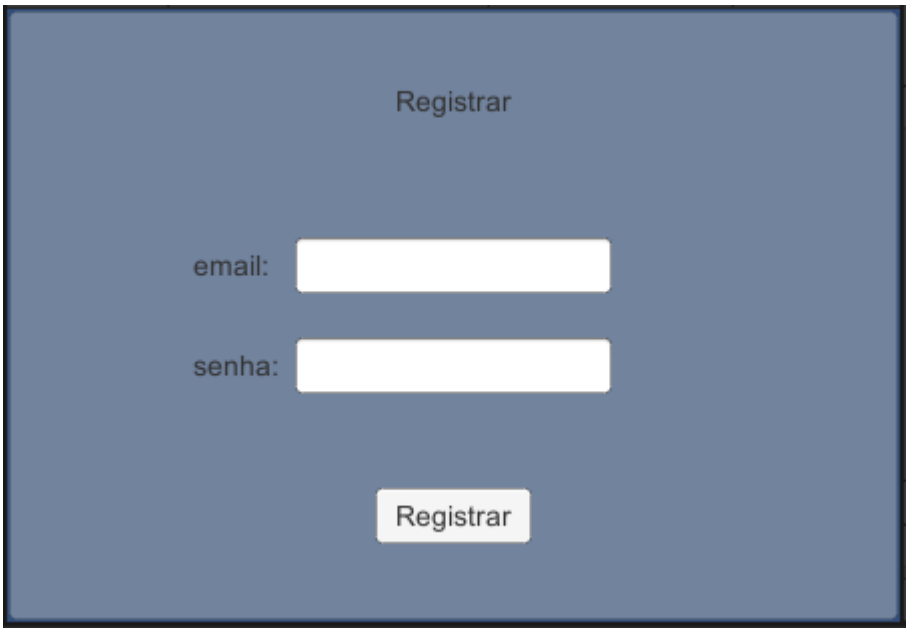

Fig. 2 Menu de protótipo de registro de novo usuário

O menu de protótipo de autenticação de usuário (Fig. 3) é bem similar ao de registro, por serem telas complementares. O botão "Registrar" é substituído por um botão "Entrar", que chama a função SignInWithEmailAndPasswordAsync. Esse método tenta autenticar no sistema

usando o email e a senha inseridos. Assim como no protótipo de registro, credenciais inválidas resultam em uma mensagem de erro, enquanto usar o email e senha corretos de um usuário já registrado autentica o seu acesso ao sistema.

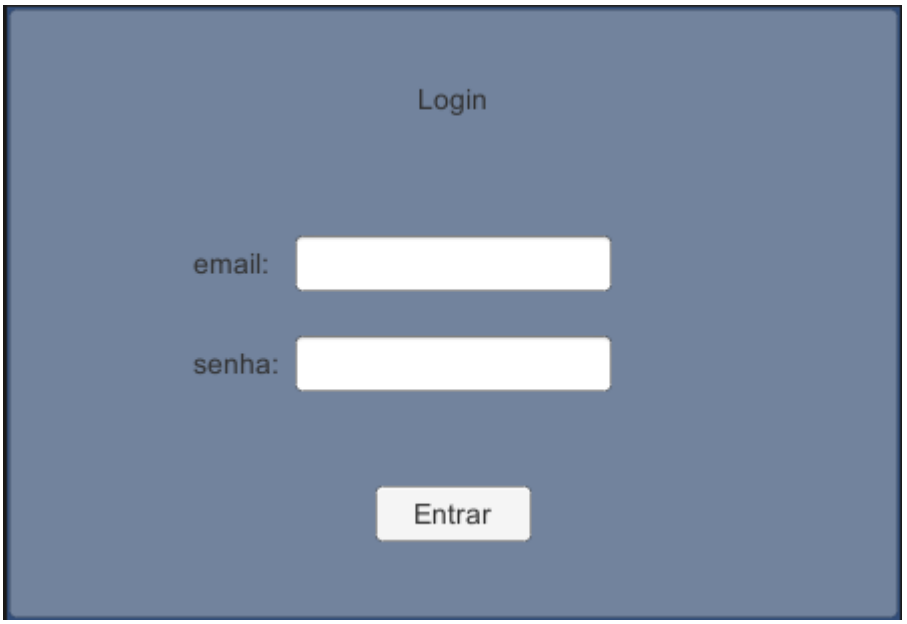

Fig. 3 Menu de protótipo de autenticação de usuário existente

Para desenvolver o protótipo do sistema de conversa entre usuários, no menu acessado pelo botão rotulado "Chat" do menu principal, foi utilizado o pacote Firebase Realtime Database[14], que permite transferir dados para o servidor Firebase do projeto de forma estruturada, e recuperar esses dados. aplicando filtros e ordenações adicionais conforme necessário. No caso do compartilhamento de mensagens, foram usados filtros para que apenas os dois usuários envolvidos numa conversa tenham acesso às mensagens, que depois são ordenadas por data de publicação para garantir que serão exibidas na ordem de envio.

No menu de conversa (Fig. 4), uma caixa de texto no alto da tela permite ao usuário inserir o email de seu destinatário. No centro da porção esquerda do menu, um botão rotulado "Refresh" usa uma combinação de métodos do Firebase para acessar o servidor e recuperar as mensagens trocadas entre o usuário usando o sistema na sessão atual e o usuário registrado com o email inserido na caixa de texto. Não há mecanismos para tratar erros quanto a essa seleção livre de destinatário porque a intenção é que a versão definitiva do sistema trate essa seleção através da navegação das telas. Portanto, apenas usuários registrados no sistema podem ser destinatários de mensagens.

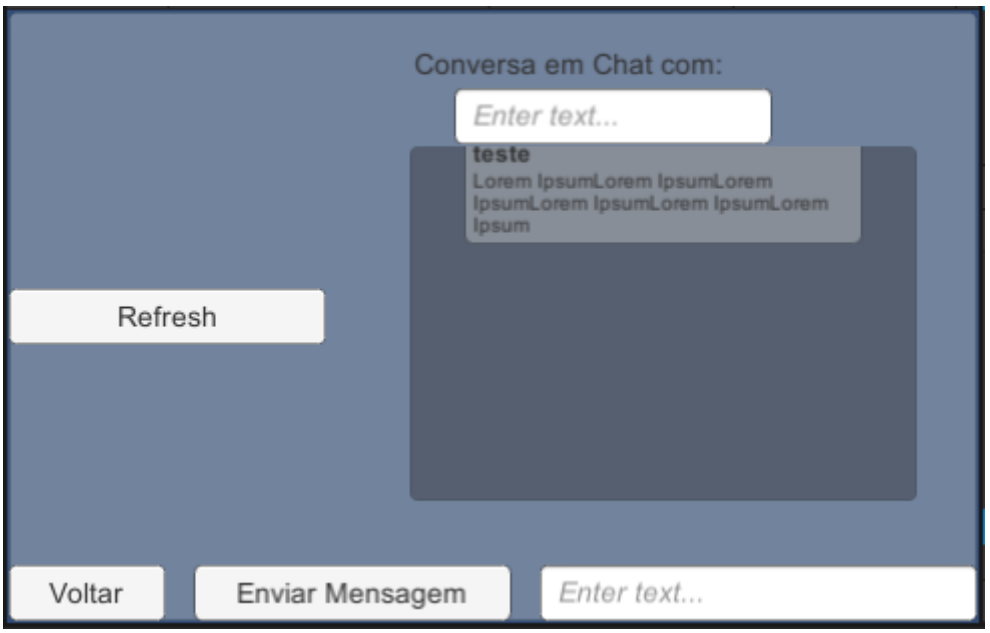

Fig. 4 Menu de protótipo de troca de mensagens

Na região à direita da página, um painel exibe em forma de lista vertical todas as mensagens trocadas pelos dois usuários, conforme o exemplo, porém tendo o texto "teste" substituído pelo email do remetente e o corpo substituído pelo texto inserido no momento do envio.

Na parte inferior, uma caixa de texto recebe o texto desejado para a mensagem e o botão onde está escrito "Enviar Mensagem" executa uma combinação de métodos do Firebase para salvar uma nova mensagem no servidor, com o email do usuário da sessão atual sendo registrado como remetente, o email inserido no topo da página como destinatário, com o corpo da mensagem inserido na caixa de texto inferior e com o horário atual para fins de ordenação das mensagens.

Para validar a possibilidade de publicar imagens associadas aos outros dados e para preparar uma versão inicial da funcionalidade principal de publicação de contratos, o último protótipo, acessado pelo botão "Ver e adicionar dados" no menu principal, combina o uso do Firebase Realtime Database com o pacote Firebase Storage[15], funcionalidade voltada à publicação de arquivos inteiros no servidor, necessário para permitir o uso de imagens nos perfis e nos contratos.

O menu de protótipo de publicação e listagem de contratos (Fig. 5) é dividido em duas seções. A metade esquerda do menu apresenta os campos de texto para inserir o título e a descrição desejados para o contrato, um botão "Escolher arquivo", que abre o navegador de arquivos do dispositivo para selecionar uma imagem, um painel para exibir a imagem selecionada à

esquerda deste botão e um botão "Postar" na parte inferior para salvar o contrato, transferindo a imagem selecionada para o Firebase Storage com um nome único e recebendo o endereço em que ela foi armazenada, que é salvo junto ao resto das informações do contrato no Firebase Database para ser acessado novamente no carregamento do arquivo. Na seção que compõe a direita do menu, um painel (similar ao do protótipo de envio de mensagens) exibe uma lista dos contratos postados, com um elemento servido como exemplo, no qual o rótulo "<Titulo>" seria substituído pelo título definido para o contrato sendo exibido, o texto "Lorem Ipsum" abaixo exibindo a descrição, e o painel branco exibindo a imagem selecionada no momento da publicação, carregada a partir do endereço salvo nos dados do contrato.

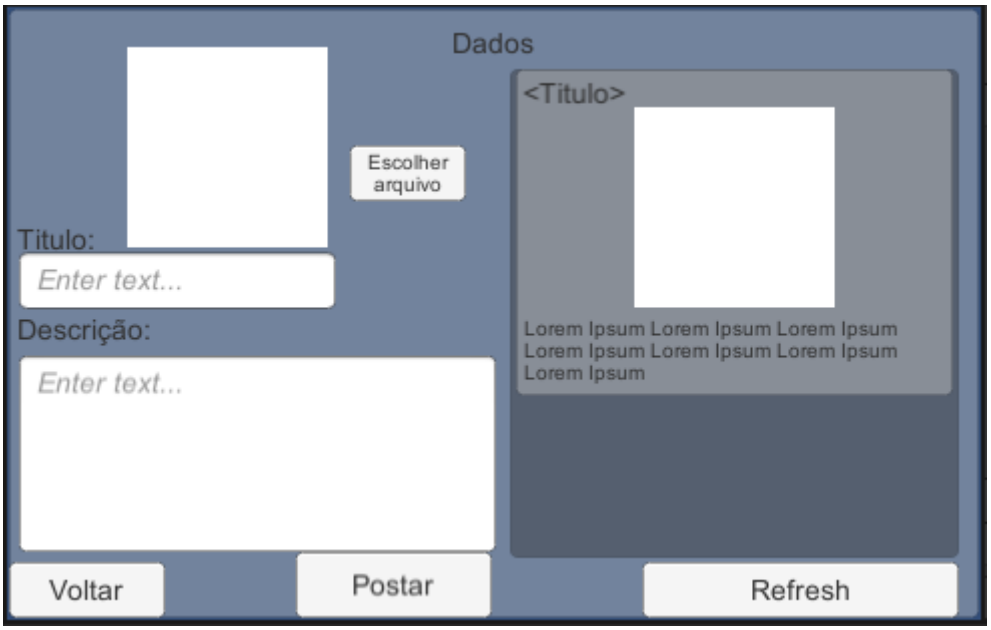

Fig. 5 Menu de protótipo de leitura e publicação de dados no servidor do sistema

Preparados estes 4 protótipos, determinou-se que o que foi estudado das bibliotecas selecionadas seria suficiente para desenvolver a versão completa do aplicativo proposto.

4.4 Método

Antes de começar o desenvolvimento da versão definitiva do Pau Pra Toda Obra, projetou-se um diagrama do fluxo de telas (Fig. 6) inicialmente pretendidas para o sistema, para contextualizar melhor as necessidades de cada funcionalidade para decidir a melhor forma de estruturar os dados que seriam usados.

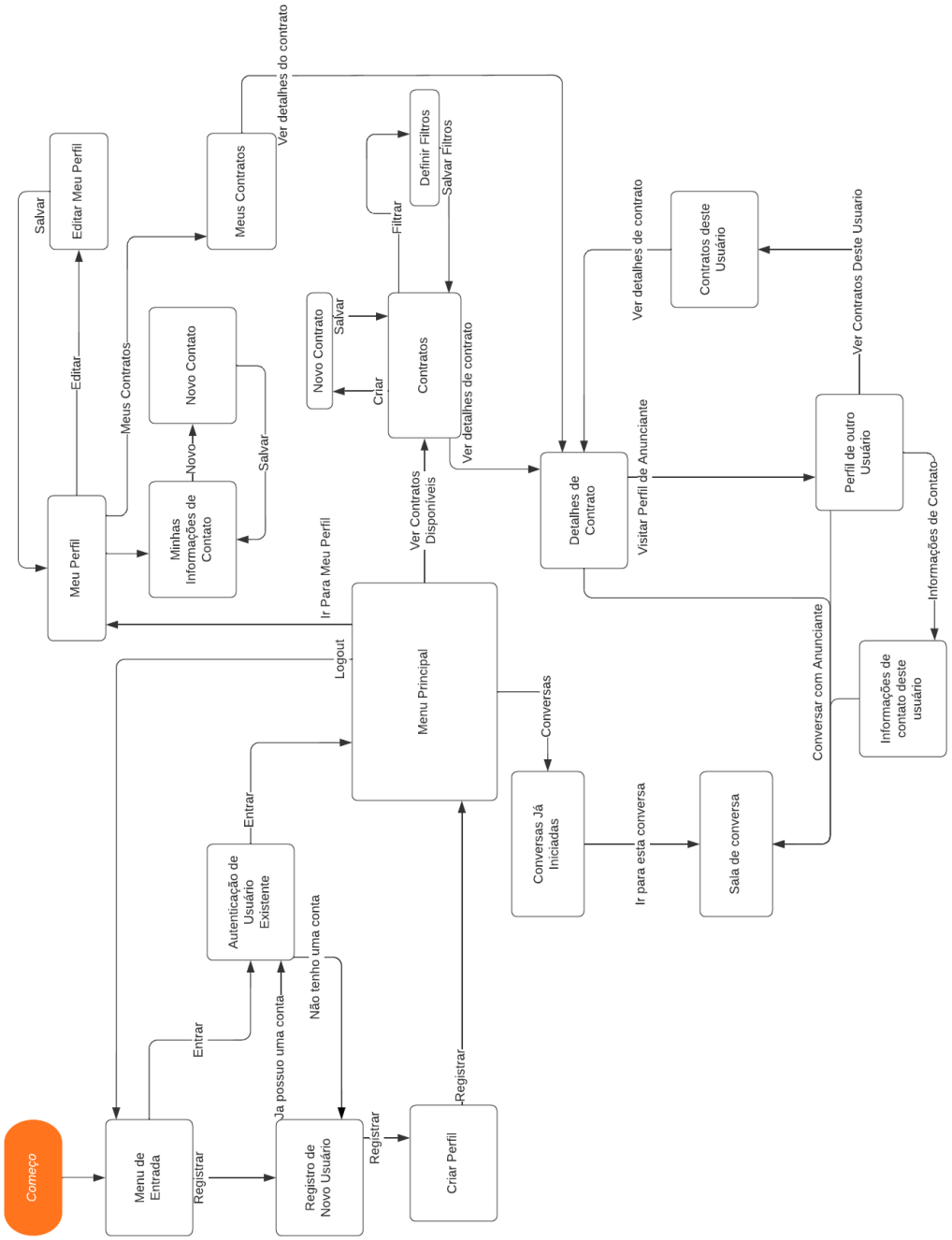

Fig 6 Diagrama de fluxo de telas

A seguir, avaliou-se, baseado nas páginas projetadas para o aplicativo, que tipos de dados seriam necessários para a implementação e como estes estariam estruturados.

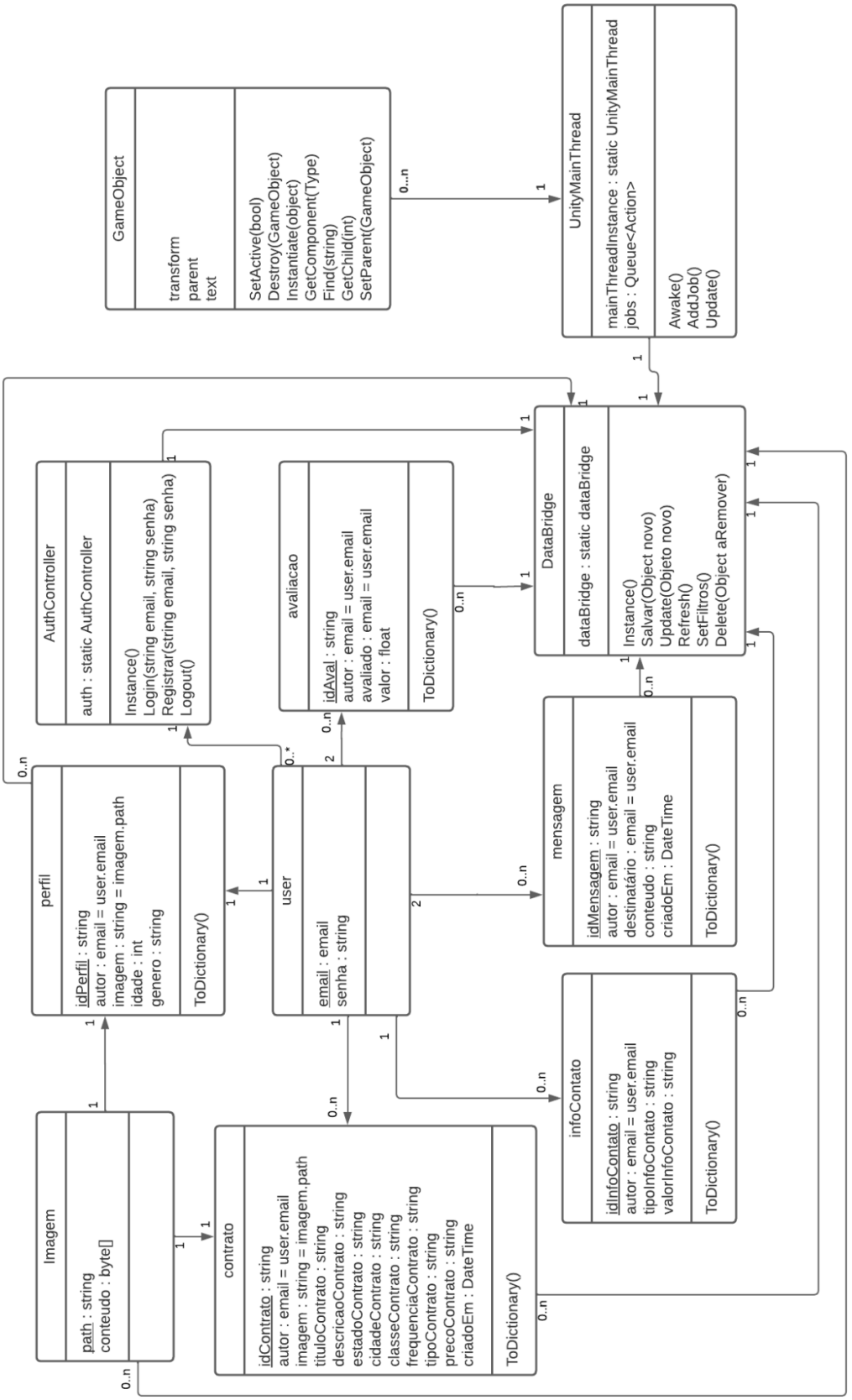

Fig. 7 Diagrama de classe

A classe 'User' é gerenciada pelo pacote Firebase Authentication, que, apesar de na realidade possuir mais propriedades, apenas o email e a senha inseridos pelo usuário no momento do registro são de fato relevantes para o escopo do projeto. As operações de Registro, Autenticação e Desconexão de usuários por sua vez são organizadas pela classe Singleton AuthController. As demais informações do usuário pertinentes ao sistema são armazenadas na classe 'Perfil', que possui o email de seu autor, sua idade, seu gênero e o endereço da imagem associada. Um Contrato possui autor, um momento de criação criadoEm, um título, uma descrição, a classe do contrato (se diz respeito a um trabalho que o autor faz, padronizado como "serviço", ou um trabalho que ele quer que seja feito por alguém, "comissão"), uma lista de dias da semana em que o serviço descrito no contrato é oferecido (registrado como 'frequenciaContrato'), o estado e cidade em que o contrato é prestado ou cobrado, o tipo de serviço prestado ou procurado ("obra", "faxina", "babá", ou "outro", essa lista podendo ser expandida conforme for necessário), o preço cobrado ou oferecido pelo contrato e o endereço da imagem associada ao contrato. Uma informação de contato InfoContato possui autor, tipo de informação (se é endereço, email, telefone, celular, ou outra forma de contato inserida pelo usuário) e a informação necessária para realizar o contato externo. Uma mensagem Mensagem possui autor, destinatário, conteúdo e instante de criação. Objetos das classes Perfil, Contrato, Contato, Mensagem e Imagem são armazenados, recuperados, atualizados e excluídos do servidor pelo Singleton DataBridge. O Singleton UnityMainThread gerencia todas as chamadas de métodos de GameObject que não podem ser executados em threads secundárias, refletindo então as interações do DataBridge com a interface.

Como na disciplina de Projeto Final I, as tarefas necessárias para desenvolver este projeto foram determinadas e listadas a seguir:

1. Escrita da Proposta
2. Criação de protótipos para verificar que as ferramentas escolhidas atendem a todas as necessidades do aplicativo
   a. Tela e funcionalidade de registro de novos usuários
      i. Entradas de texto para email e senha
      ii. Registrar dados inseridos no servidor
   b. Tela e funcionalidade de acesso de usuário existente
      i. Entradas de texto para email e senha
      ii. Autenticar a entrada do usuário com os dados no servidor
   c. Tela e funcionalidade para adicionar novos dados ao servidor

i. Entradas de texto, números e imagens para diversos campos de um único objeto

ii. Armazenar objeto no servidor

d. Tela exibindo lista de dados, que serão recebidos do banco de dados do Firebase, conforme filtro inserido pelo usuário

i. Receber dados de vários objetos (criados pela tela do item 2.c) armazenados no servidor

ii. Exibir dados em lista

iii. Permitir edição e remoção dos dados

3. Modelagem de dados

4. Escrita do Relatório

5. Desenvolvimento do aplicativo

a. Criação das telas do sistema com suas funcionalidades

i. Registro de novo usuário

ii. Acesso de usuário existente

iii. Lista de serviços

iv. Lista de comissões

v. Perfil de usuário

vi. Sala de conversa

vii. Avaliação de serviço

b. Incorporação gradual dos protótipos no sistema com ajustes conforme necessários

c. Criar transição entre as diferentes telas do sistema

6. Testes do sistema com usuários

7. Reajustes do sistema conforme resultados de testes

8. Preparação da apresentação final

Para o relatório redigido em Projeto Final I, foi elaborado um cronograma para planejar a execução destas atividades. Agora com o projeto completo, é possível fazer uma retrospectiva de que atividades foram planejadas e quais foram cumpridas dentro dos planos, e fazer uma análise sobre esses dados.

Tarefa	1	2	3	4	5	6	7	8
Abr	O							
Mai	O	O		O				
Jun		O		O				
Jul		O		O				
Ago			O	O				
Sep				X	X			
Out				X		X		
Nov				X			X	
Dez				X				X

Tabela 1: Cronograma desenvolvido para a disciplina de Projeto Final I.

Células marcadas com "O" indicam tarefas concluídas em Projeto Final I, enquanto as marcadas com "X" projetam os meses planejados para as tarefas restantes para concluir em Projeto Final II

Tarefa	1	2	3	4	5	6	7	8
Abr-2022	O							
Mai-2022	O	O		O				
Jun-2022		O		O				
Jul-2022		O		O				
Ago-2022			O	O				
Sep-2022					O			
Out-2022					O			
Nov-2022				O	O	O	O	
Dez-2022				O	O		O	
Jan-2023				O			O	
Fev-2023							O	

Mar-2023							O	
Abr-2023				O			O	
Mai-2023				O		O	O	
Jun-2023				O			O	O

Tabela 2: Mês de conclusão das tarefas estabelecidas para Projeto Final

Comparando-se os dois cronogramas, se destaca a discrepância do tempo que foi necessário para o desenvolvimento das telas e funcionalidades planejadas para o aplicativo (Tarefa 5). Isso ocorreu principalmente devido a dificuldades na transição dos testes do aplicativo, a princípio feitos no ambiente de desenvolvimento do Unity, para suas primeiras versões no sistema Android. Nessas primeiras versões, observou-se que algumas das funcionalidades planejadas e testadas nos protótipos, apesar de adequados dentro do ambiente de teste da plataforma Unity, não funcionavam como o planejado no ambiente Android, portanto foi necessário buscar alternativas, levando à extensão do cronograma do projeto para mais um semestre. A gravidade destes defeitos também necessitou que os testes realizados fossem aplicados novamente, para garantir que as alternativas encontradas não conflitavam com as demais partes do sistema.

5. Projeto e Especificações do Sistema

5.1 Módulos Desenvolvidos

Como descrito, o aplicativo Pau Pra Toda Obra usa o pacote Firebase Realtime Database para tratar suas necessidades de armazenamento, sincronização e recuperação de dados. Este serviço armazena os dados em um banco de dados orientado a documentos, em formato JSON. Portanto, o primeiro conjunto de módulos desenvolvidos para este sistema é composto pelas classes associadas aos objetos relevantes para o escopo, definidas no diagrama de classes na sessão anterior. Logo, os módulos para as classes InfoUsuario (para armazenar dados de perfis de usuários), Comission (para dados de contratos), Contato (para informações de contato de usuários), Avaliacao (para usuários avaliarem a qualidade dos serviços) e Message (para mensagens trocadas entre usuários) foram criados. Todos estes módulos

possuem apenas duas funções: seu construtor, para criar um novo objeto da classe, e uma função *ToDictionary*(), para converter os dados do objeto em um dicionário de dados. As bibliotecas FireBase fazem uso da organização dos dados como dicionário para facilitar a inserção no banco. Vale ressaltar que, apesar de serem definidos no diagrama de classe, não foram desenvolvidos módulos para objetos do tipo Imagem, Usuário nem GameObject, pois estes são elementos gerenciados, respectivamente, pelos pacotes Firebase Storage, Firebase Authentication, e pela própria Unity. Estes são listados no diagrama apenas para maior clareza da relação entre os demais objetos e a origem de seus dados.

A interface visual apresentada ao usuário foi desenvolvida inteiramente na plataforma Unity, incluindo o posicionamento de todos os elementos da tela e as chamadas de métodos executadas pelos botões exibidos. Para fazer a conexão do aplicativo em Unity com o banco de dados, os demais módulos *UnityMainThread*, *DropdownController*, *AuthController*, *MenuController* e *DataBridge*.

Devido à natureza do aplicativo sendo desenvolvido, muitas das operações executadas demandam acesso ao banco de dados, se tratando portanto de operações assíncronas que prosseguem sua execução em threads secundárias após receber a resposta do banco. Contudo, os métodos herdados do Unity que foram utilizados para tratar a exibição de informações na tela só podem ser executados na *thread* principal. O módulo UnityMainThread é um Singleton que gerencia uma fila de tarefas e as executa em ordem de chegada na *thread* principal, criado para contornar essa limitação dos objetos Unity usados na interface.

Para a funcionalidade de publicação de contrato, um dos valores que devem ser inseridos durante a criação de um novo contrato é a localização onde o usuário cobrará ou prestará o serviço anunciado no contrato, registrada como um par de estado e cidade. Para tornar a seleção dessa localização intuitiva, em ambas as telas de 'criação de novo contrato' e 'filtragem de contratos', onde essa seleção é realizada, cada tela possui um par de menus *dropdown:* o primeiro listando os 26 estados brasileiros mais o Distrito Federal e o segundo sendo atualizado de acordo para exibir as cidades do estado selecionado. Essa atualização do segundo menu é realizada pelo módulo DropdownController, que é inicializado com uma lista para cada estado, contendo todas as suas cidades, segundo listagem do IBGE[16], e carrega no *dropdown* de seleção de cidades as

opções de acordo com o estado selecionado, somando uma opção inicial que contém rótulo '--Nenhuma Selecionada--' no topo da lista.

A seguir, o módulo AuthController encapsula todas as chamadas de funções do pacote Firebase Authentication. Essa classe é responsável por receber os dados inseridos pelo usuário nos menus de registro de novo usuário e de autenticação de usuário existente, e realizar a tentativa de entrada no sistema com essas credenciais. O módulo também possui métodos para, caso os valores inseridos nas caixas de texto sejam inválidos, incorretos ou nulos, gerar uma mensagem compreensível para o usuário baseada no código de erro recebido do Firebase. Por fim, esse módulo também possui um método para desconectar o usuário do sistema quando ele desejar finalizar sua sessão ou acessar com outra conta.

Para possibilitar uma navegação mais confortável pelas telas do sistema, foi criado o módulo MenuController, que armazena, em estrutura de pilha, a ordem das páginas acessadas pelo usuário na sessão atual, seus títulos e quaisquer valores auxiliares necessários para carregar do servidor os dados a serem exibidos, e disponibiliza um botão 'Voltar', que retorna o usuário ao menu acessado imediatamente antes do atual com as informações adequadas carregadas. A funcionalidade de 'Voltar' também é executada ao utilizar o botão 'Anterior' nativo do sistema Android.

Por fim, o último módulo desenvolvido foi o DataBridge, que é responsável por conectar a interface do usuário aos demais módulos e por salvar, alterar, recuperar e remover dados no servidor Firebase, além de exibir as listas de dados recuperados na interface.

5.2 O Módulo DataBridge

Este módulo possui muito mais métodos que os demais, pois este é responsável por conectar os elementos de interface do Unity aos conjuntos adequados de instruções para armazenar, recuperar, atualizar e remover os dados no Firebase Realtime Database e no Firebase Storage.

Para a classe InfoUsuario, o módulo DataBridge disponibiliza um método para criação de novos dados, e para recuperação do objeto InfoUsuario de acordo com o email de seu autor. Na criação de um InfoUsuário, apenas o nome de exibição do usuário é obrigatório, porém o autor pode escolher fornecer também um arquivo de imagem de formato '.png' em seu dispositivo para que seja exibido como sua imagem de perfil.

Se uma imagem válida for selecionada, para salvá-la no banco de dados e simultaneamente associá-la ao perfil, esta é transferida para o Firebase Storage com um nome único: a data e hora atual do sistema são recebidos como objeto DateTime, registrado com uma precisão de 4 casas decimais, e este é concatenado ao endereço de email do usuário. Com esta combinação, para que duas imagens tenham o mesmo nome de arquivo no banco de dados, seria necessário que o mesmo usuário publicasse duas imagens no mesmo milésimo de segundo, o que é suficientemente improvável de ocorrer durante o funcionamento usual do sistema, e não pode ser usado por um usuário malicioso para manipular os dados inseridos por outros usuários. O título gerado para a imagem é usado como nome do arquivo armazenado no Firebase Storage e como chave única registrada no objeto InfoUsuario para recuperar a imagem quando um usuário carregar as informações deste perfil. Finalmente, quando um usuário finaliza a edição de seu perfil, as informações do objeto InfoUsuario criado ou editado são transferidas para o Firebase Realtime Database. Outro método neste mesmo módulo realiza a operação de recuperação destes dados no banco, preenchendo os campos equivalentes na tela de perfil e a respectiva imagem carregada.

De forma similar, no caso de elementos da classe Contato, também existe em Databridge um método para validar dados inseridos e criar um novo objeto no caso de sucesso. Nesse caso, ambos os campos Tipo de Informação de contato e Valor de Informação de contato são obrigatórios. Outro método neste módulo faz uma leitura dos contatos de um dado usuário e cria na interface uma lista de painéis contendo cada um as informações de um objeto Contato. Caso o usuário acessando essa lista tenha sido também o autor desses Contatos, cada elemento possuirá um botão que permite excluí-lo do banco de dados e, consequentemente, da sua lista.

Objetos da classe Avaliacao são similares aos da classe Contato: um usuário acessando o perfil de outro pode selecionar sua avaliação, com nota entre 1 e 5, selecionada através de um *slider*. Diferentemente dos elementos Contato, um usuário não pode apagar avaliações exibidas no seu perfil. Publicar novamente uma nova Avaliação sobre alguém que o usuário já avaliou não cria um novo elemento, apenas atualizando os valores da Avaliação anterior.

Assim como os exemplos anteriores, objetos Message também possuem métodos nesta classe para serem listadas na interface e criadas, porém a forma como são armazenadas no servidor é muito diferente das demais classes: Enquanto as classes Contanto, Comission, InfoUsuario e Avaliacao são

simplesmente inseridos em listas no servidor, associadas a chaves do mesmo nome da classe, a chave associada a cada lista de elementos Message é o par dos dois participantes da conversa. A lista de pares, por sua vez, é armazenada sob a chave Conversas. Para exemplificar: supondo uma conversa entre os usuários registrados com emails 'a@a.com' e 'b@b.com', uma mensagem trocada entre os dois seria armazenada com chave única no caminho 'conversas/a@a"com b@b"com/', onde o par de emails é sempre ordenado em ordem alfabética para garantir que consistentemente seja acessado o mesmo endereço tanto ao salvar quanto ao ler a lista de mensagens. Como o Firebase não admite o caractere ' . ' (ponto) como parte do endereço de um objeto, este é substituído pelo caractere ' " ' (aspas duplas), que é aceito pelo Firebase, mas não é aceito como caractere válido de um endereço de email. Desta forma podemos garantir que toda instância desse caractere representa um ponto e não há risco de ambiguidade na interpretação dos usuários que têm acesso a este caminho, permitindo a conversão de volta em endereço de email. Enquanto o usuário permanecer na tela de conversa, um *listener* é ativado para atualizar a lista de mensagens exibidas caso uma nova mensagem seja recebida. A lista também é atualizada quando a tela de conversa é acessada.

Além de armazenar a própria mensagem no servidor, a criação de um novo objeto Mensagem também adiciona aos elementos InfoUsuario de ambos participantes da conversa uma chave com o email identificador de seu interlocutor: Por exemplo, ao iniciar uma conversa com um usuário com quem não foi trocada uma mensagem anteriormente, é criado o elemento 'infousuario/conversas/<a@a"com>', onde <a@a"com> é substituído pelo email (convertido com caracteres válidos para o Firebase) do outro usuário envolvido na nova conversa. A propriedade 'conversas', apesar de não possuir um módulo dedicado para defini-la como classe, tem um método associado no módulo DataBridge para listar as conversas já iniciadas do usuário atual.

Por fim, a classe DataBridge também permite integração similar a objetos da classe Comission: estes podem ser salvos com ou sem imagem e ter seus dados exibidos individualmente, similar a um InfoUsuario, e podem ser listados em seu respectivo menu e excluídos do sistema por seus autores, assim como um Contato. Em telas dedicadas do sistema também são exibidos os elementos Comission em lista, porém no caso desta classe é possível filtrar os elementos a serem exibidos segundo parâmetros delimitados pelo usuário. Para cada propriedade de um Comission, há um filtro permitindo exibir apenas os contratos no qual o usuário tem interesse: estado, cidade, frequência, tipo e preço, e

ordenar a lista a ser exibida conforme o momento da criação ou ordem crescente ou decrescente de preço. Para reduzir a possibilidade de confusão entre contratos da classe "Serviço"(em que o autor é o trabalhador buscando um contratante) e "Comissão"(em que o autor é o contratante buscando um trabalhador), a lista de Comission é disponibilizada em dois menus separados, cada um filtrando exclusivamente por uma dessas classes. Finalmente, navegando pelo perfil de um usuário, é possível acessar a lista de Comissions cujo autor seja o dono do perfil, contudo esta não possui outros filtros. Caso um usuário esteja acessando essa lista em seu próprio perfil, ele tem a opção de excluir contratos de sua autoria neste menu.

5.4 Usando o Pau Pra Toda Obra

Tendo-se descrito o funcionamento interno do sistema, resta exemplificar seu funcionamento da perspectiva do usuário. Para isso, será feita uma explicação das telas e menus disponíveis no sistema.

Ao iniciar o aplicativo Pau Pra Toda Obra, o usuário é recebido no menu de autenticação, onde é apresentado ao título junto às opções de botões "Login" e "Registrar" (Fig. 8). Cada botão o leva para a tela com a funcionalidade descrita, onde ele pode inserir as credenciais adequadas para, respectivamente, registrar-se (Fig. 9) ou autenticar-se(Fig. 10) no sistema. Um botão ao fundo de ambas as páginas permite rapidamente transitar de uma para outra, caso o usuário mude de ideia. Se as credenciais inseridas forem incorretas, inválidas, não tiverem sido registradas ou preenchidas, ou não for possível se conectar, um painel exibe uma mensagem de erro adequada (como no exemplo na Fig. 11)

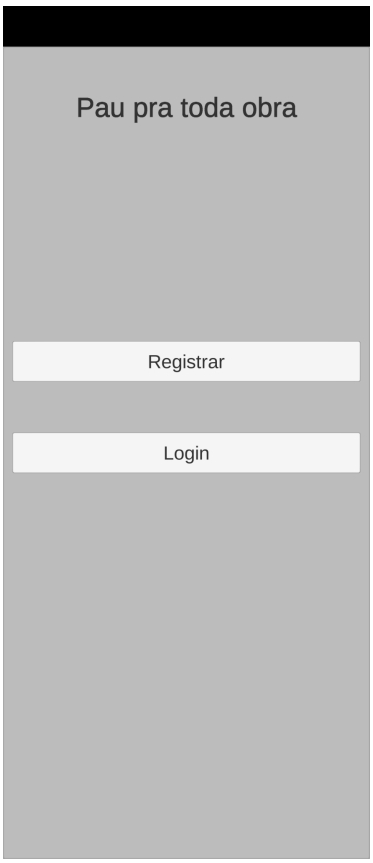

Fig. 8 Menu de entrada

Fig. 9 Menu de registro (Acessado pelo botão "Registrar")

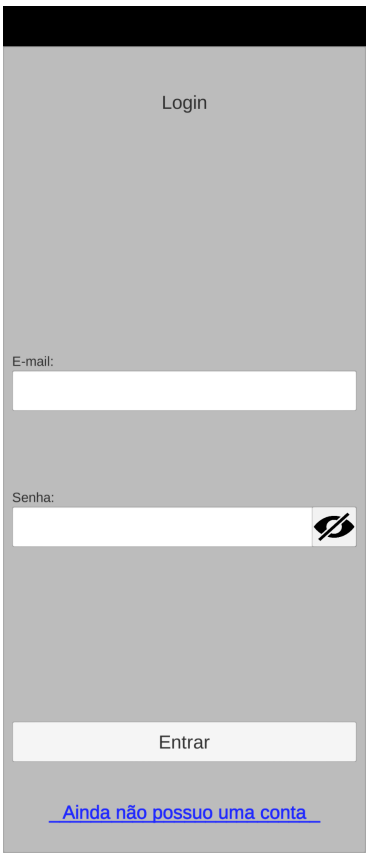

Fig. 10 Menu de autenticação (Acessado pelo botão "Login")

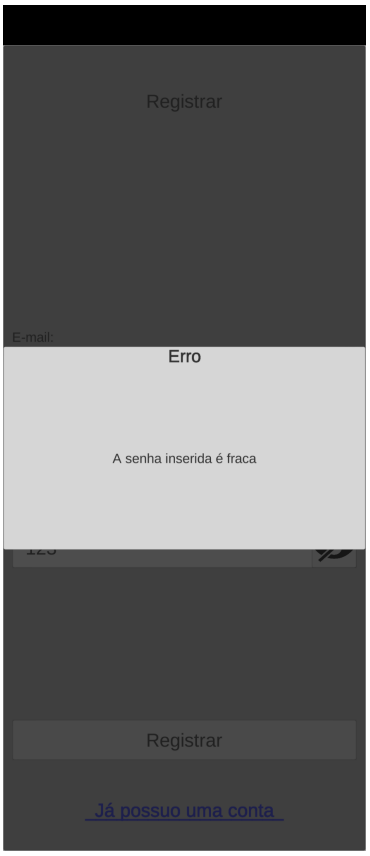

Fig. 11 Exemplo de uma mensagem de erro

Ao se registrar com sucesso, o usuário é enviado para o menu Novo Perfil(Fig. 12) para preencher seu nome e, se desejar, sua idade e gênero, podendo escolher também fornecer um arquivo de imagem '.png' de seu dispositivo para usar como imagem de perfil. A imagem é carregada ao pressionar o botão "Alterar imagem" e uma prévia é exibida no painel na parte superior do menu.

Fig. 12 Menu de edição de perfil

Após o usuário recém-registrado criar seu perfil, ou caso esteja entrando com uma conta já existente, o sistema o traz para o menu principal (Fig. 13). Este menu o conecta às demais funcionalidades do aplicativo com os 4 botões ao centro e permite desconectar da conta atual com o botão "Logout" no canto direito inferior

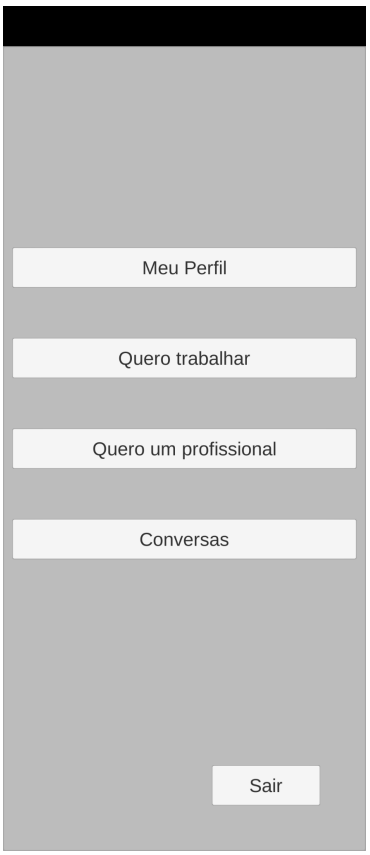

Fig. 13 Menu principal

Tocar no botão "Meu Perfil" leva o usuário à página do próprio perfil contendo as informações preenchidas anteriormente(Fig. 14). Após a criação inicial de seu perfil, em todas as páginas além do menu principal, o topo da tela é preenchido por um painel que exibe o título da página atual e um botão com uma seta para a esquerda, que retorna o usuário à última página acessada na seção atual, permitindo sempre retornar de qualquer parte do sistema até o Menu Principal. Tocar no botão 'Voltar' nativo do dispositivo também tem a mesma função.

Fig. 14 Menu "Meu Perfil", com Painel Superior

O botão "Editar Perfil" leva o usuário novamente à tela de criação de novo perfil (Fig. 12), exceto que desta vez os dados anteriormente registrados no perfil já preenchem os campos da página, e o painel superior está disponível, permitindo sair sem salvar. O botão "Informações de Contato" abre a lista de formas de contato(Fig. 15). Nesta página o usuário pode registrar diferentes formas que outras pessoas podem usar para entrar em contato com ele, ou remover informações inseridas anteriormente que não se deseja mais compartilhar com os usuários do aplicativo, ao tocar no botão com ícone de lixeira adjacente à forma de contato que se deseja remover. Para adicionar uma entrada, o botão na parte inferior da página, "Adicionar forma de contato", abre um painel (Fig. 16) onde o usuário pode inserir os dados de contato e o contexto de que tipo de informação de contato é essa.

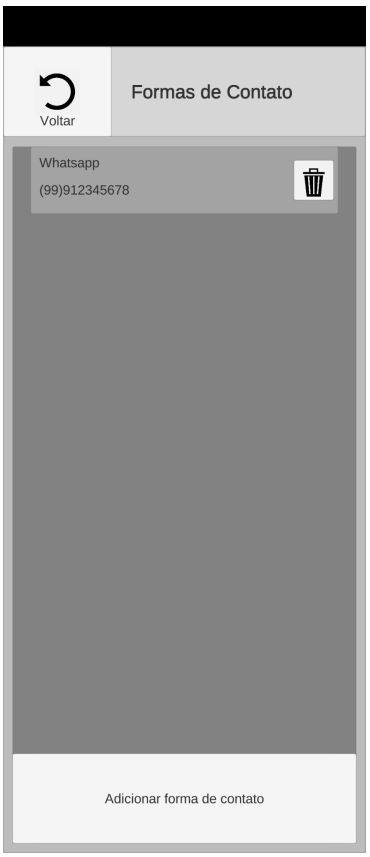

Fig. 15 Lista de Formas de Contato do usuário atual (contendo 1 Contato)

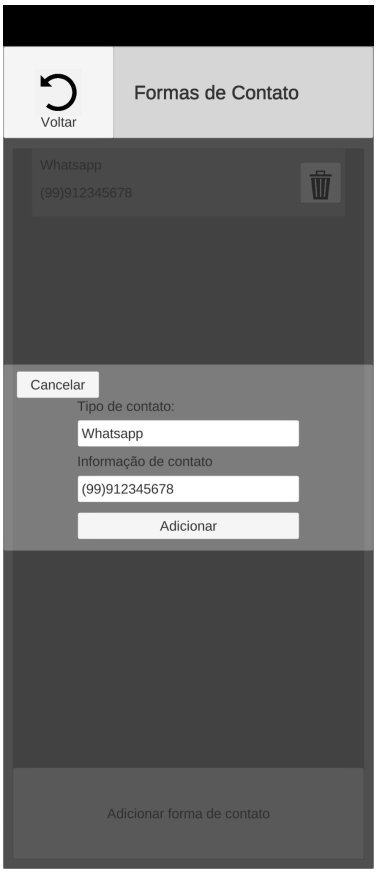

Fig. 16 Painel para adicionar nova Informação de Contato

O botão "Comissões e Serviços Ativos", novamente no menu Perfil, traz a lista dos Contratos que foram criados pelo usuário atual (Fig. 17), independente da classe do contrato. O usuário, sendo o autor destes contratos, pode removê-los também com o botão de lixeira associado.

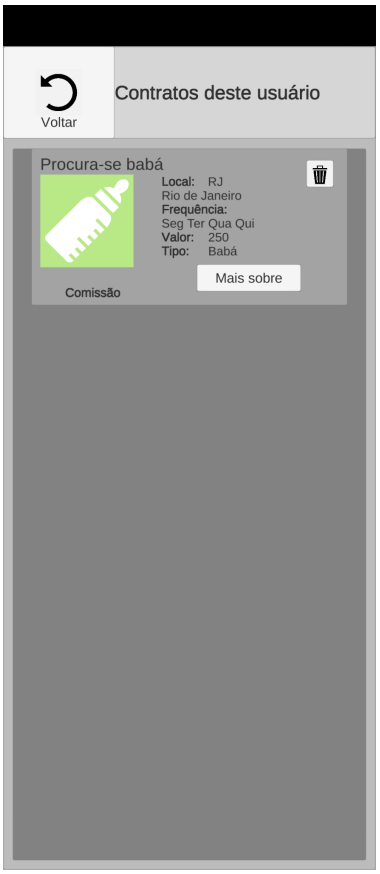

Fig. 17 Lista de Comissões e Serviços Ativos do usuário (contendo 1 Contrato)

O último botão no perfil, rotulado "Avaliações", carrega a lista dos usuários que avaliaram os serviços deste usuário (Fig. 18) e as respectivas notas atribuídas, em número de estrelas variando de 1 a 5. Se estiver acessando a lista de Avaliações de outra pessoa, uma barra deslizante torna-se disponível na parte inferior desta tela, permitindo ao usuário deixar sua própria avaliação. Caso o usuário já tenha avaliado esta pessoa e publique uma nova avaliação, o valor da anterior é substituído pelo novo.

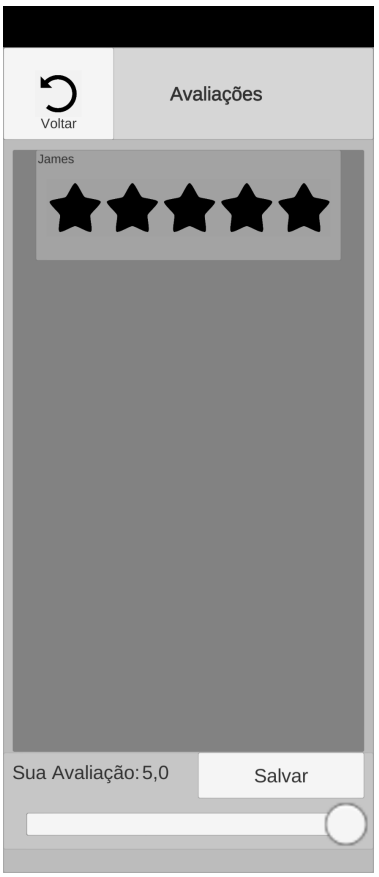

Fig. 18 Lista de Avaliações (contendo 1 Avaliação) e painel de Nova Avaliação

Voltando ao Menu Principal, os botões de rótulos "Quero Trabalhar" e "Quero um Profissional" ambos exibem a lista de contratos (Fig. 19) ao serem tocados. A diferença entre os botões é que "Quero Trabalhar" filtra por contratos que são da classe "Comissão", indicando que seu autor busca um trabalhador que presta o serviço descrito. Por sua vez, o botão "Quero um Profissional" disponibiliza apenas a lista dos contratos da classe "Serviço", indicando que seu autor está descrevendo o trabalho que ele oferece (Acredita-se que usar esses rótulos nos botões encaminhe para que esteja claro para o usuário qual classe de contrato está sendo exibida, enquanto os termos "Comissão" e "Serviço" usados no sistema poderiam causar confusão). Exceto por esses filtros e pela diferença no título exibido da página, a tela acessada pelos dois botões é idêntica.

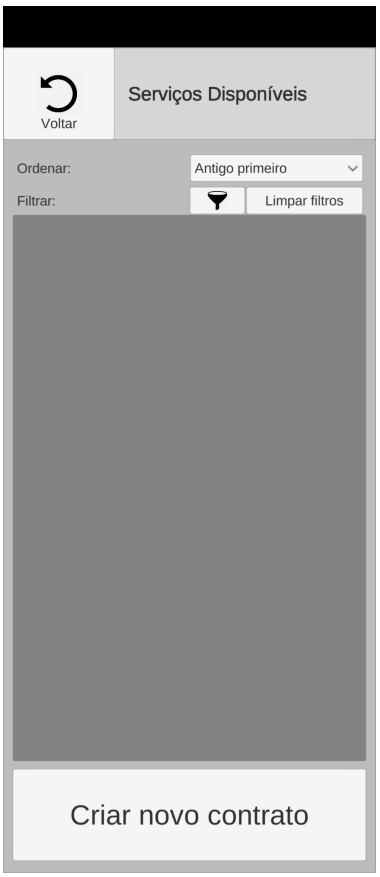

Fig. 19 Menu de lista de contratos da classe Serviço

No alto da tela, um menu de *Dropdown* permite mudar a ordem dos dados exibidos, para que sejam listados em ordem crescente ou decrescente de data de criação ou de preço do contrato. Abaixo do *dropdown*, um botão com ícone de funil dá acesso à tela de Filtros(Fig. 20), que permite determinar parâmetros para filtrar os dados exibidos, enquanto o botão ao lado reinicia os parâmetros para os valores padrão do sistema.

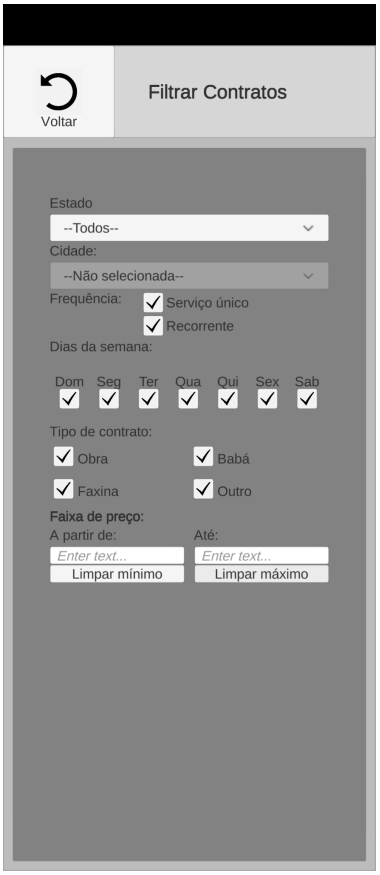

Fig. 20 Menu de Filtros de contratos

Abaixo do painel que exibe a lista de contratos (Fig. 19), o botão "Criar novo contrato" abre o menu de criação de novo Contrato (Fig. 21), onde o usuário pode preencher todos os dados do contrato que deseja publicar. Assim como na edição de perfil, é possível transferir um arquivo de imagem '.png' do dispositivo para representar o contrato. Ao final do formulário, o botão "Publicar Contrato" envia os dados preenchidos ao servidor para ser exibido com os demais quando apropriado.

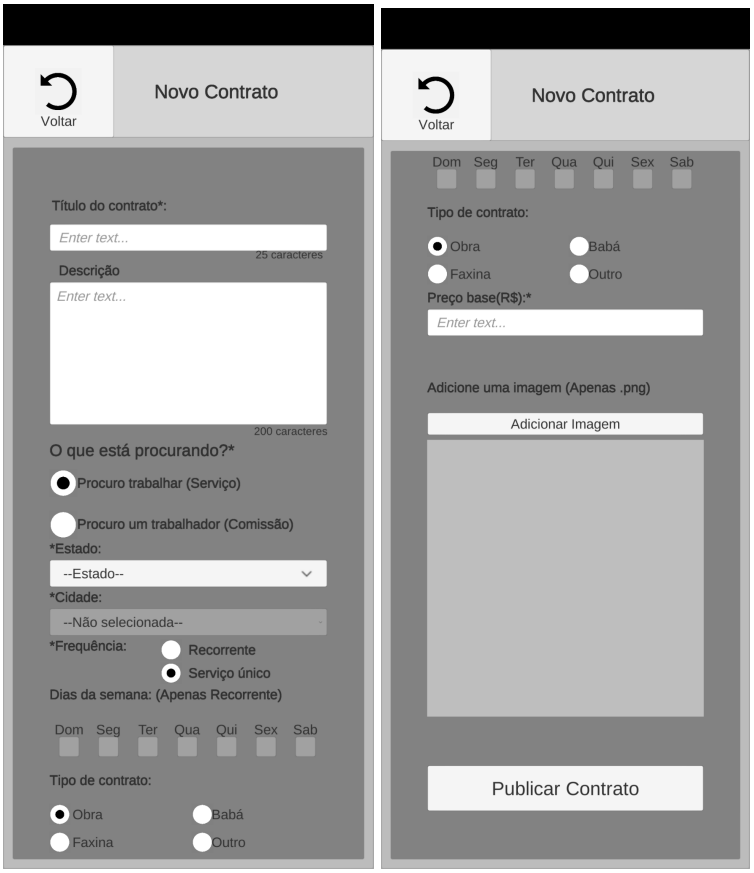

Fig. 21 Parte superior (à esquerda) e inferior (à direita) do Formulário de Novo Contrato

Sempre que um contrato é exibido numa lista, um botão "Mais sobre" acoplado permite acessar a página de Informações do Contrato (Fig. 22), onde além dos dados informados originalmente na lista, são adicionados o nome do autor e a descrição daquele contrato, inserida no formulário de sua criação, e são disponibilizados botões para abrir a sala de conversa com o autor(Fig. 23) e para acessar o seu perfil. O menu do perfil de outros usuários é muito similar ao do perfil do usuário atual, exceto que os botões "Editar Perfil" e "Adicionar forma de contato" são substituídos por botões para ir para a sala de conversa com o usuário. Enquanto se acessa o perfil de outro usuário, também não há opção para excluir seus contratos nem suas formas de contato, e a opção de adicionar avaliação se torna disponível.

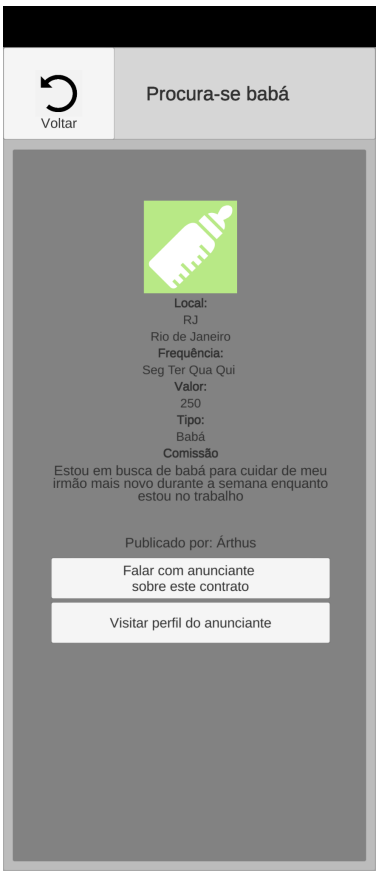

Fig. 22 Painel de Informações adicionais de contrato

Fig. 23 Perfil de outro usuário

Fig. 24 Sala de conversa (com mensagens)

Se em algum momento um usuário envia uma mensagem para outro em uma sala de conversa, essa sala fica disponível para ser acessada novamente por ambos os usuários, através da lista de conversas (Fig. 24).

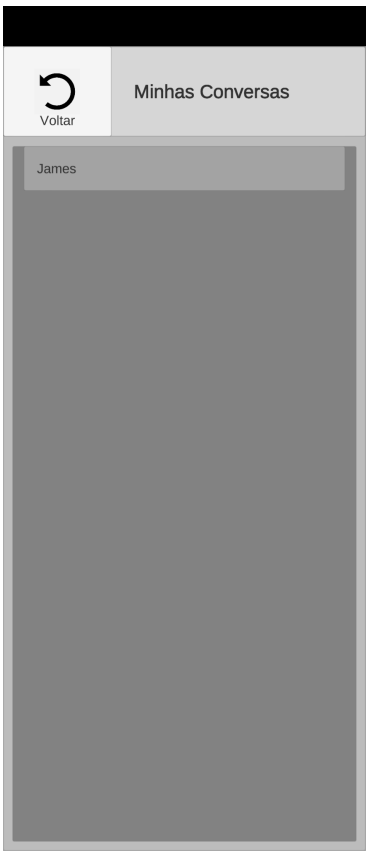

Fig. 25 Lista de conversas (com uma conversa iniciada)

6 Implementação e avaliação

6.1 Planejamento e execução de testes funcionais

Ao longo do desenvolvimento das diversas funcionalidades do aplicativo, conforme cada novo sistema era implementado, eram realizados alguns testes gerais do sistema para garantir seu funcionamento. Como o uso do aplicativo é relativamente linear, testar a implementação de uma nova capacidade do sistema significava testar também todas as etapas anteriores do fluxo que o usuário percorre até a novidade, o que acabava incluindo quase toda a aplicação: Testar a exibição de elementos na lista de conversas requer que uma mensagem seja enviada, que requer que o usuário acesse o perfil de outra pessoa, o que por sua vez requer que essa pessoa tenha publicado um contrato e criado um perfil, e todas as funcionalidades necessitam que uma conta tenha sido registrada para acessar o sistema. Fora deste fluxo geral de uso, resta apenas o teste das funcionalidades de edição de perfil existentes, e da listagem e criação de avaliações e de formas de contato do usuário.

6.2 Planejamento e execução de outros testes

Para avaliar a qualidade da interface desenvolvida, os testes iniciais foram realizados com 6 usuários, onde foi requisitado que eles se registrassem no sistema, criassem um perfil, criassem um contrato e entrassem em contato com outro usuário para conversar sobre o contrato criado. Nesses testes, observou-se que, para alguns dispositivos, a câmera frontal do celular invade a região da tela, ficando bem em cima do botão para navegar para a página anterior. Para resolver isso, o painel do título teve sua altura aumentada, para garantir que o botão seja visível em qualquer aparelho. Houve casos em que os usuários tiveram dificuldades em encontrar o botão "Criar novo contrato", que também teve sua altura incrementada para maior visibilidade. Ambas soluções se mostraram suficientes em testes subsequentes.

Durante o registro, alguns usuários recebiam a mensagem de endereço de email inválido apesar de usarem seus emails pessoais. Ao analisar a situação, percebeu-se que isso ocorria devido ao fato destes usuários usarem a funcionalidade do teclado do dispositivo de completar automaticamente o texto sendo inserido, o que adicionava um caractere de espaço ao final do endereço digitado. Foi então adicionado o tratamento de se aplicar a função C# String.Trim() ao endereço fornecido, para garantir que não haja espaços vazios antes ou depois do texto inserido que possam invalidar o email.

Por fim, em versões anteriores, o sistema não era capaz de transferir imagens do dispositivo para o banco de dados, alternativamente optando por uma transferência de imagem por URL do arquivo desejado. Os testes, porém, revelaram que o usuário médio do sistema não sabe obter uma URL de imagem na *web*. Isso foi resolvido com a nova versão permitindo a transferência de qualquer imagem '.png' do dispositivo para a nuvem.

Não foram realizados testes sobre a capacidade máxima de armazenamento e de stress do servidor, pois este é gerenciado pelo Google Firebase e deliberadamente limitado pelo plano de inscrição no serviço. O plano básico utilizado, gratuito, permite o acesso simultâneo de até 100 usuários, o armazenamento de até 5GB no repositório de imagens e o envio total de até 1GB de imagens num dia, que foi dado como suficiente para o sistema, e podendo ser escalado para uma capacidade maior com outros planos de assinatura no mesmo serviço conforme necessário.

6.3 Comentários sobre a implementação

A maior desvantagem do uso da plataforma Unity para o desenvolvimento de aplicações não associadas a jogos é o excesso de

bibliotecas disponíveis para operações que não são relevantes para esse tipo de projeto, como cálculos de física e computação gráfica. Isso pode resultar num tamanho inflado para os arquivos executáveis gerados e em uma perda de performance caso o desenvolvedor não configure corretamente o ambiente para desabilitar os processos não relevantes para o projeto. Contudo, nesta aplicação, o tamanho final do arquivo executável produzido e o tempo de espera de suas operações acabaram não distantes dos padrões estabelecidos pelos concorrentes estudados no mercado, ocupando apenas aproximadamente 50MB de memória e raramente demorando mais que 1 segundo para trazer os resultados de qualquer operação, a depender principalmente da velocidade de conexão do usuário, tempo considerado perfeitamente razoável dado que nenhuma operação do sistema demanda uma resposta mais imediata.

Algumas funções de objetos GameObject, do editor Unity, são limitadas a serem executáveis apenas na *thread* principal, o que entrou em conflito com a necessidade de muitas das funcionalidades do Firebase de serem executadas em corrotinas para não paralisar o sistema durante acessos ao servidor. A solução foi desenvolver o módulo Singleton UnityMainThread para ser o responsável por preparar em fila e executar na *thread* principal todas as funções que tem esse requisito.

Sendo o Firebase um sistema de banco de dados NoSQL, os dados armazenados no JSON que compõem o servidor são acessados por seus respectivos endereços, que por padrão não aceitam alguns caracteres como valores válidos para criar subpastas. Dentre estes caracteres está incluso o ponto (.), que é um requisito para considerar um email como válido. Como o email dos usuários estava sendo utilizado como seu identificador, isso gerou problemas para organizar dados que possuíam múltiplos elementos associados a um mesmo usuário. Uma substituição seria necessária, porém o novo caractere deve ser escolhido cuidadosamente para permitir que o email original seja restaurado sem ambiguidade. Portanto, o caractere de aspas duplas ("), que é válido como parte de endereço e inválido como parte de email, foi utilizado para substituir os dados quando necessário.

7 Considerações finais

O foco deste trabalho, desde o início, foi disponibilizar uma plataforma em que trabalhadores de todo o Brasil possam compartilhar com as pessoas ao seu redor uma amostra do e o que estão dispostos a fazer para contribuir para a

sociedade. A prioridade do sistema era facilitar essa divulgação desses indivíduos, e a forma como foi implementado resolve essa necessidade básica.

Apesar de já ter desenvolvido jogos em Unity em projetos passados, tanto para celulares Android quanto para Desktop, não tinha usado a plataforma para desenvolver outros tipos de aplicação. Foi um ambiente de desenvolvimento incomum, mas não inadequado: todas as ferramentas que normalmente servem para construir a interface do usuário nos jogos são úteis também para a interface de outros tipos de aplicação, e o editor do Unity permite acompanhar e testar o que foi construído imediatamente, em várias opções de resolução de tela.

Usar o Google Firebase e, num geral, desenvolver um aplicativo inteiro que acessa um banco de dados, também foi uma novidade. As bibliotecas da plataforma trivializam uma boa parte das dificuldades que eu teria encontrado caso fosse necessário estabelecer pessoalmente o servidor do aplicativo, além de disponibilizar o acesso a qualquer momento, já que esse servidor normalmente não é desligado.

Se fosse iniciar o projeto novamente com o conhecimento adquirido, eu teria optado por criar mais versões de teste do aplicativo no android ao longo do desenvolvimento, ao invés de confiar tanto nos testes realizados dentro do editor Unity. Isso teria poupado tempo perdido reconstruindo partes do sistema que funcionaram perfeitamente dentro do Unity, mas apresentaram problemas quando testadas no dispositivo Android.

Atender o escopo inicial não significa que o aplicativo não pode ser melhorado. A implementação deixa a desejar em alguns aspectos que deixo como recomendação para o caso de alunos futuros optarem por expandir o projeto:

Enquanto a segurança dos dados disponibilizados no banco de dados é encriptada pela tecnologia do próprio Google Firebase, não há mecanismos para proteger o sistema de ataques de *spam* de *bots*, que podem criar contas em massa para publicar repetidas vezes um mesmo contrato ou uma mesma avaliação, ou mesmo simplesmente ocupar todas as vagas de acesso simultâneo que o servidor suporta, tornando a aplicação completamente inacessível. Seria adequado implementar meios de evitar tais eventos, como por exemplo verificação do email de registro antes de conceder acesso ao sistema, testes *captcha*, e a possibilidade de denunciar usuários, a qual é recomendada mesmo que não existam contas *bot* no sistema, para controlar usuários que podem usar o sistema para realizar atividades ilícitas ou indesejáveis.

Quanto à experiência do usuário, avaliações publicadas não possuem nenhuma descrição que contextualiza o porquê da nota atribuída e a média das avaliações de uma pessoa não é exibida no aplicativo, devendo ser calculada pelo usuário manualmente ao ver a lista. Não é possível na versão atual acessar o perfil de um usuário a partir da conversa com ele ou de uma avaliação que ele publicou, sendo necessário encontrar um contrato postado pela pessoa para ver seu perfil, tornando a navegação inconveniente. Como o Brasil é um país muito extenso, a seleção de estados e cidades no momento é incômoda, sendo necessário encontrar manualmente a cidade procurada por ordem alfabética. Recomenda-se a implementação de uma caixa de busca para esse menu em versões futuras do aplicativo.

É interessante também tornar possível enviar outros tipos de conteúdo pelo sistema de mensagens, como por exemplo imagens, *links*, endereços no mapa, páginas de contratos e usuários no sistema, e formas de contato externas. Também seria conveniente poder compartilhar *links* de contratos por outros canais de comunicação, de forma que, quando acessados, direcionassem o usuário para a página do contrato. O recebimento de mensagens atualmente não cria nenhum tipo de notificação, sendo necessário que o usuário visite a tela de conversas para ver que há novas mensagens. É desejável também que eventualmente seja possível publicar múltiplas imagens por contrato, e que se permita ampliar as imagens na tela.

8.Referências

[1] **Aplicativo Uber**. Disponível em <https://play.google.com/store/apps/details?id=com.ubercab> Acesso em 09/04/2021.

[2] **Aplicativo UberEats.** Disponível em: <https://play.google.com/store/apps/details?id=com.ubercab.eats>. Acesso em 09/04/2021.

[3] **Aplicativo Craigslist.** Disponível em: <https://play.google.com/store/apps/details?id=org.craigslist.CraigslistMobile&hl=pt_BR&gl=US>. Acesso em 09/04/2021.

[4] **Aplicativo Craigslist web.** Disponível em: <https://rio.craigslist.org/>. Acesso em 09/04/2021.

[5] **Aplicativo SOS Costura.** Disponível em: <https://play.google.com/store/apps/details?id=com.soscostura.soscosturaclientemobile>. Acesso em 09/04/2021.

[6] **Aplicativo Sitly.** Disponível em: <https://play.google.com/store/apps/details?id=com.sitly.app>. Acesso em 09/04/2021.

[7] **Aplicativo Parafuzo.** Disponível em: <https://play.google.com/store/apps/details?id=com.parafuzo>. Acesso em 09/04/2021.

[8] **Aplicativo Mãos à Obra.** Disponível em: <https://play.google.com/store/apps/details?id=app.maosaobra.maosaobraapp2>. Acesso em 09/04/2021.

[9] **Unity Real Time Development Platform.** Disponível em: <https://unity.com/>. Acesso em 05/05/2022

[10] **Google Firebase.** Disponível em <https://firebase.google.com/?hl=pt-br>. Acesso em 05/05/2022

[11] **Prisma Game Lab.** Participação no desenvolvimento dos projetos 'Keimagus', 'Por trás da máscara', 'O formigueiro', 'Vish, Vice' e 'Sweet Punch'. Disponível em <https://prismagamelab.itch.io>.

[12] Google Firebase. **Documentação do Firebase.** Disponível em: <https://firebase.google.com/docs/>

[13] Google Firebase. **Documentação do Firebase.Auth** Disponível em: <https://firebase.google.com/docs/reference/unity/class/firebase/auth/firebase-auth>

[14] Google Firebase. **Documentação do Firebase.Database.DatabaseReference.** Disponível em: <https://firebase.google.com/docs/reference/unity/class/firebase/database/database-reference>

[15] **Documentação: Firebase.Storage.FirebaseStorage.** Disponível em: https://firebase.google.com/docs/reference/unity/class/firebase/storage/firebase-storage

[16] **Rede SUAS.** Lista de Municípios Brasileiros e Informações Adicionais. Disponível em: http://blog.mds.gov.br/redesuas/lista-de-municipios-brasileiros/