Recycler: Um Aplicativo Móvel Colaborativo para Reciclagem de Materiais

Gabriel de Lira Ferreira[1], Fernando Sambinelli[1]

Curso Superior de Tecnologia em Análise e Desenvolvimento de Sistemas – Instituto Federal de Educação, Ciência e Tecnologia de São Paulo (IFSP) – Campus Hortolândia – São Paulo – SP – Brasil

gliraferr@gmail.com, sambinelli@ifsp.edu.br

Abstract. *Currently, in Brazil, there are two ways to discard a recyclable material: leave it in a place that someone can get it or take it to a specific delivery point. This work aimed to develop an application for mobile devices that is an alternative to this problem. It can connect people interested in discarding recyclable materials and people who want to collect these materials, bringing benefits to both parties.*

Resumo*. Atualmente, no Brasil, há duas formas para uma pessoa física descartar um material reciclável: deixá-lo em algum local público para que posteriormente seja coletado ou levá-lo a um ponto de entrega voluntária. Este trabalho teve como objetivo o desenvolvimento de um aplicativo para dispositivos móveis que seja uma alternativa a esse problema, conectando pessoas interessadas em descartar materiais recicláveis e pessoas que queiram coletar esses materiais, trazendo benefícios para ambas as partes.*

1. Introdução

Nos últimos anos, o volume de resíduos sólidos urbanos no Brasil tem aumentado, deixando o Brasil como quarto maior gerador de resíduos no mundo segundo uma pesquisa realizada pela [Abrelpe 2016]. Uma maneira de tratar isso é através da coleta seletiva de materiais recicláveis. Segundo o [Ministério do Meio Ambiente 2015], no Brasil há duas formas mais comuns de coleta seletiva: Coleta porta a porta, onde caminhões ou coletores passam nos domicílios e comércios recolhendo os materiais recicláveis; Coleta por pontos de entrega voluntária, que ficam localizados próximos aos conjuntos de residências ou instituições para a entrega dos materiais. Embora a coleta seja benéfica para o tratamento do volume de resíduos sólidos, nem todos os brasileiros possuem acesso a um serviço de coleta seletiva. Segundo a pesquisa Ciclosoft [Cempre 2015], apenas 18% das cidades brasileiras possuem algum programa de coleta seletiva.

Uma pesquisa feita pelo IBGE [Bôas 2016] mostra que o celular se tornou a principal forma de acesso à internet nos lares brasileiros. Outro estudo feito pela Fundação Getúlio Vargas de São Paulo (FGV-SP) mostra que a quantidade de celulares no Brasil já chega a 168 milhões, como mostra a [Folha de São Paulo 2016]. Dentre os diversos dispositivos móveis no mercado, existe uma variedade de sistemas operacionais, sendo os mais populares,

segundo [Woods e Meulen 2016], o *Android*, presente em 78,8% dos dispositivos e o *iOS* com 17,9%.

Considerando o crescimento do acesso à internet por parte dos brasileiros, em especial por meio dos dispositivos móveis a diversidade de seus sistemas operacionais, este trabalho teve como objetivo desenvolver um aplicativo colaborativo para dispositivos móveis, usando tecnologias que pudessem alcançar a maor parte dos sistemas operacionais móveis, voltado para as pessoas interessadas em coleta seletiva. A proposta do aplicativo é facilitar o trabalho dos coletores de recicláveis que fazem coleta porta a porta possibilitando-lhes combinarem coletas em locais específicos de materiais recicláveis disponibilizados por outros usuários.

2. Trabalhos Correlatos

Esta seção apresenta as soluções existentes que possuem relação com o tema deste trabalho.

2.1 Aplicativo Cataki

O *Cataki* é um aplicativo para dispositivos móveis com uma proposta bem parecida com a do *Recycler*: Conectar coletores de materiais recicláveis a pessoas que querem descartá-los. O aplicativo foi criado pelo projeto "Pimp My Carroça" [1], que é descrito em seu site oficial como um movimento que luta para tirar os catadores de materiais recicláveis da invisibilidade, promover a sua auto estima e sensibilizar a sociedade para a causa em questão com ações criativas que utilizam a arte do Grafite para conscientizar, engajar e transformar a vida desses indivíduos.

No aplicativo é possível encontrar coletores de materiais recicláveis ou cooperativas próximas através de um mapa onde cada coletor é representado pelo ícone de uma carroça, como mostra a figura 1.

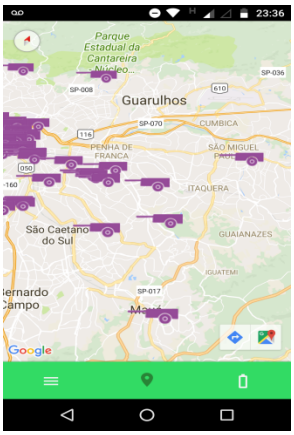

Figura 1. Tela inicial do aplicativo *Cataki*. Fonte: Captura de tela realizada pelo autor.

[11]http://pimpmycarroca.com/

No mapa é possível obter detalhes do coletor ao clicar no ponto que o representa. Na tela de detalhes é exibida uma foto do coletor, seu nome, alcunha (apelido), telefone, endereço onde reside, os bairros em que realiza coletas, os tipos de materiais que coleta e a sua história de vida. Para realizar uma solicitação de coleta é necessário entrar em contato com o coletor através de mensagens instantâneas em um aplicativo externo ou através de uma ligação para o seu número de telefone.

2.2 Projeto *Relix*

Criado no estado de Alagoas, em parceria com o SESI, o projeto *Relix* promove ações voltadas para sustentabilidade através de apresentações teatrais em locais como escolas, empresas e teatros. O projeto também já distribuiu 100 lixeiras para coleta seletiva e 30 bicicletas que auxiliam catadores na coleta [Carvalho 2014].

Associado ao projeto existe um aplicativo para dispositivos móveis *Android* e *iOS*, que permite a localização de pontos de coleta seletiva e de cooperativas de catadores em Pernambuco e Alagoas, como mostra a figura 2.

Figura 2. Tela inicial do aplicativo *Relix*. Fonte: Imagem retirada da página do aplicativo na Play Store.

No aplicativo, além da funcionalidade para localizar pontos de coleta e cooperativas, há uma funcionalidade para solicitar coleta em domicílio. Porém, quando essa opção é acessada, é exibida uma lista com informações para contato com cooperativas de reciclagem,

a fim de que a solicitação seja feita através de uma ligação telefônica para os profissionais que executam a coleta, conforme mostra a figura 3.

Figura 3. Tela com a lista de cooperativas. Fonte: Imagem retirada da página do aplicativo[2] na Play Store.

3. Referencial Teórico

Nesta seção são descritos os principais conceitos que fundamentaram os aspectos técnicos e de negócio aplicados durante o desenvolvimento deste trabalho.

3.1 Economia Compartilhada

Tendo origem nos Estados Unidos na década de 1990, a economia compartilhada é um conceito que vem ganhando cada vez mais força nos dias atuais. Segundo [Pinelli 2017], a tecnologia está ajudando a economia compartilhada a crescer. A ideia principal é o compartilhamento de bens ou serviços eliminando o desperdício e contribuindo com a sustentabilidade. A economia compartilhada leva as pessoas a possuírem menos e compartilharem mais.

Esse conceito trouxe uma formalização para relações já existentes. Por exemplo, antes sempre que surgisse a necessidade de realizar alguma manutenção dentro de casa as pessoas chamavam profissionais que elas conheciam ou ouviram falar. Atualmente, soluções como o

[2]Site oficial: http://projetorelix.com.br/

Task Rabbit[3] permitem que pessoas escolham, a partir de avaliações do público e reputação, sem necessariamente conhecê-los [Pinelli 2017].

Um dos exemplos mais comuns de economia compartilhada é o *Uber* [Moretzsohn 2017]. É uma plataforma que oferece serviço de mobilidade através dos motoristas disponibilizados e providos de seus próprios veículos para trafegar. Além do *Uber*, também existem o *Airbnb* [Moretzsohn 2017], e, o já citado *Task Rabbit*, como exemplos claros de economia compartilhada. O *AIrbnb* possibilita a hospedagem de pessoas nas casas de outras pessoas. Já o *Task Rabbit* é uma plataforma destinada àquelas mais possuidoras de dinheiro que tempo, pois é possível contratar todo tipo de serviço através do programa.

3.2 Histórias de Usuário

Histórias de usuário é uma forma de expressar os requisitos de um *software* em *Extreme Programming*, uma metodologia ágil para o desenvolvimento de software [Sommerville 2011]. Histórias de usuário são descrições curtas e simples de uma funcionalidade do sistema vista pela perspectiva de quem a está pedindo (normalmente um usuário ou um cliente) [Cohn 2004].

Os objetivos de uma história de usuário são: focar no problema de negócio que precisa ser resolvido, não na solução; começar uma conversa sobre o porquê do problema carecer de resoluções e saber quem deu resolução [Levison 2017]. A figura 4 mostra um exemplo de uma história de usuário.

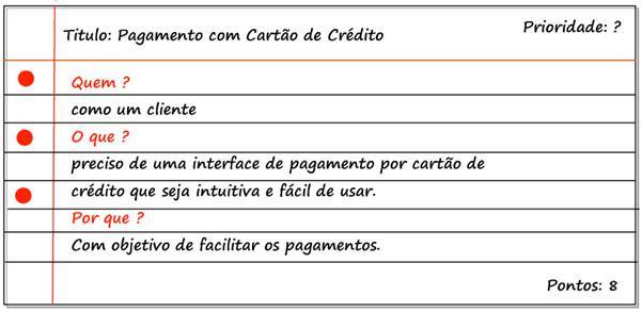

Figura 4. Exemplo de uma história de usuário [Levison 2017].

3.3 Aplicativos Híbridos

Atualmente, existem diversos *smartphones* no mercado e, portanto, diversos sistemas operacionais em uso. Para cada sistema operacional existe uma linguagem de programação para ser usada no desenvolvimento de aplicativos - no *Android*, por exemplo, os aplicativos são feitos em *Java* ou *Kotlin* e no *iOS* é usada a linguagem *Swift* [Austins 2017]. Os aplicativos codificados pela linguagem do sistema operacional são chamados Nativos e

[32] https://www.taskrabbit.com/

possuem acesso a recursos intrínsecos do dispositivo como câmera, geolocalização e arquivos.

Além dos aplicativos nativos, existe atualmente outra opção arquitetural para o desenvolvimento que são os aplicativos híbridos ou *mobile* híbrido. Um aplicativo híbrido trabalha com o conceito de *Web Apps*, que são aplicativos desenvolvidos com tecnologias para a Web como HTML, CSS e *JavaScript* e rodam no navegador do dispositivo [Viswanathan 2017], porém não possuem acesso a recursos específicos do equipamento.

Unindo os conceitos supracitados, os aplicativos híbridos são desenvolvidos com tecnologias para a Web e, no entanto, possuem acesso aos recursos nativos do dispositivo como câmera e geolocalização. Esses aplicativos são escritos em HTML, CSS e *JavaScript* e depois compilados para sistemas operacionais de dispositivos móveis como o *Android* e o *iOS*. Assim o aplicativo é escrito apenas uma vez e funciona da mesma forma, com os mesmos recursos em diferentes sistemas operacionais [Cheng 2017]. Isso facilita o trabalho do desenvolvedor, pois não é necessária a sapiência de todas as linguagens de diferentes sistemas operacionais e/ou a necessidade de escrever várias vezes a mesma lógica para alcançar a maioria do mercado de *smartphones*. Alguns exemplos de ferramentas utilizadas no mercado com esse objetivo são o *Cordova*[4] (da *Apache Software Foundation)* e o *PhoneGap*[5] (produzido pela Adobe).

3.4 Web Services

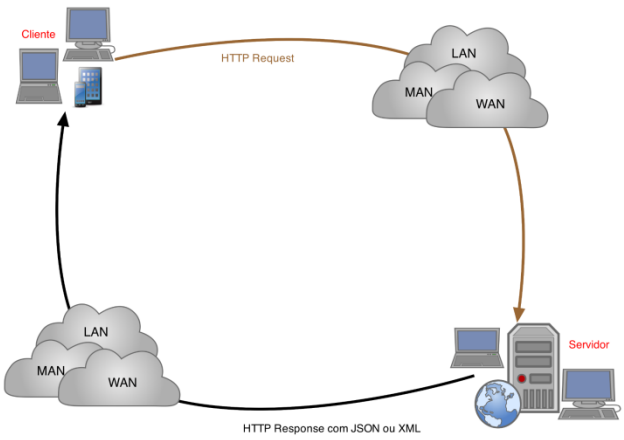

Figura 5. Funcionamento de um *web service* REST [Carneiro 2013].

Com o crescimento dos sistemas de informação surge a necessidade da troca de dados entre diferentes sistemas. Essa troca de dados ocorre por meio de *web services*. Segundo [Carneiro 2013], *web service* é uma arquitetura de comunicação entre *softwares* que independem da

[4] http://cordova.apache.org/
[5] https://phonegap.com/

plataforma na qual foram desenvolvidos. Essa comunicação é feita por meio de chamadas de serviços feitas usando algum protocolo web, que trafegam arquivos XML ou JSON. Atualmente existem dois tipos de arquiteturas de *web services* comumente utilizados no mercado: SOAP (*Simple Object Data Protocol*) e REST (*Representational State Transfer*). A figura 5 ilustra o funcionamento de um *web service* REST.

3.5 Bancos de Dados NoSQL

Bancos de dados relacionais têm dominado a indústria de *software* por um longo período de tempo, provendo mecanismos para persistir dados estruturados, controlar concorrência e transações [Sadalage 2014]. Um banco de dados relacional é um conjunto de tabelas montadas em categorias predefinidas onde cada tabela possui uma ou mais categorias de dados em colunas [Leavitt 2010]. Exemplos de bancos de dados relacionais são: *Oracle*, feito pela empresa que leva o mesmo nome; MySQL, também feito pela *Oracle*; *PostgreSQL*, banco de dados de código aberto feito pela comunidade de *software livre*. Embora os bancos de dados relacionais sejam amplamente utilizados no mercado, eles possuem algumas limitações [Leavitt 2010]:

- Escalabilidade: para obter escalabilidade é necessário distribuir o banco de dados em vários sistemas. Porém, os bancos de dados relacionais não trabalham de uma maneira simples com sistemas distribuídos, pois não foram projetados para o particionamento de dados;
- Complexidade: todos os dados devem ser estruturados em tabelas e, dependendo do formato dos dados, estruturar os dados pode ser um trabalho complexo, difícil e lento de se fazer.

Dadas as limitações que os bancos de dados relacionais possuem, o movimento dos bancos de dados NoSQL ganha cada vez mais força na indústria de *software*. Algumas características, descritas por [Strauch 2011], dos bancos de dados NoSQL, também conhecidos como *bancos de dados não relacionais* são: evitam complexidade desnecessária, alta taxa de transferência de dados, escalabilidade e evitam o caro mapeamento objeto-relacional.

Os bancos de dados não relacionais são classificados em quatro categorias [Sadalage 2014]:

- Chave-valor: guardam um valor associado a uma chave que o identifica. Exemplos de bancos de dados chave-valor são o *Redis* e o *Memcached*;
- Orientados a documentos: armazenam -como o próprio nome sugere - documentos que podem ser arquivos XML, JSON, entre outros. Alguns exemplos de bancos de dados orientados a documentos são o *MongoDB* e o *CouchDB*.
- Orientados à coluna: diferem dos bancos tradicionais, pois não armazenam os dados em linhas, mas em colunas. Com os dados armazenados em colunas é possível obter um ganho de performance nas consultas por informações do mesmo tipo, como os

nomes dos usuários. Exemplos são o *Cassandra*, do *Facebook*, e o *Amazon DynamoDB*.

- Grafo: permite que sejam armazenadas entidades e o relacionamento entre essas entidades. Entidades são conhecidas como nós e os relacionamentos são conhecidos como as arestas. Exemplos comuns são o *Neo4J* e o *Infinite Graph*.

4. Metodologia

A princípio, houve uma elaboração detalhada da ideia principal identificando o problema e buscando soluções através de reuniões de *brainstorm*, com alguns discentes e docentes do Instituto Federal de São Paulo Campus Hortolândia. Nessas reuniões, os participantes expressavam suas ideias sobre o que deveria ser feito, o que era mais importante no contexto de uso da economia compartilhada para a coleta seletiva e o tipo de solução a ser feita.

Desta forma, foi tomada a decisão de criar um aplicativo para dispositivos móveis e, após análises de estatísticas de uso dos sistemas operacionais nesses dispositivos, foi escolhida uma solução de aplicativos híbridos que alcançasse o maior público possível de usuários de *Android* e *iOS*. Com a ideia consolidada (e o público alvo definido), foi feita a extração de requisitos do aplicativo usando histórias de usuário e prototipagem. Assim que os requisitos foram elucidados, houve uma divisão das histórias de usuário em funcionalidades. Tendo essas funcionalidades definidas, fez-se um desenho da solução técnica para definir quais tecnologias seriam utilizadas e um desenho da visão geral da arquitetura.

Para a solução técnica foi decidido usar o *Ionic Framework* objetivando a implementação do aplicativo, o *Spring Framework* trabalhando na criação do *Web Service* e o *MongoDB* sendo utilizado como banco de dados. Com a solução técnica definida, foi implementado, primeiramente, o *Web Service* e em seguida o aplicativo que passou por algumas evoluções durante o desenvolvimento.

5. Desenvolvimento

Esta seção apresenta fases do desenvolvimento que iniciou-se pelo entendimento do fluxo de negócio do aplicativo, passando pelos requisitos e o desenho da solução técnica, até a implementação do aplicativo *Recycler* e do *Web Service*.

5.1 Entendimento do Fluxo do Aplicativo

Primeiro, foi definido o fluxo de negócio do aplicativo, ou seja, como e por quem o aplicativo seria usado. A figura 6 ilustra este fluxo.

O fluxo começa no passo 1, quando um usuário deseja descartar algum material reciclável. Através do aplicativo, ele solicita uma coleta informando uma data, um intervalo de horário para coleta, a descrição do material e o endereço onde o mesmo se encontra. Essa solicitação é enviada para um servidor ficando disponível para outros usuários consultarem. O coletor de material reciclável, por sua vez, busca solicitações de coleta no aplicativo, como mostra o passo 2. Após escolher uma solicitação, o coletor informa sobre seu interesse na

solicitação, onde está ilustrado no passo 3. No passo 4, o autor da solicitação recebe a notificação e aceita ou não a coleta em seu domicílio. Após a confirmação, o coletor passa no dia e horário combinados na solicitação para coletar o material, como pode se ver no passo 5. Sendo realizada a coleta, o usuário avalia o coletor; e, através dessas avaliações, os coletores ganham reputação no aplicativo, como mostra passo 6.

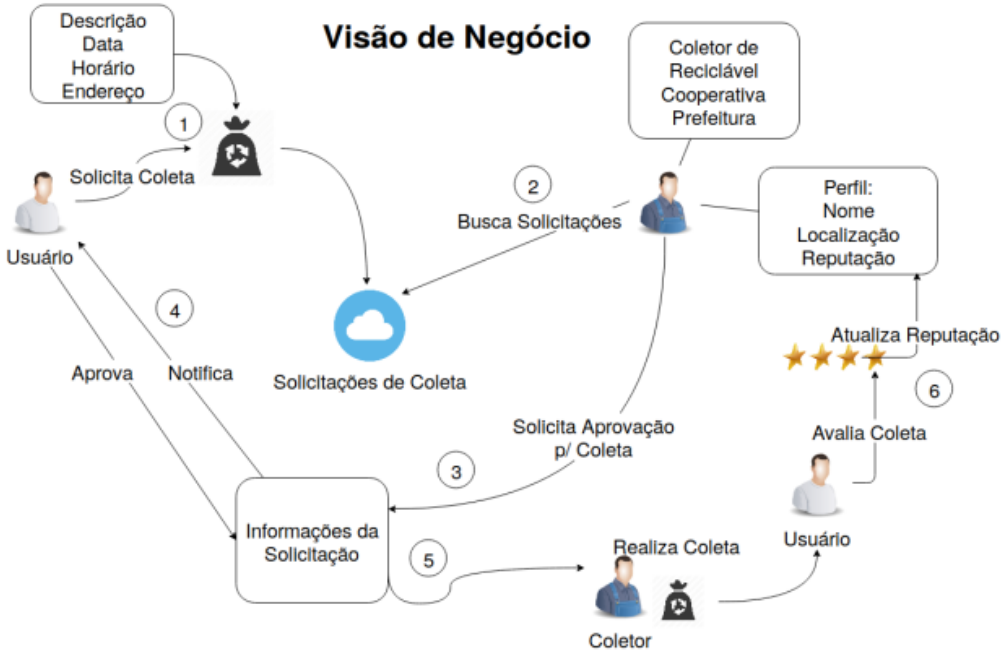

Figura 6. Visão geral de negócio. Fonte: Elaborado pelo autor.

5.2 Criação dos Protótipos das Telas e Histórias de Usuário

Após a definição do fluxo de negócio, foram criados os protótipos das telas do aplicativo. Um protótipo de uma interface de usuário é uma hipótese, uma solução candidata a um problema específico de *design* [Pernice 2016]. No protótipo é definido onde cada elemento deve ser posicionado, quais campos devem existir, quais as ações que serão feitas quando o usuário fizer alguma interação, etc.

Conforme os protótipos ficavam prontos, eram definidos as histórias de usuários e os critérios de “aceite” atrelados às respectivas telas. A figura 7 mostra um exemplo de um protótipo de uma tela, localizada na parte esquerda da imagem, e seus critérios de “aceite”, localizados à direita. A tela é para avaliação de um usuário após o ato da coleta. Tanto quem solicitou a coleta quanto quem coletou se avaliam e, a partir dessas avaliações, o aplicativo atualiza a reputação desses usuários com base na média de avaliações já feitas para os mesmos.

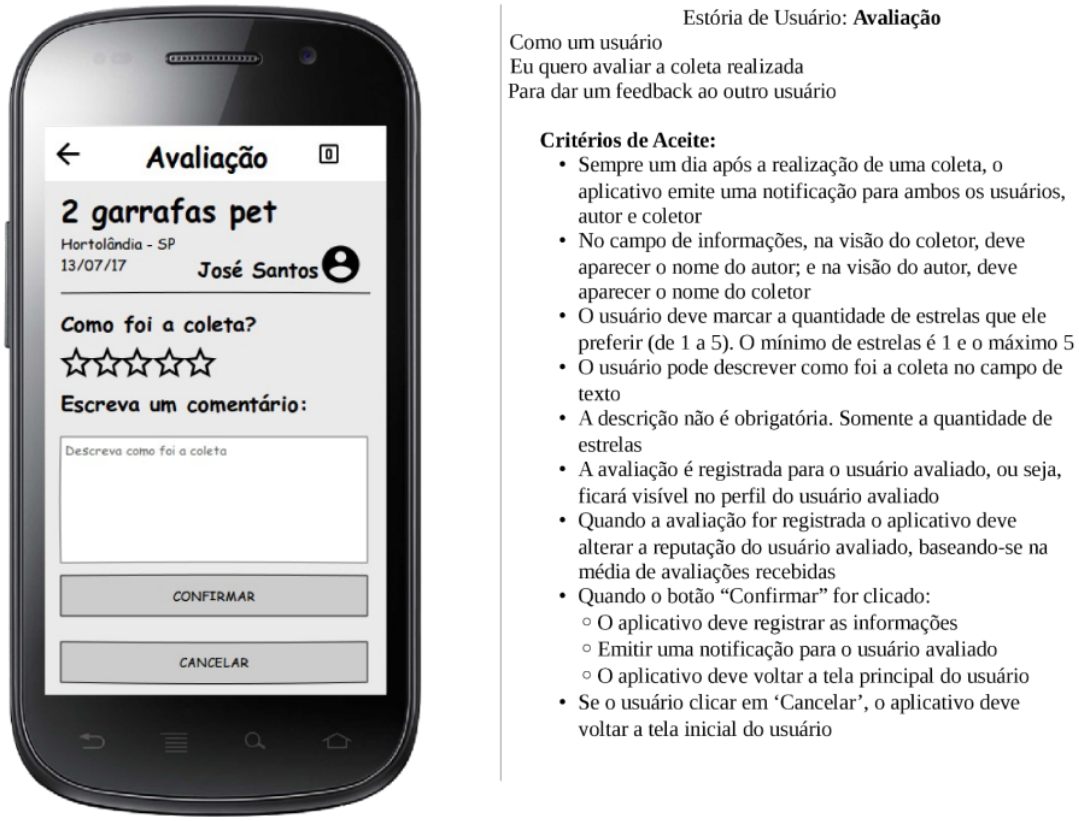

Figura 7. Protótipo da tela de avaliação de usuário. Fonte: Elaborado pelo autor.

5.3 Desenho da Solução Técnica

Na fase de elaboração do desenho de solução técnica foi feito um levantamento das ferramentas apropriadas para o desenvolvimento do aplicativo *Recycler* e do *Web Service*, além da definição da arquitetura da solução.

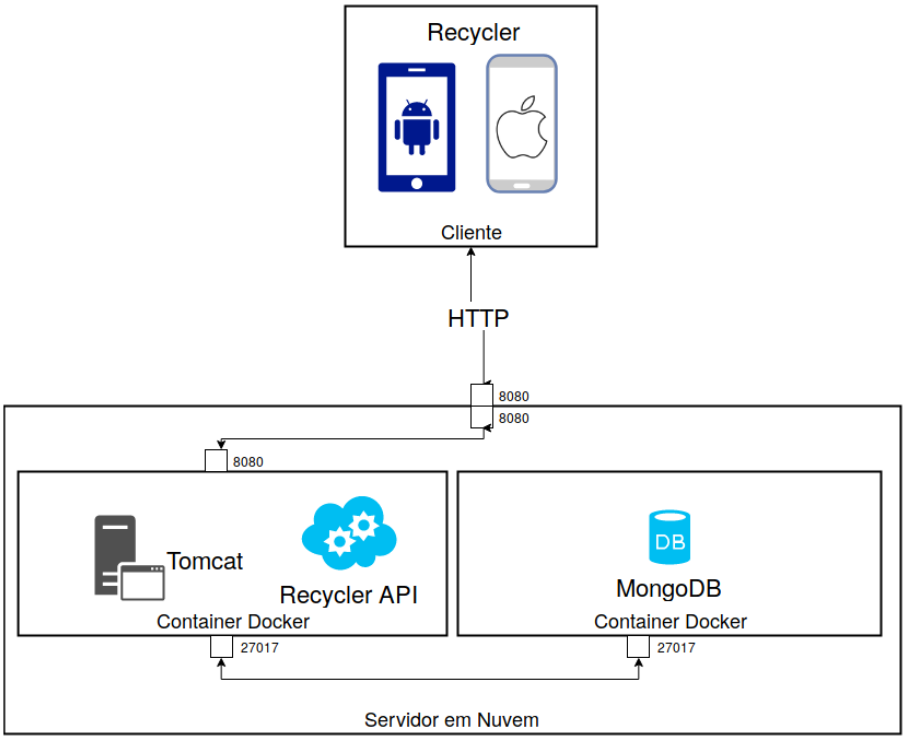

Figura 8. Visão Geral da Arquitetura. Fonte: Elaborado pelo autor.

A figura 8 mostra a visão geral da arquitetura. Ela é composta pelos seguintes itens:

- Aplicativo *Recycler*, que atua como cliente consumidor do *Web Service*. O aplicativo pode estar em um dispositivo *Android* ou *iOS*. O *Recycler* se comunica com o servidor em "nuvem", usando o protocolo HTTP, através da porta 8080;
- Servidor em "nuvem", que possui dois contêineres *Docker* [Wang 2017];
- Contêiner usado para executar o servidor de aplicações *Tomcat*, que é usado para servir a aplicação *Recycler API*. Esse contêiner se comunica com servidor através da porta 8080;
- Contêiner usado para executar o *MongoDB*, que é o gerenciador de bancos de dados usado pela aplicação *Recycler API*;
- Os dois contêineres, do *Web Service* e do banco de dados se comunicam através da porta 27017.

5.4 Banco de Dados

O sistema gerenciador do banco de dados usado foi o *MongoDB*. Este é projetado para desenvolver aplicações *Web* rapidamente, além de possuir um modelo de dados e estratégias de persistências construídos para uma alta taxa de transferência durante a leitura e escrita dos dados [Banker 2016].

O *MongoDB* foi escolhido, principalmente, pelo ganho de produtividade que ele agrega, pois armazena dados no formato JSON. Esse formato é usado em todas as partes da aplicação, tanto no *Web Service*, quanto no aplicativo. Sendo assim, não há a necessidade de conversões dos dados persistidos. Outro motivo importante está no fato do *MongoDB* suportar agregações de objetos. Essas agregações otimizam as consultas dos dados que, no modelo relacional, estariam separados em várias tabelas havendo a necessidade de juntar essas tabelas durante uma consulta.

Figura 9. Exemplo de um registro de uma solicitação de coleta feita pelo aplicativo *Recycler*, no *MongoDB*. Fonte: Elaborado pelo autor.

A figura 9. mostra um exemplo de um registro que contém as informações de uma solicitação de coleta. É possível observar que no campo *interested Users* e *address* é armazenado um objeto inteiro em vez de ser armazenada uma referência, como é o caso do campo *user*.

6. Desenvolvimento do Web Service

Após a definição dos protótipos das telas foi feita a especificação dos serviços que seriam consumidos pelo aplicativo em cada uma das telas. O *Web Service* usa a arquitetura REST (*Representational State Transfer*) baseado no modelo de maturidade introduzido por Leonard Richardson [Fowler 2010] no nível 0 e a aplicação está usando o protocolo HTTP para interações remotas. No nível 1, tem-se a separação dos serviços em recursos; no nível 2, os verbos HTTP são usados de acordo com o que foram projetados e no nível 3 é usado o conceito de controle de hipermídia.

Para alcançar o nível 1, foi preciso identificar os recursos que são chamados de URIs (*Uniform Resource Identifier*) no qual seriam expostos pelo *Web Service* e, para o nível 2, o uso correto dos verbos HTTP. Alguns recursos e verbos são descritos na tabela 1.

Tabela 1. URIs e verbos HTTP usados no *Web Service*

URI	Verbos HTTP	Descrição
/collectionrequests	GET	Listar as solicitações de coletas
/collectionrequests	POST	Solicitar uma coleta
/collectionrequests/{id}	PUT	Atualizar uma solicitação
/collectionrequests/{id}	DELETE	Remover uma solicitaão
/collectionrequests/{id}/comments	GET	Listar os comentários de uma solicitação
/collectionrequests/{id}/comments	POST	Adicionar um comentário a uma solicitação

O aplicativo *Recycler* realiza chamadas à Web *Service* usando objetos JSON (*Java Script Object Notation*) e recebe as respostas com o mesmo tipo de objeto. Esse objeto é recebido pelo *Web Service*, tratado e persistido no banco de dados. A figura 10 mostra um exemplo de um objeto usado para realizar uma solicitação de coleta.

```
{
  "description": "2 caixas de papelão",
  "authorId": "593a8b1b995ff11341fb6735",
  "collectionDate": "2017-05-14",
  "minTime": "1503000000000",
  "maxTime": "1502973000000",
  "address": {
    "fullAddress": "Rua joaninha",
    "neighborhood": "Jd. Nova Europa",
    "city": "Hortolândia",
    "state": "SP"
  }
}
```

Figura 10. Exemplo de um objeto de chamada para solicitar uma coleta. Fonte: Elaborado pelo autor.

Para a implantação do *Web Service* foi usada a linguagem de programação *Java* na versão 8 com o auxílio do *Spring Framework* na versão 4. O servidor de aplicação usado foi o *Apache Tomcat* 8. Importante destacar que o servidor de aplicação *Tomcat* é embutido no artefato onde carrega a aplicação, diferenciando de métodos tradicionais onde uma aplicação era executada em um servidor de aplicações instalado e configurado em algum servidor físico. O artefato da aplicação é executado dentro de um *container Docker*, assim como o banco de dados. Isso facilita a implantação de novas versões, pois não é necessária a realização de diversas configurações para que a aplicação o execute.

7. Desenvolvimento do Aplicativo

O aplicativo foi implementado após a construção do *Web Service*, pois assim não há a necessidade de simular as chamadas ao *Web Service* e, posteriormente, trocar as chamadas

para o serviço real. Para a implementação do aplicativo *Recycler* foi usado o *Ionic Framework*, que é uma ferramenta para o desenvolvimento de aplicativos híbridos e possui as seguintes características, que serviram como base para a decisão de usá-la no trabalho:

- Tem como base o *Angular Framework*, que é uma ferramenta, mantida pela *Google*, para o desenvolvimento *Web*;
- Possui componentes de interface que se adaptam de acordo com a plataforma;
- Possui *plugins* para acesso a funcionalidades nativas de dispositivos móveis.

7.1 Funcionalidades do Aplicativo

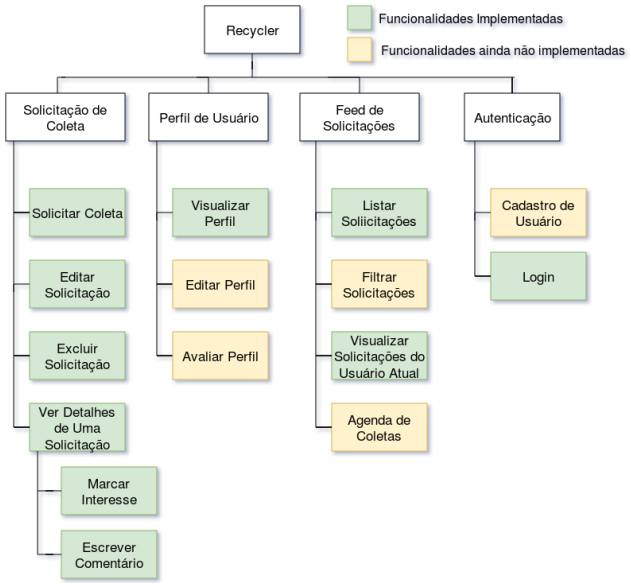

Figura 11. Exemplo de um objeto de chamada para solicitar uma coleta. Fonte: Elaborado pelo autor.

A figura 11 mostra todas as funcionalidades do aplicativo. Em verde estão as funcionalidades que foram desenvolvidas ao longo do trabalho. Em amarelo estão as funcionalidades que foram planejadas, porém ainda precisam ser implementadas.

A seguir serão mostradas algumas funcionalidades desenvolvidas no aplicativo *Recycler*.

7.1.1 Autenticação de Usuário

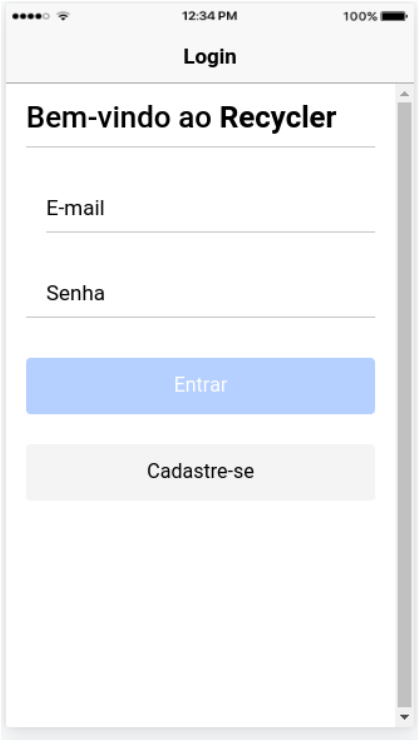

Figura 12. Tela de autenticação de usuário vista em um dispositivo *iOS*. Fonte: Elaborado pelo autor.

O aplicativo conta com uma tela inicial de autenticação onde o usuário fornece seu *e-mail* e senha respectivamente. Caso os dados informados correspondam a algum usuário já registrado, o aplicativo abre a tela principal, que é a tela para listagem de solicitação de coletas na qual guarda as credenciais de acesso desse usuário localmente para que no próximo acesso não seja preciso efetuar o *login* novamente. O usuário, ainda na tela de autenticação, pode clicar no botão para cadastrar-se no aplicativo. A figura 12 demonstra como é a tela de autenticação.

7.1.2 Solicitar Nova Coleta

Quando for selecionada no *menu* a opção para solicitar coleta, o aplicativo exibe a tela correspondente à designada ação. Nessa tela são exibidos alguns campos de texto objetivando o preenchimento das informações. Alguns campos são preenchidos automaticamente, como a data, o horário e o endereço. O usuário pode usar um endereço diferente do seu padrão, tornando mais flexível a disponibilidade de materiais em diferentes locais. Após o usuário informar os dados necessários, essa solicitação é registrada e fica disponível na lista de solicitações para os demais.

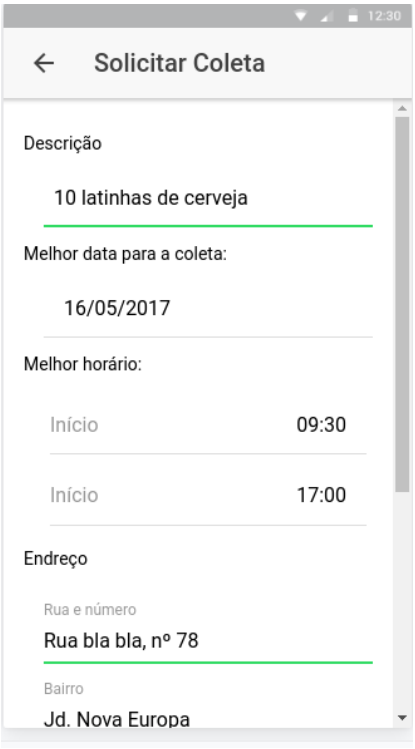

Figura 13. Tela para solicitar uma coleta vista em um dispositivo *Android.* Fonte: Elaborado pelo autor.

7.1.3 Detalhes de Uma Solicitação de Coleta

Na tela de detalhes são apresentadas todas as informações de uma solicitação de coleta como, por exemplo, a melhor data e melhor horário para que o material possa ser coletado e/ou o endereço do autor da solicitação. Vale ressaltar que o endereço é visível parcialmente, sendo exibido por completo apenas para quem for designado para coletar o material. Nessa tela é possível enviar um comentário ao autor da solicitação a fim de uma melhor comunicação entre as partes. Na tela de detalhes é possível ver a lista de usuários interessados em realizar a coleta do material publicado, onde o autor da solicitação pode confirmar (ou não) a coleta com algum interessado. No panorama de um usuário, que não é o autor da solicitação, é exibido um botão no meio da tela mostrando que o usuário possui interesse em coletar o material. Caso seja clicado, ele é encaminhado à lista de interessados. A figura 14 ilustra como é a tela de detalhes em um dispositivo *Android*.

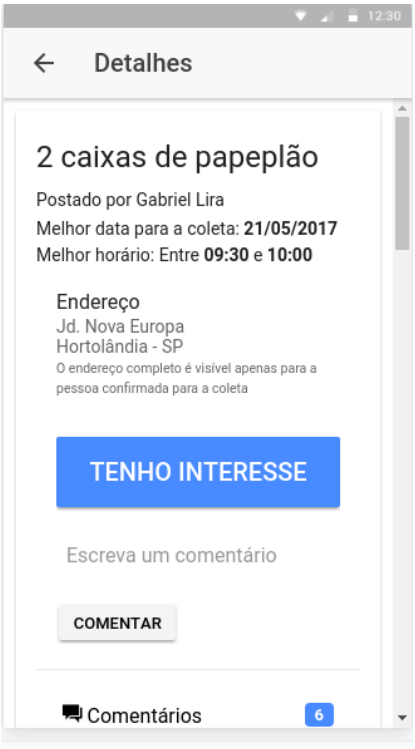

Figura 14. Tela com os detalhes de uma solicitação de coleta na visão de um coletor de recicláveis em um dispositivo *Android*. Fonte: Elaborado pelo autor.

7.1.4 Listagem de Solicitações de Coleta

A tela principal do aplicativo é destinada à listagem de solicitações de coleta. Nessa tela são exibidas todas as solicitações feitas que ainda não foram confirmadas. É possível realizar buscas por solicitações filtrando pela descrição do material a ser coletado ou pela cidade onde o mesmo encontra-se disponível. Cada item da lista representa uma solicitação mostrando a descrição do material para ser coletado, a cidade e o estado em que o material está localizado, bem como a data de publicação dessa solicitação e a foto do autor que a fez, como mostra a figura 15. Ao clicar em algum item, o usuário é direcionado à tela detalhada da solicitação de coleta que o item representa.

Figura 15. Tela com a lista de solicitações de coleta vista em um dispositivo Android. Fonte: Elaborado pelo autor.

8. Conclusões

A coleta seletiva no Brasil ainda não é realizada na maioria das cidades. Para descartar algum material, atualmente, é preciso levar a algum ponto de entrega voluntária que receba esse material ou deixá-lo exposto à rua ansiando que alguém possa coletá-lo. Dado o constante crescimento no uso de dispositivos móveis, foi desenvolvido o aplicativo *Recycler* para oferecer outra forma de descarte aos materiais recicláveis. Essa nova forma, apresenta-se, potencialmente, como um meio de incentivar e otimizar esse serviço no Brasil, sendo viabilizadora da comunicação entres todos os seus envolvidos. Por isso, destaca-se a relevância social desse projeto, em particular, se sua utilização ocorrer de forma massiva em diversas municípios e comunidades.

Com o desenvolvimento deste trabalho foi possível empregar conhecimentos adquiridos ao longo do curso, tais como engenharia de *software*, análise orientada a objetos, arquitetura de *software* e desenvolvimento web. Os conhecimentos das linguagens de programação *Java* e *JavaScript* foram importantes para o desenvolvimento do *Recycler*.

O aplicativo *Recycler* foi implementado parcialmente, pois nem todas as funcionalidades planejadas foram desenvolvidas possuindo apenas as principais para o objetivo do aplicativo. Durante o desenvolvimento do aplicativo foram necessários

moderados ajustes na implementação do *Web Service* para atender corretamente aos fluxos possuídos pelo aplicativo.

Como contribuição, o código fonte do aplicativo *Recycler*[6], assim como o do *Web Service*[7], está disponível no *Github* podendo ser consultado, modificado e expandido.

Por fim, o objetivo do *Recycler* é facilitar a vida de pessoas interessadas em descartar materiais recicláveis e daquelas que têm interesse em coletar tais materiais, possibilitando assim um aumento na renda de trabalhadores e autônomos que trabalham com esse tipo de material.

9. Trabalhos Futuros

Esta seção apresenta sugestões de trabalhos futuros que podem ser seguidas por pessoas interessadas em expandir o aplicativo *Recycler*.

Durante a fase de planejamento do aplicativo foi cogitada a ideia de realizar avaliações de usabilidade com as pessoas pertencentes ao público alvo do projeto, como interessados em descartar materiais recicláveis e, principalmente, os coletores desses materiais, que são os maiores beneficiados pelo aplicativo. Através dessas avaliações, podem ser colhidos *feedbacks* dos usuários ocasionando a evolução do aplicativo para melhor atender as necessidades deles.

Como o aplicativo foi desenvolvido parcialmente, algumas funcionalidades ainda podem ser desenvolvidas, como a exibição das coletas confirmadas para um usuário na forma de um calendário.Podem ser adicionadas mais opções de login, usando contas do *Google* e do *Facebook.*

Esse trabalho compõe apenas um aplicativo para dispositivos móveis, podendo ser criado um *site* para divulgar informações do aplicativo e até mesmo realizar o fluxo de solicitação de coletas.

10. Referências

ABRELPE (2016). “Panorama dos resíduos sólidos no brasil.”, http://www.abrelpe.org.br/Panorama/panorama2016.pdf.

Austins, C. (2017). “Top 6 programming languages for mobile app development.”, https://dzone.com/articles/top-6-programming-languages-for-mobile-app-develop.

Bôas, B. V. (2016). “Celular se torna principal meio de acesso à internet nos lares, diz ibge.”, http://www1.folha.uol.com.br/mercado/2016/04/1757972-celular-se-torna-principal-meio-de-acesso-a-internet-nos-lares-diz-ibge.shtml

Carneiro, A. (2013). “Web services em aplicações android e ios.”, https://www.devmedia.com.br/web-services-em-aplicacoes-android-e-ios/28901.

[6] https://github.com/gabriellira/recycler

[7] https://github.com/gabriellira/recycler-api

Carvalho, D. (2014). “Catadores do recife ’usam’ aplicativo para fazer reciclagem.”, http://www1.folha.uol.com.br/cotidiano/2014/11/1544092-catadores-do-recife-usam-aplicativo-para-fazer-reciclagem.shtml.

CEMPRE (2015). “Pesquisa Ciclosoft 2016.”, http://cempre.org.br/ciclosoft/id/8.

Cheng, F. (2017). Build Mobile Apps with Ionic 2 and Firebase: Hybrid Mobile App Development, Apress Media LLC, 1ª edição.

Cohn, M. (2004). “Advantages of user stories for requirements.”, https://www.mountaingoatsoftware.com/articles/advantages-of-user-stories-for-requirements

Folha de São Paulo (2016). “O número de smartphones em uso no brasil chega a 168 milhões, diz estudo.”, http://www1.folha.uol.com.br/mercado/2016/04/1761310-numero-de-smartphones-em-uso-no-brasil-chega-a-168-milhoes-diz-estudo.shtml

Fowler, M. (2010). “Richardson maturity model.”, https://martinfowler.com/articles/richardsonMaturityModel.html

Banker, K. e Bakkum, P. (2016). MongoDB in Action, MANNING, 2ª edição.

Leavitt, N. (2010). “Will nosql databases live up to their promise.“, http://leavcom.com/pdf/NoSQL.pdf

Levison, M. (2017). “Definition of done vs. user stories vs. acceptance criteria.“, https://agilepainrelief.com/notesfromatooluser/2017/05/definition-of-done-vs-user-stories-vs-acceptance-criteria.html#.WgDenBYfO01.

Ministério do Meio Ambiente. (2015). “Coleta seletiva.”, http://www.mma.gov.br/cidades-sustentaveis/residuos-solidos/catadores-de-materiais-reciclaveis/reciclagem-e-reaproveitamento

Moretzsohn, L. (2017). “IR 2017: Uber, Airbnb e plataformas da economia compartilhada.“, https://oglobo.globo.com/economia/ir-2017-uber-airbnb-plataformas-da-economia-compartilhada-21217750.

Paganini, S. (2010). “Melhorando a comunicação com estórias do usuário.”, https://www.devmedia.com.br/melhorando-a-comunicacao-com-estorias-do-usuario-java-magazine-86/18785.

Pernice, K. (2016). “Ux prototypes: Low fidelity vs. high fidelity.”, https://www.thoughtworks.com/insights/blog/nosql-databases-overview.

Pinelli, N. (2017). “Economia compartilhada.”, http://epocanegocios.globo.com/Caminhos-para-o-futuro/Desenvolvimento/noticia/2017/01/economia-compartilhada.html.

Sadalage, P. (2014). “Nosql databases: An overview.”, https://www.thoughtworks.com/insights/blog/nosql-databases-overview.

Sommerville, I. (2011). Software Engineering, Addison-Wesley, 9ª edição.

Strauch, C. (2011). “Nosql databases.”, Stuttgart Media University, p. 30-50.

Viswanathan, P. (2017). “Native apps vs. web apps: What is the better choice.”, https://www.lifewire.com/native-apps-vs-web-apps-2373133

Wang, C. (2017). “What is docker? linux containers explained.”, https://www.infoworld.com/article/3204171/linux/what-is-docker-linux-containers-explained.html.

Woods, V. and Meulen, R. (2016). “Worldwide smartphone sales grew 3.9 percent in first quarter of 2016.”, http://www.gartner.com/newsroom/id/3323017.