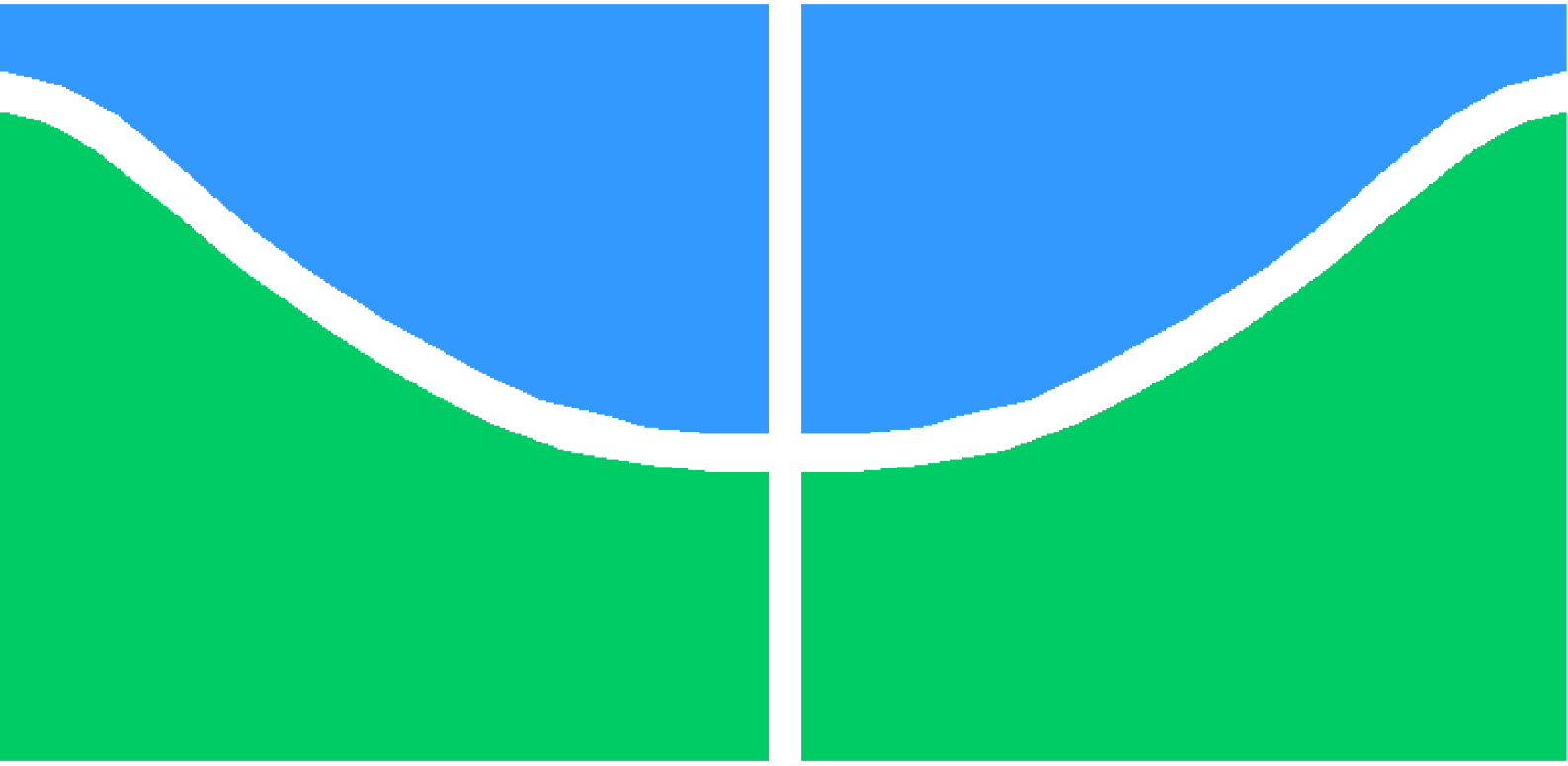


Universidade de Brasília - UnB
Faculdade UnB Gama - FGA
Engenharia de Software

Sustentaê: aplicativo para divulgação de locais sustentáveis

Autores: Gabriel de Araújo Silva, Luís Filipe Resende Vilela
Orientador: Prof. Msc. Giovanni Almeida Santos

Brasília, DF
2018

Gabriel de Araújo Silva, Luís Filipe Resende Vilela

Sustentaê: aplicativo para divulgação de locais sustentáveis

Monografia submetida ao curso de graduação em *Engenharia de Software* da Universidade de Brasília, como requisito parcial para obtenção do Título de Bacharel em Engenharia de Software.

Universidade de Brasília - UnB

Faculdade UnB Gama - FGA

Orientador: Prof. Msc. Giovanni Almeida Santos

Brasília, DF

2018

Gabriel de Araújo Silva, Luís Filipe Resende Vilela

Sustentaê: aplicativo para divulgação de locais sustentáveis/ Gabriel de Araújo Silva, Luís Filipe Resende Vilela. – Brasília, DF, 2018-

106 p. : il. (algumas color.) ; 30 cm.

Orientador: Prof. Msc. Giovanni Almeida Santos

Trabalho de Conclusão de Curso – Universidade de Brasília - UnB
Faculdade UnB Gama - FGA , 2018.

1. sustentável. 2. rede social. I. Prof. Msc. Giovanni Almeida Santos. II. Universidade de Brasília. III. Faculdade UnB Gama. IV. *Sustentaê*: aplicativo para divulgação de locais sustentáveis

CDU 02:141:005.6

Gabriel de Araújo Silva, Luís Filipe Resende Vilela

Sustentaê: aplicativo para divulgação de locais sustentáveis

Monografia submetida ao curso de graduação em *Engenharia de Software* da Universidade de Brasília, como requisito parcial para obtenção do Título de Bacharel em Engenharia de Software.

Trabalho aprovado. Brasília, DF, 03 de Dezembro de 2018:

Prof. Msc. Giovanni Almeida Santos
Orientador

Prof. Dr. Wander Cleber M. Pereira da Silva
Convidado 1

Prof. Dr. Vandor Roberto Vilardi Rissoli
Convidado 2

Brasília, DF
2018

Agradecimentos

Eu, Gabriel de Araújo Silva, agradeço primeiro a Deus. Agradeço aos meus pais e a minha irmã por sempre me apoiarem durante a graduação e sempre me animaram quando desanimado. Agradeço aos companheiros dessa jornada que, muitas vezes, viraram noites estudando e/ou realizando trabalhos. A contribuição, os conselhos e a amizade de todos teve papel essencial para a conclusão dessa graduação.

Eu, Luís Filipe Resende Vilela, agradeço primeiramente a Deus por me dar essa oportunidade de poder entrar e caminhar no meio acadêmico. Agradeço aos meus pais, Doralice e Dárcio, e meu irmão, Gustavo, por estarem comigo me apoiando e me incentivando durante essa caminhada. Agradeço a minha namorada, Laura Freire, por sempre estar ao meu lado desde o início de minha graduação e em todos os outros momentos em minha vida. Quero agradecer a todo o restante da minha família, meus amigos e companheiros de graduação que compartilharam seus conhecimentos, esforços, conselhos e apoio para me ajudar de alguma forma a chegar a conclusão dessa graduação.

"Se A é o sucesso, então A é igual a X mais Y mais Z.
O trabalho é X; Y é a dedicação; e Z é não fazer nada sozinho."

Resumo

Um estilo de vida que leva em consideração a sustentabilidade tem-se tornado comum nos dias atuais. Pode-se dizer que isso é consequência do fácil acesso à informação como também do alto grau de exploração dos recursos naturais realizados pelo homem. O *Sustentaê* é um aplicativo desenvolvido para *iOS* que tem como objetivo impulsionar a descoberta de locais sustentáveis e, consequentemente, incentivar a adoção de um estilo de vida mais sustentável. O aplicativo escrito em *Swift*, linguagem utilizada em desenvolvimento nativo de aplicativo *iOS*, interage com uma *API* desenvolvida com o *framework Parse*. Para o desenvolvimento do aplicativo foram empregados conceitos, técnicas e metodologias relacionadas à Engenharia de Software, isto é, a utilização de um processo de desenvolvimento, *Scrum*, testes automatizados, integração e entrega contínua.

Palavras-chaves: Sustentabilidade. Aplicativo. iOS.

Abstract

A lifestyle that takes sustainability into account has become commonplace today. It can be said that this is a consequence of easy access to information as well as the high degree of exploitation of natural resources carried out by man. *Sustentaê* is an application developed for iOS that aims to boost the discovery of sustainable places and consequently encourage the adoption of a more sustainable lifestyle. The application written in Swift, a language used in native iOS application development, interacted with an API developed under the Parse framework. For application development, concepts, techniques and methodologies of Software Engineering were employed, in the use of a development process, Scrum, automated testing, integration and continuous delivery.

Key-words: Sustainability. App. iOS.

Lista de figuras

Lista de tabelas

Lista de abreviaturas e siglas

API - *Application Programming Interface*

AWS - *Amazon Web Services*

BPMN - *Business Process Modeling Notation*

CI - *Continuous integration*

CSS - *Cascading Style Sheets*

DAO - *Data Access Object*

EITA - Educação, Informação e Tecnologia para Autogestão

EUA - Estados Unidos da América

HTML - *Hypertext Markup Language*

HTTP - *Hypertext Transfer Protocol*

Ibope - Instituto Brasileiro de Opinião Pública e Estatística

IDE - *Integrated development environment*

Idec - Instituto Brasileiro de Defesa do Consumidor

iOS - *iPhone Operating System*

JS - *JavaScript*

JSON - *JavaScript Object Notation*

MVC - *Model-View-Controller*

NoSQL - *Not Only Structured Query Language*

ONU - Organização das Nações Unidas

PaaS - *Platform as a Service*

PNUMA - Programa das Nações Unidas para o Meio Ambiente

REST - *Representational state transfer*

SDK - *Software Development Kit*

SUS - *System Usability Scale*

TCC - Trabalho de Conclusão de Curso

URI - *Uniform Resource Identifier*

URL - *Uniform Resource Locator*

XML - *Extensible Markup Language*

Sumário

1 Introdução

No contexto da sociedade atual, há uma grande preocupação com a forma de consumo vigente (GRANERO; COUTO, 2013). Pode-se atribuir essa preocupação aos estudos e indícios que mostram transformações climáticas na terra (SHOVE, 2010). Além disso, está atrelada à percepção do consumidor de que a escala das ações humanas se tornaram tão grandes que está alterando os ecosistemas de forma mais rápida que a sua capacidade de restauração (SCHUMACHER, 1999). Tendo em vista isso, é evidente a necessidade e a importância de se viver de forma mais sustentável em um planeta com recursos limitados afetado também por vários impactos ambientais (ALVES; SILVA; FERREIRA, 2008).

A sustentabilidade é um conceito que, relacionando os aspectos econômicos, sociais, culturais e ambientais, busca suprir as necessidades do presente sem afetar as gerações futuras (FERREIRA, 2004). Este conceito aliado ao impacto que o tema possui, gerou um termo denominado **economia verde**.

O aumento da preocupação da sociedade e as normas governamentais sobre o desenvolvimento econômico sustentável levou um maior numero de indústrias e serviços a lançarem linhas de produtos que preferencialmente se preocupam com os impactos ambientais que causam, além de aderirem conceitos sustentáveis no processo de fabricação (HU, 2012). Como consequência, companhias entenderam que o mercado de produtos sustentáveis representa uma oportunidade de lucro e, portanto, a adesão do fator ambiente na estratégia de tomada de decisões se torna essencial (ALVES; SILVA; FERREIRA, 2008).

Outra caracteristica na sociedade moderna é o uso constante de aplicativos móveis que são acessados por meio de *smartphones*. Esses dispositivos facilitam o acesso a informação, pois podem ser levados a qualquer lugar por qualquer um que o possua. Eles são cada vez mais usados conectados à Internet (GRANERO; COUTO, 2013). Segundo Granero e Couto (2013), de acordo com um estudo global feito pela *AdReaction*, em parceria com o Ibope (Instituto Brasileiro de Opinião Pública e Estatística) no Brasil, dois terços das pessoas consideram que são mais eficientes usando os *smartphones*, e que 60% dos seus usuários julgam os dispositivos móveis indispensáveis no dia a dia.

Levando em consideração os fatores da economia verde, a necessidade de se criar uma rede global de consumo sustentável e a capacidade de penetração na sociedade dos aplicativos móveis, o foco deste trabalho foi o desenvolvimento de um aplicativo capaz de colaborar para a promoção da sustentabilidade.

Esse aplicativo foi idealizado e criado por Vanessa Davim Raulino, no seu TCC

(Trabalho de Conclusão de Curso) em Bacharel em Comunicação Social, com habilitação em Publicidade e Propaganda na Universidade Católica de Brasília em 2017. A ideia principal do aplicativo é ajudar na preservação do meio ambiente e na promoção da sustentabilidade, por meio da conexão entre produtores e consumidores, contribuindo para a construção de um ecossistema saudável e sustentável (RAULINO, 2017).

O aplicativo *Sustentaê* estabelece uma plataforma unificada de locais e produtos ligados a sustentabilidade, criando micro redes de interação entre indivíduos para a divulgação de maneiras, lugares e produtos que visam um consumo sustentável. O trabalho realizado por Vanessa Raulino teve como um dos resultados a concretização do conceito do *Sustentaê*, por meio do design da *interface* das principais funcionalidades do aplicativo. Entretanto, os resultados não abrangem qualquer estudo ou produção relacionados à implementação do aplicativo como produto de software (RAULINO, 2017).

A proposta deste Trabalho de Conclusão de Curso (TCC) em Bacharel em Engenharia de Software foi a implementação do aplicativo *Sustentaê* em que foi realizado o levantamento dos seus requisitos e a validação e a aceitação pelos seus usuários.

1.1 Questão de Pesquisa

O intuito desse trabalho foi realizar o desenvolvimento do aplicativo *Sustentaê* utilizando-se de conceitos da Engenharia de Software, que foi realizado em três atividades principais. A primeira se refere à engenharia de requisitos do projeto, a segunda, a implementação destes requisitos e a terceira contempla a validação do projeto por usuários em ambiente controlado.

Em resumo, esse trabalho visou responder a seguinte pergunta: *Como realizar a implementação do aplicativo Sustentaê, de modo que forneça à comunidade uma plataforma unificada de produtos e locais ligados à sustentabilidade, aproveitando conceitos da Engenharia de Software?*

1.2 Justificativa

Segundo Alves, Silva e Ferreira (2008), em um mundo com recursos limitados e com vários impactos ambientais, é óbvio que um estilo de vida mais sustentável será muito importante. Dessa forma, as empresas passaram a investir espontaneamente no mercado verde adotando os selos de meio ambiente amigável (CAIRNCROSS, 1992). Entretanto, é importante saber se as ações realizadas por tais empresas, como aquelas expostas em suas propagandas, são realmente realizadas e se são relacionadas ao meio ambiente (ALVES; SILVA; FERREIRA, 2008).

Uma forma de se confirmar essas ações é a validação pela própria sociedade. Atualmente, no meio digital, essa validação é feita de diversas formas: por meio de avaliações em *sites*, *blogs* e mensagens em redes sociais. No entanto, este comportamento não fornece um meio unificado destas informações, existindo apenas pontos desconectados das validações. Isso torna difícil realizar uma análise dos dados coletados pela sociedade.

Considerando isto, o projeto *Sustentaê* propõe uma plataforma para a unificação destas informações, o que torna possível a criação de um perfil da sustentabilidade das empresas, lugares e serviçoes presentes na sociedade.

1.3 Objetivos

1.3.1 Objetivo Geral

Desenvolver uma plataforma unificada para a disponibilização de informações sobre estabelecimentos e produtos sustentáveis, acessada por meio de aplicativo móvel.

1.3.2 Objetivos Específicos

1. Elicitar requisitos do projeto levando em conta as denifições atuais do trabalho de conclusão de curso (TCC) que inspirou este trabalho.

2. Estabelecer metodologia de desenvolvimento, padrões de projeto, arquitetura da solução e demais boas práticas da Engenharia de Software mais adequadas para elaboração do projeto.

3. Priorizar e implementar os requisitos do projeto.

4. Definir metodologia para ser utilizada na validação do produto com usuários em ambiente controlado.

5. Executar a metodologia de validação e analisar os resultados.

1.4 Organização do Documento

Os próximos capítulos desse documento estão organizados da seguinte forma:

- **Capítulo 2 - Referencial Teórico:** apresenta os principais conceitos relacionados ao tema desse trabalho.

- **Capítulo 3 - Suporte Tecnológico:** apresenta e justifica o uso das principais ferramentas utilizadas como suporte para a realização desse trabalho.

- **Capítulo 4 - Metodologia:** especifica a metodologia utilizada para o desenvolvimento da pesquisa e implementação do trabalho.
- **Capítulo 5 - Sustentaê:** apresenta o produto final desenvolvido.
- **Capítulo 6 - Considerações Finais:** apresenta as conclusões do trabalho.

2 Referencial Teórico

2.1 Sustentabilidade

O conceito de sustentabilidade tem sido citado com muita frequência nos últimos anos. Sendo debatido em pesquisas, em cúpulas ambientais entre países de todo o mundo e no meio empresarial. Dessa forma, várias interpretações desse conceito foram desenvolvidas e sofreram uma grande influência dos objetivos dos estudos relacionados, tornando amplo o significado do conceito de sustentabilidade (MIKHAILOVA, 2004).

Nas subsessões seguintes, visa-se explicar o conceito de sustentabilidade que influenciou este trabalho.

2.1.1 Conceito

A ONU (Organização das Nações Unidas), por meio do relatório Nosso Futuro Comum, publicado pela Comissão Mundial para o Meio Ambiente e o Desenvolvimento em 1987, elaborou o seguinte conceito:

> *Desenvolvimento sustentável é aquele que busca as necessidades presentes sem comprometer a capacidade das gerações futuras de atender suas próprias necessidades (BRUNDTLAND et al., 1987).*

De forma direta, pode-se dizer que a sustentabilidade é a habilidade de se sustentar, se manter. Dessa forma, uma atividade sustentável é aquela que pode ser explorada continuamente e não se esgotará. Semelhantemente, o desenvolvimento sustentável é a capacidade de evoluir a qualidade de vida do homem na terra respeitando a capacidade de produção dos ecossistemas (MIKHAILOVA, 2004).

2.1.2 Economia Verde

De acordo com o PNUMA (Programa das Nações Unidas para o Meio Ambiente), economia verde é definida como:

> *Uma economia que resulta em melhoria do bem estar da humanidade e igualdade social, ao mesmo tempo em que reduz significativamente riscos ambientais e escassez ecológica (PNUMA, 2011).*

Já a Secretaria do Meio Ambiente do Estado de São Paulo elaborou um documento em que define a economia verde como sendo:

> *uma agenda de desenvolvimento que propõe uma transformação na maneira de se encarar a relação entre crescimento econômico e desenvolvimento, indo muito além da visão tradicional do meio ambiente com um conjunto de limites para o crescimento ao encontrar nas mudanças climáticas e no escasseamento ecológico vetores para um crescimento mais sustentável.*

> *É uma forma de trazer a sustentabilidade, tão frequentemente e equivocadamente tratada como "tema do futuro", para um patamar de objetividade e pragmatismo que evidencia as vantagens econômicas e sociais da aliança entre inovação e melhora de qualidade ambiental (AMBIENTAL, 2010).*

De forma geral, na visão dos documentos citados a economia verde seria a prática de todos os preceitos que fazem parte do conceito de sustentabilidade, ou seja, cuidar do meio ambiente sem diminuir o desenvolvimento tecnológico.

2.2 *Sustentaê*

O conceito do aplicativo *Sustentaê* é o produto de um trabalho de conclusão de curso (TCC) realizado por Vanessa Raulino no ano de 2017. O trabalho foi concebido com o objetivo geral de que a aplicação agrupe locais e produtos que são sustentáveis, atingindo alguns objetivos específicos como: estabelecer uma plataforma unificada de locais e produtos sustentáveis; desenvolver um design intuitivo e atrativo para os usuários da aplicação e oportunizar aos usuários, por meio do aplicativo, a sua integração e interação (RAULINO, 2017).

O objetivo principal do conceito do aplicativo é interligar produtores e consumidores que possuem valores coletivos em comum, capazes de aplicar o direito de todos a um meio ambiente saudável, assim como o dever ético, moral e político de preservá-lo para as presentes e futuras gerações. O conceito do aplicativo foi concebido utilizando diversas técinas: como a produção baseada nos *designers* de experiência e de *interface* do usuário, *Wireframe*, criação do ícone e da identidade visual (RAULINO, 2017). Na subseção seguinte será apresentado as principais funcionalidades idealizadas para o aplicativo.

2.2.1 Funcionalidades Principais

A principal funcionalidade do aplicativo é a publicação de locais sustentáveis ou que vendam produtos sustentáveis (são chamados "achados"), por meio do compartilhamento da localização, um breve comentário e uma "nota sustentável", chamada de "Sustentei" que será feita em uma escala simples de 1 a 5. A ideia é que funcione de forma colaborativa, incentivando os próprios usuários a encontrarem esses lugares sustentáveis,

indicando e compartilhando com amigos (RAULINO, 2017). A Figura 1 mostra o conceito da tela de publicação de locais.

Figura 1 – Tela de publicação de novo local sustentável.

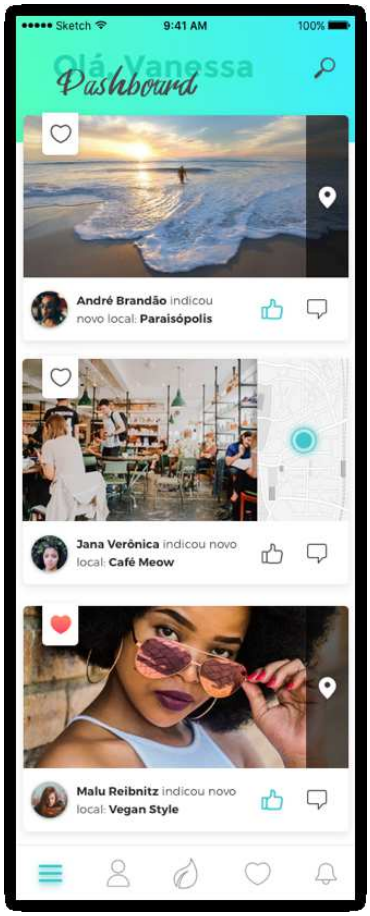
Fonte: (RAULINO, 2017).

Outra funcionalidade é o *Feed* - *feed*, do verbo em inglês "alimentar" é uma palavra que simboliza uma página que é atualizada frequentemente com novos contéudos - que mostra um painel com todas as novidades e publicações de quem o usuário segue. Ao clicar em uma publicação mostrada no *Feed* o usuário visualiza a localização em que o novo "achado" foi encontrado, quem publicou, a nota "Sustentei", o comentário feito, além de quantas pessoas já visitaram o local, número de "curtidas" e comentários de outros usuários (RAULINO, 2017).

Por fim, o aplicativo ainda possui algumas funcionalidades secundárias: busca de locais postados, perfil do usuário, lista de favoritos. A Figura 2 mostra o formato da tela de *Feed* e da tela de um "achado".

Figura 2 – Telas do *Feed*.

(a) Tela principal



(b) Detalhe de postagem

Fonte: (RAULINO, 2017).

2.2.2 Aplicativos na área de sustentabilidade

Atualmente, existem diversos aplicativos relacionados a área de sustentabilidade, desde guias para levar uma vida mais saudável, como o "Manual de Etiqueta – Planeta Sustentável", até outros que medem a emissão de gás carbônico que o usuário produz, o Carbon Z (CARBONZ, 2018).

Dentre esses aplicativos, foram analisados dois voltados a avaliação de locais, produtos ou serviços sustentáveis:

- Responsa[1]

[1] Link: <https://play.google.com/store/apps/details?id=br.org.eita.responsa>

- Mapa de Feiras Orgânicas[2]

Esses aplicativos foram selecionados para análise porque todos possuem a funcionalidade de descoberta de produtos e/ou produtores sustentáveis, uma vez que o conceito do aplicativo *Sustentaê* tem como principal funcionalidade a divulgação de informações similares. Uma análise comparativa entre o conceito e aplicativos que estão disponíveis no mercado é relevante para a validação do conceito. Pode-se mostrar assim quais as vantagens do *Sustentaê* sobre as outras soluções.

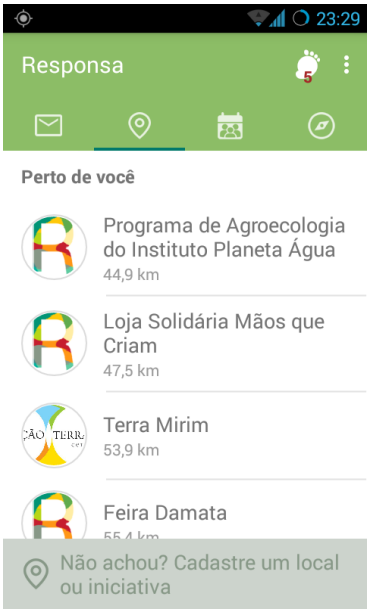
Outro fator que foi levado em consideração para a seleção desses dois aplicativos foi a área de atuação. Isto é, deve-se atuar no mercado brasileiro.

2.2.2.1 Responsa

O aplicativo Responsa foi criado em parceria entre o Instituto Kairós, situado em Nova Lima (MG), e a cooperativa de trabalho EITA (Educação, Informação e Tecnologia para Autogestão), localizado em Novo Hamburgo (RS). Por meio do Responsa o usuário pode encontrar locais, iniciativas e grupos que adotam e fomentam práticas de responsabilidade na produção e consumo. A Figura 3 mostra as telas onde o usuário pode identificar os estabelecimentos catalogados próximos a sua localização.

Figura 3 – Telas do aplicativo Responsa.

(a) Iniciativas perto do usuário



(b) Detalhe do local

Fonte: Aplicativo Responsa[3].

[2] Link: <https://itunes.apple.com/br/app/mapa-de-feiras-organicas/id1093189260?mt=8>

A ideia do aplicativo é de colocar em evidência essas iniciativas que adotam, em diferentes níveis, práticas de produção e comercialização com respeito e cuidado com o meio ambiente e com o próprio trabalhador atrelado á ela. É possível também interagir com as pessoas dentro do próprio aplicativo, sendo possível sugeir a criação de iniciativas em uma região específica, marcar encontros ou lançar desafios para sua comunidade. O aplicativo, porém, só está disponível para a plataforma Android.

2.2.2.2 Mapa de Feiras Orgânicas

O Mapa de Feiras Orgânicas é uma iniciativa da campanha Brasil Saudável e Sustentável, coordenada pelo Ministério do Desenvolvimento Social e Combate à Fome, por meio de parceria com o Instituto Brasileiro de Defesa do Consumidor (Idec). O objetivo do aplicativo é ajudar o usuário na busca por alimentos orgânicos e agroecológicos e propiciar melhores escolhas alimentares, encurtanto o caminho entre produtor e consumidor.

O aplicativo realiza a busca de feiras orgânicas no Brasil que estão mais próximas do usuário e apresenta detalhes de dia e horário em que as feiras estão abertas. Além disso, o sistema fornece uma base de dados de receitas associadas a alimentos regionais e de épocas, mostrando informações curiosas sobre o ingrediente principal da receita. As informações contidas no aplicativo são cadastradas de forma colaborativa e o Idec atualiza as informações sobre as feiras no *site* Mapa de Feiras Orgânicas (<https://www.feirasorganicas.org.br>). O aplicativo está disponível apenas para a plataforma iOS. A Figura 4 mostra a interface do aplicativo onde as feiras podem ser localizadas utilizando-se o mapa.

2.2.2.3 Avaliação

Diferentemente da alta interação dos usuários proposta no conceito do aplicativo *Sustentaê*, como: comentários, curtidas, compartilhamentos, seguidores, perfil do local, número de visitas, os aplicativos Responsa e Mapa de Feiras Orgânicas possuem uma baixa interação dos usuários com a comunidade presente no aplicativo e também não possuem funcionalidades de avaliação dos locais, servindo mais como catálogos informativos.

Outro fator favorável ao conceito do aplicativo do *Sustentaê* é a adição de informações pelos próprios usuários. Dessa forma, novos estabelecimentos podem facilmente ser adicionandos pela comunidade no aplicativo diferente dos outros dois aplicativos em que há uma organização que adiciona essas informações.

Figura 4 – Telas do aplicativo Mapa de Feiras Orgânicas.

(a) Menu principal

(b) Mapa de Feiras

Fonte: Aplicativo Mapa de Feiras Orgânicas[4].

2.3 Engenharia de Software

2.3.1 Engenharia de Requisitos

Segundo Sommerville (2000), engenharia de requisitos é o processo de definir os serviços e restrições do sistema, que são necessitados pelo cliente, sob as quais ele opera e é desenvolvido. Essas restrições e serviços do sistema são os próprios requisitos gerados durante o processo. Eles podem variar de uma especificação matemática funcional à uma declaração abstrata de alto nível e podem ser divididos em dois tipos gerais, os funcionais e os não-funcionais.

Os requisitos funcionais são aqueles que se referem as funcionalidades esperadas que o sistema forneça, dependendo somente do conhecimento passado pelo cliente sobre o negócio em si. Já os não-funcionais estão ligados a padrões de qualidade como desempenho, robustez, manutenabilidade e não possuem relação direta com as funcionalidades do sistema (MACHADO, 2011).

Os processos utilizados para a engenharia de requisitos são muito variados, dependendo do domínio da aplicação, das pessoas envolvidas e da organização que desenvolve os requisitos. Entretanto, existem uma série de atividades genéricas comuns a todos os processos, sendo elas (SOMMERVILLE, 2000):

- Elicitação de requisitos

- Análise de requisitos
- Validação de requisitos
- Gerenciamento de requisitos

O autor ainda resume que engenharia de requisitos é um processo iterativo em que estas atividades são intercaladas.

2.3.2 Metodologia de Desenvolvimento

À medida que empreendimentos relacionados com tecnologia da informação se tornam cada vez maiores e mais complexos, metodologias de desenvolvimento de software, e suas ferramentas de suporte, tornam-se fatores importantes para alcançar êxito nos projetos (ASADI; RAMSIN, 2009). Devido ao contexto em que esses se encontram, isto é, a necessidade de um desenvolvimento mais ágil, com capacidade de adaptação, inovação, melhoria contínua e com recursos limitados, a área de gerencimento de projetos se fortaleceu para organizar e admisnistrar tais projetos com o auxilio das metodologias de desenvolvimento (MAURER, 2005).

As metodologias tradicionais de desenvolvimento não fornecem suporte suficiente para projetos de desenvolvimento de software mais modernos, visto que são consideradas metologias rígidas (ASADI; RAMSIN, 2009). Por outro lado, as metodologias ágeis de desenvolvimento tem como foco principal construir software de forma mais produtiva, enxergando os projetos de forma diferente das metodologias tradicionais e incentivando a utilização de princípios e práticas específicas (TAVARES, 2008).

2.3.2.1 *Scrum*

Scrum é uma aborgagem enxuta para oraganização e gerenciamento de projetos, dentro das metodologias ágeis, para desenvolvimento de produtos. De maneira geral, o *Scrum* é um processo ágil que tem como foco o desenvolvimento de produto ou gerenciamento de trabalhos iterativos e incrementais, podendo ser utilizado no desenvolvimento de qualquer produto (RISING, 2000).

O *Scrum* é oraganizado em equipes e dentro de cada equipe são definidos papéis, eventos, artefatos, regras e atuam em ciclos de incremento ao produto (*sprint*). Cada elemento dentro do *Scrum* tem um propósito específico. Entretanto, há a possibilidade de que sejam empregados outros processos ou técnicas externas ao *Scrum* a fim de melhora-lo e adapta-lo para um contexto específico (SCHWABER; SUTHERLAND, 2013).

A equipe do *Scrum* é caracterizada por ser auto-organizável e multifuncional. Ela é composta pelo *Product Owner*(Dono do Produto), o *Scrum Master* e a equipe de desenvolvimento em si. Cada um desses papéis possui responsabilidades definidas e distintas. O

Product Owner é a única pessoa responsável pelo gerenciamento do *backlog* do Produto, ele também tem a responsabilidade de maximizar o valor do produto e do trabalho da equipe de desenvolvimento. Já o *Scrum Master* tem o papel de garantir que o *Scrum* seja entendido e aplicado, aderindo a teorias, práticas e regras. Por fim, a equipe de desenvolvimento é composta pelos desenvolvedores do projeto que tem a responsabilidade de entregar uma versão usável que potencialmente incrementa o produto ao final de cada ciclo de incremento de produto (SCHWABER; SUTHERLAND, 2013).

2.3.3 Desenvolvimento para dispositivos móveis

O mercado de *smartphones* está em constante crescimento no mundo, assim como o de aplicações móveis. Nessa última década, houve um tremendo crescimento na área de desenvolvimento de aplicações para dispositivos móveis (BERKMAN-CHARDON et al., 2016). Aplicativos móveis consistem em arquivos binários executáveis que são carregados e armazenados diretamente no dispositivo levado pelo usuário. Eles são distribuídos por meio de lojas de aplicativos específicas, como a *Play Store* para sistemas *Android* e a *App Store* para *iOS* (MALAVOLTA, 2016).

2.3.3.1 Desenvolvimento nativo

Aplicativos nativos são desenvolvidos diretamente por serviços fornecidos pela plataforma móvel a quem pertencem. Esses serviços são prestados por meio de uma *API (Application Programming Interface)* com métodos relacionados a comunicação, localização, segurança, etc (QUE; GUO; ZHU, 2016).

Linguagens de programação e ferramentas para construir aplicações nativas são específicas de cada plataforma. No *Android*, os aplicativos são criados em *Java* ou *Kotlin* por meio do *Android SDK (Software Development Kit)*, utilizando por exemplo o *Android Studio* como ferramenta de desenvolvimento. Por outro lado, no *iOS* as linguagens utilizadas são o *Objective-C* ou *Swift* por meio da ferramenta *Xcode* (MALAVOLTA, 2016).

2.3.3.2 Desenvolvimento híbrido

As tecnologias consideradas padrões da *Web*, como *HTML5 (Hypertext Markup Language), CSS3 (Cascading Style Sheets), JavaScript*, podem ajudar na construção de aplicativos móveis em uma linha de conhecimento tecnológico geral. Os aplicativos construídos utilizando essas tecnologias são conhecidos como híbridos. Hoje, existem três estratégias principais para o desenvolvimento desses sistemas: *mobile web apps*, *mobile apps* híbridos baseados em *web* e *web apps progressivos* (FORTUNATO; BERNARDINO, 2018).

Primeiramente, os *mobile web apps* são desenvolvidas com tecnologias *web*, hospedados em servidores remotos, disponibilizados via protocolos padrão, como o *HTTP (Hypertext Transfer Protocol)*, acessados por meio de uma única *URL (Uniform Resource Locator)*. Basicamente, eles são *sites* otimizados para acesso usando navegadores instalados em dispositivos móveis. Já os *mobile apps* híbridos baseados em *web* são desenvolvidos por meio de tecnologias *web*, utilizando um *framework* híbrido para o desenvolvimento, como o *Apache Cordova*, e podem ser distribuídos para qualquer plataforma móvel (BESSGHAIER; SOUII, 2017).

Por fim, os *web apps progressivos* são tipos especiais de aplicativos híbridos no qual a melhora progressiva fornece suporte para baixa ou nenhuma conectividade com a *Internet*, suporte a notificações *push*. Assim como os *mobile web apps* eles podem ser acessados diretamente pelo navegador, porém eles também podem ser instalados no próprio dispositivo do usuário (FORTUNATO; BERNARDINO, 2018).

2.3.4 Padrões de Projeto

Segundo Gamma (1994), padrão de projeto, do inglês *Design Pattern*, é uma solução simples e elegante voltada a resolver um problema específico. Já segundo MARTIN (2000), padrão de projeto é uma solução bastante utilizada e bem conhecida para problemas comuns e é formada pelos seguintes elementos: nome, problema associado, solução e as consequências.

O nome do padrão normalmente é formado por uma ou duas palavras capazes de referenciar tanto o problema a ser tratado, quanto a sua solução. O problema associado é a descrição de onde se deve implementar o padrão de projeto. Já a solução se refere aos elementos que devem ser inseridos no projeto e as consequências dizem respeito aos resultados da implementação do padrão ao projeto (MARTIN, 2000).

2.3.5 Arquitetura de Software

A arquitetura de software apareceu nos anos 90 como uma importante subárea da Engenharia de Software. Atualmente, é reconhecida como fator crítico para o sucesso de qualquer sistema de software complexo. Antes disso a prática de arquitetura se baseava em projetos *ad-hoc*, sem padrões e de formar empírica. Na década de 2000, a área teve um amplo reconhecimento, surgindo taxonomias de padrões arquiteturais, técnicas para formalização da arquitetura, métodos de revisão e produtos que asseguravam a conformidade entre o projeto arquitetônico e a implementação (GARLAN; SHAW, 2011).

De forma geral, a arquitetura de software tem como objetivo estabelecer a estrutura geral de um sistema de software. Isto é, descreve aspectos críticos, trás um visão geral para os *stakeholders*, permite a avaliação do produto antes de ser desenvolvido, projeta

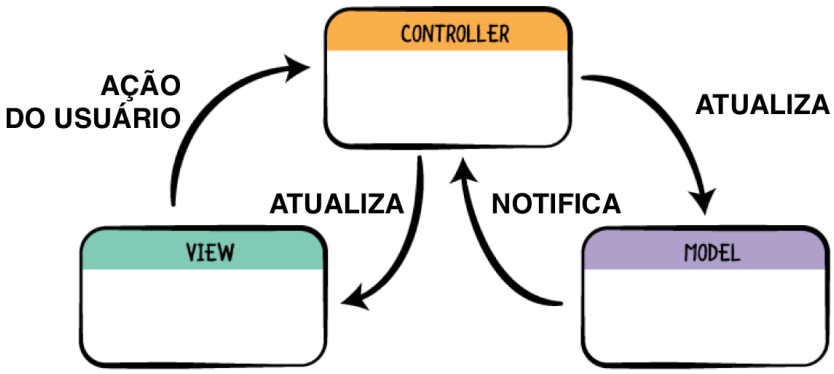
seus componentes e suas interações (SOMMERVILLE, 2000).

2.3.5.1 Model-View-Controller

O padrão MVC *(Model-View-Controller)* é comumente utilizado em aplicações web. Consiste em um padrão arquitetural organizado em três níveis, descritos abaixo (GUANGCHUN; LU; HANHONG, 2003):

- ***View***: nível que contém os elementos gráficos da aplicação, sendo a camanda mais próxima do usuário.
- ***Controller***: nível intermediário que gerencia as chamadas entre os níveis de *view* e *model*. Isso garante que não haverá ligação direta entre elementos de interface e entidades da camada de modelo, facilitando a manutenção e permitindo maior reutilização do software.
- ***Model***: Nível resposável pelas entidades relacionadas ao domínio cognitivo. Geralmente, esse nível faz acesso ao SGBD, que corresponde a real camada de persistência de um software.

Figura 5 – Modelo genérico da arquitetura MVC.



Fonte: (RAYWENDERLICH, 2016).

De formar geral, o MVC separa *views* e *models* estabelecendo um protocolo entre eles, conforme pode ser visto na Figura 5. Uma *view* deve garantir que sua aparência reflita o estado do *model* corretamente. Sempre que há mudanças de dados no *model*, este deve notificar as *views* que dependam dele. Em resposta, cada *view* tem a oportunidade de se atualizar. Essa abordagem permite anexar várias *views* a um *model* para fornecer apresentações diferentes (GUANGCHUN; LU; HANHONG, 2003).

2.3.6 *Web Service*

O *Web Service* (serviço da Web) é uma arquitetura de *software* para construção de programas de computador e ou aplicativos móveis de forma descentralizada se baseando no paradigma *Service Oriented Architecture* (Arquitetura Orientada a Serviço) (KUMAR; SUREKA, 2018). Isto é, a construção de serviços de *software* que podem funcionar isoladamente, mas que também podem interagir entre si. Dessa forma, pode-se construir pequenos serviços de *software* que possuem pouca ou apenas uma única responsabilidade e a integração desses serviços será o produto final (KUN, 2004).

Um exemplo desse tipo de arquitetura é a criação de um serviço cuja a resposabilidade é realizar determinado cálculo. Sendo assim, outros serviços podem interargir com o serviço de cálculo para obter o resultado da conta sem a necessidade de se saber como será realizada essa conta.

2.3.7 *API - Application Programming Interface*

API é um conjunto de pontos de entrada, símbolos, um conjuntos de parâmetros e especificações que permite a comunicação entre sistemas. API's são geralmente entendidas como as especificações de como utilizar uma biblioteca de classes, ou como acessar um determinado serviço para o fornecimento ou processamento de dados (BOONCHUAY, 2017).

Pode-se entender que um conjunto de classes e ou funções são a implementação das regras de uma API. Disponibilizando uma série de rotinas pré-definidas que devem prover soluções as necessidades de um contexto. Por exemplo: APIs para manipulação de *strings*, processamento de imagens, criação e manipulação de dados de um banco de dados. Possibilitando ao desenvolvedor escrever códigos melhores e mais modularizados (FIELDING, 2000).

2.3.7.1 *REST - Representational State Transfer*

Fielding (2000) descreveu as restrições e decisões de design por trás da World Wide Web como um padrão arquitetônico chamado *Representational State Transfer* (REST) que foi aplicado ao design do HTTP (FIELDING, 1999) e identificadores de recursos uniformes (URI) (BERNERS-LEE, 2005).

A abordagem REST visa apoiar sistema de hipermídia distribuídos, com interfaces gerais e componentes intermediários que reduzem a latência de interação (FIELDING, 2000). Algumas das partes fundamentais do REST quando aplicadas para HTTP e URIs são:

- Recursos são alvos de referências e são identificados por identificadores de recursos

na forma de URIs;

- Recursos podem ter diferentes representações (formatos), exemplos: HTML, XML (*Extensible Markup Language*), JSON (*JavaScript Object Notation*), texto simples, objetos Java serializados. As representações são enviadas no corpo do mensagem;

- Um conjunto predefinido de métodos (correspondente a verbos) que agem sobre os recursos. O método é fornecido no pedido do cliente. Métodos como OPTIONS e GET não devem levar as mudanças no lado do servidor, então os resultados desses métodos geralmente podem ser em *cache*. O método GET é usado para recuperar a representação do recurso que o URI identifica, que é como a maioria das páginas da web normais são buscadas. O método PUT substitui o conteúdo no URI de destino (ou cria-o se estiver ausente) e DELETE exclui recursos. O método POST é frequentemente usado para modificar ou adicionar informações, ou para executar outras operações determinadas pelo servidor;

- A resposta contém cabeçalhos, um código de status HTTP e possivelmente um corpo. Códigos de status indicam o resultado da requisição como por exemplo: sucesso (“200 OK”, “201 Criado”, etc.), redirecionamento (“301 Moved Permanently”, “303 Ver Outro”, “304 Not Modified”) e outros.

2.3.8 Testes Automatizados de Software

A produção de software é um processo que envolve diversas atividades e diferentes pessoas. Isto é, há a necessidade da denifinição de um escopo, de codificar, de verificar as funcionalidades entre outas. Cada etapa pode ser de responsabilidade de uma pessoa diferente, e assim, esse número de atividades e diferentes pessoas pode levar à introdução de defeitos ao produto final, resultado de entendimentos diferente entre os responsáveis do produto de software (TRODO, 2009).

Buscando mitigar os defeitos encontrados em produtos de software, a atividade de testes de software passou a fazer parte do processo de produção (BARBOSA, 2009). De forma geral, a atividade de teste de software é um trabalho de análise do código-fonte para verificar se condiz com o desejado. Isso significa que há a necessidade do funcionamento/existência do software para que esta atividade seja realizada. Nessa análise é possivel identificar defeitos e previnir o aparecimento de futuros problemas (BARBOSA, 2009).

Existem diferentes tipos de teste de software. Cada um é voltado para análisar o comportamento de diferentes aspectos do produto. São eles: testes de aceitação, testes de sistema, teste de integração e testes unitários (SOMMERVILLE, 2000).

A lista a seguir trás uma definição de cada tipo de teste apresentado acima (PEZZE, 2008).

- **Testes de aceitação**: é a última atividade realizada antes da distribuição do software. Visa verificar se o software está pronto e realiza o que foi planejado;
- **Testes de sistema**: verifica se o sistema está funcionando como um todo. Isto é, verifica se todos os componentes do produto estão operando de forma esperada em conjunto;
- **Testes de integração**: tem como objetivo verificar a integração entre dois componentes do sistema;
- **Testes unitário**: testa apenas um único componente do sistema, de maneira isolada, e simulando as prováveis informações/dependências que aquele componente tem.

2.3.9 Integração Contínua

A Integração Contínua ou CI (*Continuous integration*) é uma prática de desenvolvimento que exige que os desenvolvedores verifiquem a integridade do código em um repositório compartilhado várias vezes no decorrer do desenvolvimento do projeto. Cada verificação ocorre por meio de uma compilação automatizada do código e analisa-se os resultados dos testes automatizados, permitindo que as equipes detectem problemas antecipadamente (BECK, 2004).

Ao integrar-se regularmente pode-se detectar erros rapidamente e localizá-los com mais facilidade. Com isso, os impactos causados por erros encontrados apenas na etapa de implatação do software são reduzidos e os recursos destinados ao teste de software podem ser realocados para outra etapa da produção de software (BECK, 2004).

Geralmente, o processo de CI segundo Beck é composto pelas seguintes etapas:

- Compilar o código-fonte;
- Executar os testes automatizados;
- Verificar se a qualidade do código está dentro dos níveis desejados;
- Disponibilizar uma versão para instalação.

2.3.10 Entrega Contínua

A Entrega Contínua é a capacidade de entregar alterações de todos os tipos, incluindo novas funcionalidades, alterações de configuração, correções de *bugs* e experi-

mentos no código, ao ambiente de produção ou nas mãos dos usuários, com segurança e rapidez, de maneira sustentável (BECK, 2004).

As práticas no centro da entrega contínua ajudam as equipes de desenvolvimento a alcançar vários benefícios importantes, alguns deles estão na lista a seguir (HUMBLE, 2010):

- **Diminuição do risco**: o principal objetivo da entrega contínua é tornar as implantações de software simples, eventos de baixo risco que podem ser executados a qualquer momento, sob demanda;
- **Maior qualidade**: quando os desenvolvedores têm ferramentas automatizadas que descobrem regressões em minutos, as equipes são liberadas para concentrar seus esforços na pesquisa de usuários e nas atividades de teste. Ao construir um *pipeline* de implantação, essas atividades podem ser realizadas continuamente durante todo o processo de entrega, garantindo que a qualidade seja incorporada aos produtos e serviços desde o início;
- **Custos mais baixos** qualquer produto ou serviço de software bem-sucedido evoluirá significativamente ao longo de sua vida útil. Ao investir em automação de construção, teste, implantação e ambiente, se reduz substancialmente o custo de fazer e entregar mudanças incrementais no software, eliminando muitos dos custos fixos associados ao processo de disponibilização.

3 Suporte Tecnológico

Este capítulo aborda as principais ferramentas que foram utilizadas no desenvolvimento do software, bem como as utilizadas para o apoio da pesquisa. Com o objetivo de melhorar a organização, os recursos tecnológicos foram divididos em quatro categorias: Gerência do Projeto, Desenvolvimento de Software, Gerência de Configuração de Software e Desenvolvimento do TCC.

3.1 Gerência do Projeto

As ferramentas a seguir foram utilizadas no âmbito do gerenciamento de projeto.

3.1.1 *Zoho Sprints*

É uma ferramenta *on-line*, com inúmeras funções, voltada para o planejamento e monitoramento de equipes ágeis. Ela possui um painel central que fornece diversas informações sobre o andamento geral do projeto, como: porcentagem de conclusão geral (contagem de itens em aberto no *backlog* e itens já concluídos), contagem de *sprints* totais, futuras, ativas e concluídas, *status* de produtividade dos usuários, histórico de atualizações do *backlog* e ainda um gráfico "Real x Planejado" de todo o projeto. Ela possibilita a criação de itens para o *backlog*, classificando-os em tarefas, *bugs* ou histórias de usuário, adicionando prioridade (nenhuma, baixa, média ou alta), pontos de estimativa de execução da atividade, sendo possível também atribuir usuários responsáveis, anexar arquivos referentes aos itens ou ainda adicionar a tarefa como dependente de outra tarefa (SPRINTS, 2018).

Outra funcionalidade interessante dessa ferramenta é a possibilidade de realizar o planejamento das *sprints* de todo o projeto, criando previamente as *sprints* com datas de início e de fim, além de pré-adicionar itens do *backlog* a uma *sprint* futura planejada. A ferramenta ainda oferece um acompanhamento completo da *sprint* atual e das *sprints* concluídas, fornecendo dados como: *burndown*, *burnup*, *velocity*, fluxo cumulativo e um *kanban* das atividades que estiverem atribuídas a *sprint* atual (SPRINTS, 2018).

Por fim, a ferramenta ainda fornece a funcionalidade de agendar e documentar reuniões realizadas no *Scrum*, como: planejamento da *sprint*, *daily*, revisão da *sprint*, retrospectiva. Assim sendo, o *Zoho Sprints* foi utilizado para o gerenciamento do projeto como um todo, centralizando as atividades a serem executadas, descrição, prazos, requisitos, documentação das reuniões e demais relatórios relacionados.

3.1.2 *Slack*

É um software focado na comunicação para o trabalho realizado em equipe. Ele basicamente simplifica e organiza a comunicação, por meio da criação de canais específicos para assuntos distintos, onde podem ser anexados arquivos, marcações de usuários e integrações com outras aplicações. Um exemplo dessa integração é com repositórios remotos no *GitHub, GitLab* ou *Bitbucket* que ao realizar uma atualização no repositório o *Slack* pode notificar os usuários de um canal específico que essa atualização foi feita (SLACK, 2018).

Esse software foi utilizado para facilitar a comunicação da equipe, criando canais dedicados a escrita do TCC, a integração dos reposítorios de desenvolvimento e de integração contínua.

3.1.3 Bonita BPM

A Bonita é uma ferramenta gratuita de código-aberto que possui suporte para *macOS* sendo usada para modelagem de processos com a notação BPMN (*Business Process Modeling Notation*). Essa ferramenta foi utilizada para realizar a modelagem dos processos presentes nesse TCC (BONITA, 2018).

3.2 Desenvolvimento de Software

No que tange à área de desenvolvimento de software, as seguintes ferramentas foram utilizadas:

3.2.1 *Xcode*

O *Xcode* é um *IDE* (Ambiente de desenvolvimento integrado) para o *macOS* que possui um conjunto de ferramentas de desenvolvimento de software desenvolvido pela empresa *Apple* para o desenvolvimento de software para *macOS, iOS, watchOS e tvOS*. A ferramenta está disponível gratuitamente por meio da *Mac App Store* para *macOS* (APPLE, 2018).

Dentre as ferramentas do *Xcode* existem dois *frameworks* voltados à escrita e execução de testes de software, o *XCTest* e o *XCUITest*. O *XCTest* engloba os testes unitários enquanto o *XCUITest* os testes de integração. O *Xcode* foi utilizado para o desenvolvimento móvel voltado ao *iOS* nesse trabalho.

3.2.2 *Parse Platform*

O *Parse* é uma plataforma de auxílio para construção de um *backend mobile* como serviço. Originalmente foi desenvolvida pelo provedor da *Parse, Inc.* A empresa porém, foi adquirida pelo *Facebook* em 2013 e fechou em janeiro de 2017. Entretanto, no anúncio em 2016 do fechamento iminente, a plataforma foi disponibilizada e transformada em *open source* (código-fonte aberto). Atualmente, a plataforma principal e suas várias extensões, são mantidas em repositórios abertos no *GitHub* (PARSE, 2018).

Foram utilizados os módulos *parse-dashboard* e o módulo *parse-server* da plataforma para criação e manutenção de um serviço *RESTful* para o desenvolvimento do software, sendo utilizado como *Web Service*.

3.2.3 *Jest*

O *Jest* é um *framework* de testes unitários para projetos escritos na linguagem *JS* (*JavaScript*) e é mantido por uma comunidade de contribuidores de código aberto e por funcionários do *Facebook*. Uma das funcionalidades mais interessantes do *Jest* é o *feedback* instantâneo, que ao modificar algum arquivo que já possui testes unitários relacionados, executa esses testes de forma instantânea, fornecendo um relatório imediato acerca das modificações feitas (JEST, 2018). Esse *framework* foi utilizado para auxiliar no processo de testes de software do servidor.

3.2.4 *MongoDB*

O *MongoDB* é um banco de dados *open-source* multi-plataforma orientado a documentos. Ele é classificado como um banco de dados *NoSQL* que utiliza *JSON* como documentos com esquemas (MONGODB, 2018). Esse banco de dados é utilizado pela plataforma do *Parse* para armazenamento dos arquivos.

3.2.5 *Amazon S3*

O *Amazon S3 (Simple Storage Service)* é um serviço de armazenamento de objetos criado para armazenar e recuperar qualquer quantidade de dados de qualquer local: *sites*, aplicativos móveis, aplicativos corporativos e até dados de sensores. O serviço é oferecido pela *Amazon Web Services (AWS)* e possui integração com outros serviços da própria *AWS* (AMAZON, 2018). O *Amazon S3* foi utilizado para armazenamento de imagens e integrado a plataforma do *Parse*.

3.3 Gerência de Configuração de Software

Afim de fornecer suporte ao desenvolvimento, visando automatizar o máximo de tarefas possíveis, as seguintes ferramentas foram utilizadas:

3.3.1 *Git*

O *Git* é um sistema de controle de versão livre e de código aberto construído para lidar com pequenos ou grandes projetos com eficiência e rapidez (GIT, 2018). O *Git* foi utilizado para o versionamento e controle de versões do código-fonte e da monografia desse trabalho.

3.3.2 *GitLab*

O *GitLab* é um serviço *web*, de código-fonte aberto e de hospedagem de repositórios *Git*. Ele fornece diversas funcionalidades, como: gerenciamento dos repositórios, uso de *wiki*, revisão de código, gerenciamento de *pull-requests* e controle de permissão de *branchs* (GITLAB, 2018). Esse serviço foi utilizado para hospedagem do código-fonte de todo o software e do texto do TCC.

3.3.3 *Fastlane*

O *Fastlane* é uma plataforma *open source* que tem como objetivo simplificar a implantação de aplicativos para *iOS e Android*. Ela realiza a conexão direta com as lojas de distribuição das plataformas, *App Store e Play Store*, a fim de disponibilizar diferentes *deploys* das versões do aplicativo, seja em produção, ou em modo restrito para testes (FASTLANE, 2018). Essa ferramenta foi utilizada para automatizar a entrega contínua das versões do aplicativo.

3.3.4 *Bitrise*

O *Bitrise* é uma ferramenta *open source* focada na entrega e integração contínua voltada ao desenvolvimento móvel, que possui diversas integrações com outras ferramentas do meio. Ela automatiza o processo de compilação e disponibilização do aplicativo por meio de "fluxos de tarefas"customizáveis e executa esses fluxos em máquinas virtuais hospedadas em servidores próprios (BITRISE, 2018). O *Bitrise* foi utilizado junto ao *Fastlane* para automatizar o processo de integração contínua do aplicativo que foi construído.

3.3.5 *Heroku*

O *Heroku* é uma plataforma de nuvem com um serviço (*PaaS*) que suporta várias linguagens de programação, como: *Java, NodeJS, Python, PHP, Go*. A plataforma permite

criar, gerenciar, executar e escalar aplicações *web*, simplificando a arquitetura e o *deploy* dessas aplicações (HEROKU, 2018). Essa ferramenta foi utilizada para hospedagem do serviço *RESTful* do *Parse Platform*.

3.3.6 *TestFlight*

O *TestFlight* é um serviço *on-line* que permite a instalação e testes de aplicações móveis para a plataforma *iOS*, por meio da nuvem, possibilitando que os desenvolvedores recebam os *logs* remotos, relatórios de erros e *feedback* dos testadores. O serviço viabiliza até 25 testadores internos, com até 10 dispositivos cada e 10.000 externos que podem baixar e testar a compilação do aplicativo. Os testadores podem ser agrupados para recebrem diferentes compilações do aplicativo (TESTFLIGHT, 2018). Esse serviço foi utilizado como forma de distribuição do aplicativo na metodologia de validação do produto com usuários em ambiente controlado.

3.4 Desenvolvimento do TCC

As ferramentas a seguir foram utilizadas para auxiliar na dissertação do trabalho de conclusão de curso (TCC).

3.4.1 *LaTeX*

O *LaTeX* é um sistema de preparação de documentos. Ele possui recursos projetados para facilitar a produção e a formatação de textos técnicos ou científicos. O sistema é mantido pela *LaTeX3 Project* e está disponível como software livre (LATEX, 2018). O *LaTeX* foi utilizado para dar suporte à escrita e formatação deste trabalho de conclusão de curso.

3.4.2 *Atom*

Atom é um editor de texto e de código-fonte disponível para *macOS, Linux e Microsoft Windows* com suporte para *plugins* feitos em *NodeJS*. Desenvolvido pelo *GitHub*, ele é integrado ao controle de versão *Git*, além de também poder ser utilizado como *IDE* (ATOM, 2018). O editor foi utilizado para escrita do TCC na linguagem *TeX*, com o suporte do *LaTeX*.

3.4.3 *Zotero*

O *Zotero* é um software *open-source* de gerencimanto de dados bibliográficos e materiais relacionados à pesquisa. Dentre as suas principais funcionalidades estão a integração com navegadores da *web* e a geração e exportação do relatório de referências, sendo

feito em formato escolhido pelo usuário (ZOTERO, 2018). Essa ferramenta foi utilizada para gerenciamento e exportação da bibliografia deste trabalho.

4 Metodologia

4.1 Metodologias de Pesquisa

Pesquisar pode ser definido como a forma de encontrar respostas para perguntas não compreendidas. Segundo a filósofa Minayo (1993), o termo representa um processo das ciências que buscam aproximar o desconhecido da realidade, uma junção entre teoria e dados.

Analisando a pesquisa de forma mais prática, tem-se um processo formal e estruturado de desenvolvimento do método científico. Tendo como seu principal foco a solução de problemas que se utilizam de atividades científicas (GIL, 2002).

Pesquisa é um aglomerado de atividades e ideias fundamentadas em mecanismos racionais e sistemáticos, sendo realizada quando não se há soluções para problemas ou fatos desconhecidos com o intuito de esclarecê-los e torná-los conhecidos (MORESI, 2003).

4.2 Classificações de pesquisas

Exitem diversas classificações de pesquisas devido ao fato que cada pesquisa visa um determinado objetivo específico e utiliza-se de diferentes técnicas. Isto é, uma tem como objetivo analisar o conhecimento atual sobre determinado assunto, outra visa estabelecer um modelo que explique o comportamento de determinado processo natural, dentre outros (MORESI, 2003).

A seguir serão expostos as principais classificações de pesquisa do ponto de vista de sua natureza, forma e objetivo.

4.2.0.1 Natureza da pesquisa

- **Pesquisa Básica**: tem como objetivo desenvolver novos conhecimentos sem a necessidade de aplicação prática (PRODANOV; FREITAS, 2013);
- **Pesquisa Aplicada**: tem como objetivo desenvolver conhecimentos para aplicação prática direcionadas à problemas específicos (RUIZ, 1996).

4.2.0.2 Formas da pesquisa

- **Pesquisa Quantitativa**: parte do pressuposto que tudo pode ser medido/quantificável. Tenta, por meio de dígitos, expor o conhecimento adquirido para estabelecer

um significado após uma análise (PRODANOV; FREITAS, 2013);

- **Pesquisa Qualitativa**: parte do pressuposto que nem sempre há uma relação direta entre o mundo real e o desconhecido, e que a subjetividade do mundo não pode ser traduzida em números. Não necessita de métodos ou técnicas da matemática para trazer significado aos fenômenos desconhecidos, mas por meio das observações do pesquisador e sua interpretação para estabelecer o significado (RODRIGUES, 2007).

4.2.0.3 Objetivo da pesquisa

- **Pesquisa exploratória**: geralmente é feita em áreas na qual não se há muito conhecimento consolidado e estruturado, buscando deixá-lo mais compreensível e dessa forma também serve para hipóteses (RUIZ, 1996);

- **Pesquisa descritiva**: tem como base as características do objeto de estudo. Utiliza-se também de relacionamento entre variáveis e a natureza do objeto de estudo. Contudo, não tem obrigação de explicá-lo ou defini-lo. Exemplo: pesquisa de opinião (PRODANOV; FREITAS, 2013);

- **Pesquisa explicativa**: tem como objetivo esclarecer quais condições/elementos contribuem, de alguma forma, para a ocorrência de um certo fenômeno. Exemplo: as ações bases para o sucesso de determinado empreendimento. Tem como base a pesquisa descritiva para suas explicações (RUIZ, 1996);

- **Pesquisa experimental**: é realizada de forma empírica, em que o pesquisador altera e gerencia as variáveis observando quais os impactos que essas variações causam. Ela permite analisar um determinado objeto de estudo em um ambiente específico (RODRIGUES, 2007).

4.3 Aplicação da metodologia

4.3.1 Planejamento da pesquisa

Do ponto de vista da **natureza da pesquisa** foi empregada a pesquisa aplicada, uma vez que o foco do projeto foi o desenvolvimento do aplicativo *Sustentaê* que possui um público alvo definido.

Do **objetivo da pesquisa** foi empregada a pesquisa exploratória, no qual o objetivo foi adquirir conhecimento na área foco do aplicativo (publico alvo, relevância, utilidade), bem como aprimorar os conhecimentos no que diz respeito as atividades e tecnologias utilizadas para o desenvolvimento do software.

Quanto a **abordagem da pesquisa** foi empregado os métodos de pesquisa qualitativa e quantitativa (quanti-quali), uma vez que o projeto teve como objetivo desenvolver uma ferramenta capaz de coletar e analisar a opinião pública acerca de estabelecimentos e o seu grau de sustentabilidade.

4.3.1.1 Atividades

Baseando-se na metodologia adotada para a realização deste trabalho foi elaborada uma sequência de atividades que organizaram os envolvidos na execução do projeto.

Na primeira etapa foi realizado um estudo preliminar sobre o tema abordado e o estudo de viabilidade de sua execução, uma vez que o projeto é a continuação de outro trabalho, foi necessário verificar a disponibilidade do tema com a autora. Nessa mesma etapa foi elaborado um cronograma geral de atividades para organização do projeto. Na etapa seguinte foi definido um escopo inicial, os objetivos do trabalho e a sua relevância. Em seguida, foram definidas as bases de artigos bem como o tipo de pesquisa a ser realizada. Uma vez definidos, deu-se início a escrita do referencial teórico e suporte tecnológico para o desenvolvimento do aplicativo.

Na próxima etapa foi construído e planejado o desenvolvimento de software, isto é, foi definida a arquitetura que foi utilizada bem como o processo de desenvolvimento e boas práticas. Por fim, na última etapa foi executada a metodologia de validação do produto e os seus resultados foram analisados. A atividade de elaboração da monografia foi realizada em paralelo durante todas as outras atividades necessárias neste trabalho.

4.3.1.2 Cronograma de atividades

A Tabela 1 mostra o cronograma das atividades gerais executadas na segunda etapa deste trabalho (TCC 2).

Tabela 1 – Cronograma de atividades realizadas no TCC 2.

Atividades	Agosto	Setembro	Outubro	Novembro	Dezembro
Realizar correções solicitadas pela banca	X				
Realização de *sprints* de desenvolvimento	X	X	X		
Testar evolução do produto		X	X		
Realização da validação do produto			X		
Analisar Resultados Obtidos			X	X	
Atualizar Monografia				X	
Criar e Realizar Apresentação					X

A Figura 28 presente no apendice B recorda o fluxo de atividades executado no TCC 1 e a Tabela 2 mostra o cronograma geral das atividades realizadas no TCC 1.

Tabela 2 – Cronograma geral de atividades do TCC 1.

Atividade	Março	Abril	Maio	Junho
Escrever TCC	X	X	X	X
Estudo Preliminar	X			
Definição de Escopo, Objetivos e Relevância	X	X		
Definição das Bases de Pesquisa		X		
Classificação da Pesquisa		X		
Suporte Tecnológico		X	X	
Referencial Teórico		X	X	
Definição da Proposta			X	X
Propor prova de conceito			X	X
Conclusão e Revisão				X
Definir Cronograma da segunda etapa do projeto				X

4.3.2 Desenvolvimento do aplicativo *Sustentaê*

No contexto de desenvolvimento de software, foi utilizado uma adaptação do *Scrum* e *eXtreme Programming*. Foram aplicadas práticas como folha de estilo de código, *backlog*, *planning poker*, retrospectivas de *sprint* e quadro de atividades (*Kanban*) (SCHWABER; SUTHERLAND, 2013).

4.3.2.1 Papéis

A equipe de desenvolvimento foi composta por duas pessoas, sendo necessário aplicar alguns ajustes nas práticas e papéis do *Scrum*. Assim, foi definido que o papel de *Scrum master* não seria desempenhado por um integrante específico, mas que as suas responsabilidades seriam diluídas entre os membros da equipe de desenvolvimento. O papel de *Product Owner* foi desempenhado pela autora do conceito do aplicativo *Sustentaê*.

A Tabela 3 mostra as responsabilidades que cada papel desempenhou dentro do processo de desenvolvimento de software.

Tabela 3 – Responsabilidade e Responsável.

Papel	Responsabilidade	Responsável
Product Owner	Trabalhou em conjunto com a equipe de desenvolvimento na definição e priorização das tarefas necessárias para realização do projeto.	Vanessa Raulino
Scrum Master	Responsável por zelar pelas boas práticas do método *Scrum*, isto é, auxiliou na realização dos ritos, e definição das atividades de cada *sprint*.	Equipe de Desenvolvimento

4.3.2.2 *Sprints* e Ritos

As *sprints* tiveram duração de duas semanas (quatorze dias). Dentro de cada *sprint* realizou-se os seguintes ritos:

- **Planejamento de *Sprint***: atividade que definiu quais tarefas foram selecionadas para realização na *sprint* atual;
- **Revisão da *Sprint***: atividade em que revisou as taferas realizadas na *sprint* estando elas finalizadas ou não. Dessa forma, todos os *stakeholders* possuíram uma visão do avanço do projeto;
- **Retrospectiva**: atividade em que se debateu acerca dos pontos negativos, positivos e eventuais mudanças para melhoria do processo de desenvolvimento dentro de cada *sprint*;

4.4 Metodologia de validação

Para a validação do produto final gerado por esse trabalho foi proposto a realização de testes em ambiente controlado. Isto é, o produto foi distribuído para um grupo de usuários que pode utilizá-lo durante um período de tempo.

Para a distribuição controlada do produto foi utilizada a ferramenta *TestFlight* descrita na seção 3.3.6 de suporte tecnológico . Dessa forma, foi possível distribuí-lo para um número controlado de pessoas, além de ser possível a seleção dessas pessoas. Para essa validação buscou-se avaliar os aspectos de usabilidade do aplicativo. A definição do roteiro e o que foi testado está descrita no Capítulo 5 desse trabalho.

4.4.1 Número de participantes

Segundo Nielsen (2000), o número de usuários necessários para se descobrir as falhas de usabilidade é de 15, sendo que os testes devem ser realizados com três grupos de 5. Dessa forma, os testes de usabilidade que foram realizados no aplicativo *Sustentaê* tiveram 15 pessoas divididas em três grupos de 5.

5 Sustentaê

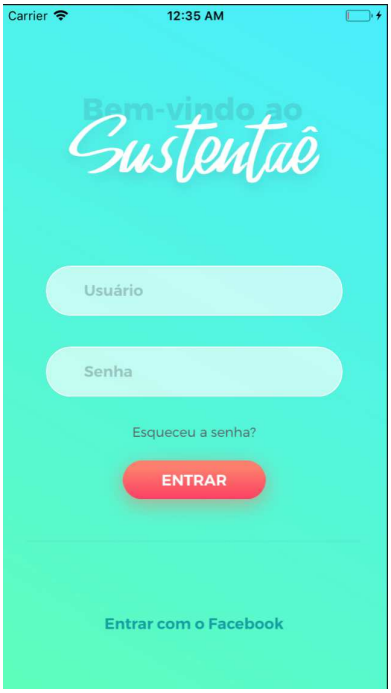
Decidiu-se por desenvolver o conceito do aplicativo criado por Vanessa Raulino em seu trabalho de conclusão de curso (RAULINO, 2017). Este, de forma geral, faz a integração entre publicações de estabelecimentos sustentáveis e pessoas com interesse nesse tipo de consumo. A ideia do conceito é fazer a integração através do formato de rede social, utilizando-se de conceitos como o de seguidores, postagens e "gostei", uma vez que esse conceito de aplicativo se encaixa com o objetivo geral descrito na seção 1.3.1.

Na seção 5.1, o aplicativo Sustentaê será detalhado colocando em evidência as principais funcionalidades e explicando, de maneira geral as tecnologias utilizadas para a criação de cada parte do produto. A seção 5.2 trás a organização utilizada para o desenvolvimento, isto é, os requisitos e os processos. A seção 5.3 mostra como foi desenhado o aplicativo do ponto de vista da arquitetura de software. E por fim a seção 5.5 trás os resultados da execução dos testes de usabilidade, também proposto como objetivo desse trabalho.

5.1 O Aplicativo

5.1.1 Login

Figura 6 – Tela de Login.

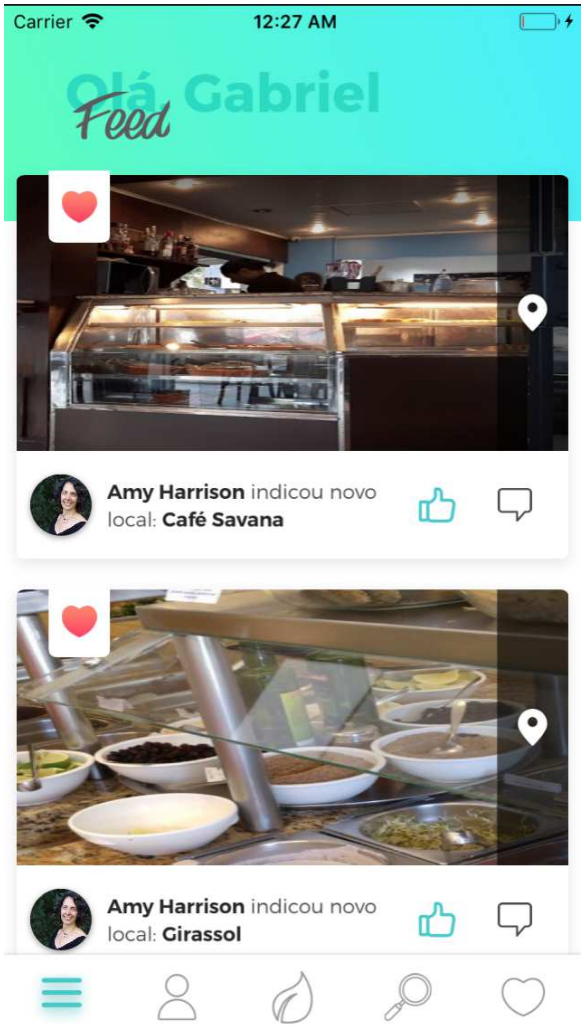


Fonte: Elaborada pelos autores.

A Figura 6 mostra a primeira tela do aplicativo Sustentaê. Uma vez que o aplicativo é uma rede social para divulgação de locais sustentáveis, há a necessidade de se autenticar para utilizar as suas funcionalidades. A forma selecionada para autenticação nesse trabalho foi a autenticação através do *Facebook*. Portanto, ao tocar no botão "Entrar com Facebook" o aplicativo requisita à API do *Facebook* que autentica o usuário em sua rede social e retorne com os dados de seu perfil. Caso seja o primeiro acesso o aplicativo Sustentaê cadastra esses dados como um novo perfil e o autentica. Caso não seja o primeiro acesso, há apenas a autenticação.

5.1.2 Feed

Figura 7 – *Feed* de achados.



Fonte: Elaborada pelos autores.

A tela de *feed*, Figura 7, é a tela principal do aplicativo. Os estabelecimentos sustentáveis indicados pelos perfis que o usuário segue são denominados "achados" e aparecem nessa tela na ordem em que foram compartilhados. Ainda nessa tela é possível

visualizar imagens de cada achado, comentar, curtir e favoritar. Ao selecionar um achado, o aplicativo navegará para o detalhe desse achado (seção 5.1.3).

A diferença entre favoritar e curtir é que a ação de curtir representa uma manifestação pública à outros usuários do seu interesse nesse achado, ou seja, essa informação é visível dentro do aplicativo. Já a ação de favoritar é uma maneira de guardar esse achado em uma lista pessoal de cada perfil para visualizações posteriores.

5.1.3 Detalhe de um achado

Figura 8 – Detalhe de um achado.

Fonte: Elaborada pelos autores.

Na tela da Figura 8 é onde se encontra os detalhes do achado. Isto é, a descrição completa do estabelecimento escrita pelo perfil que o compartilhou, fotos, a localização, número total de curtidas e a nota “sustentei”. Essa nota representa o nível de sustentabilidade do “achado” segundo o perfil que o publicou, em que 1 é a nota mínima e 5 a máxima.

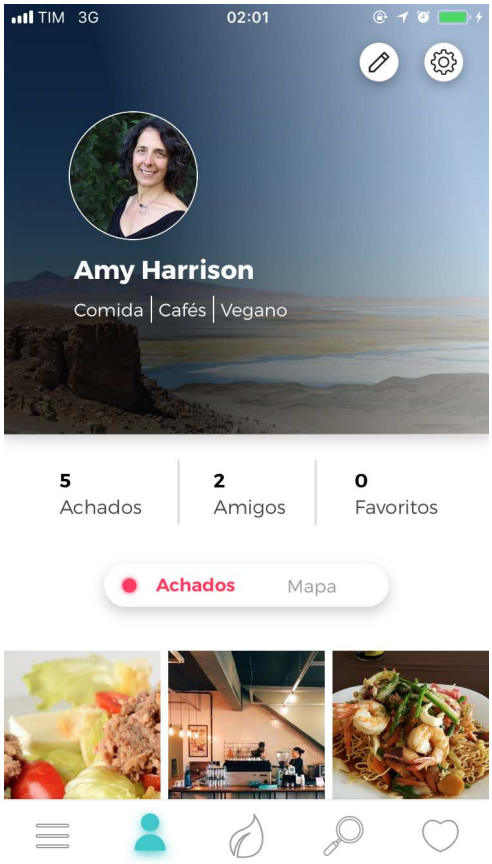
Também nessa tela é possível abrir o mapa pré-carregado com as informações de como chegar ao estabelecimento do achado a partir do endereço atual. É possível também comentar, favoritar e curtir o achado.

Há ainda a opção de visualizar mais informações do perfil que compartilhou o achado, tocando no nome ou na foto do perfil como descrito na seção 5.1.4, como também visualizar todos os achados desse estabelecimento, tocando no nome do estabelecimento conforme descrito na seção 5.1.5.

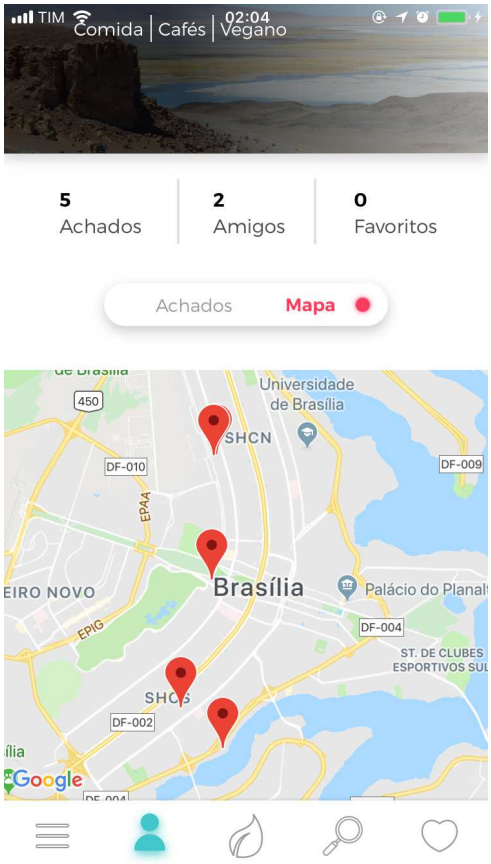
5.1.4 Perfil

Figura 9 – Telas do perfil do usuário.

(a) Perfil - fotos



(b) Perfil - mapa



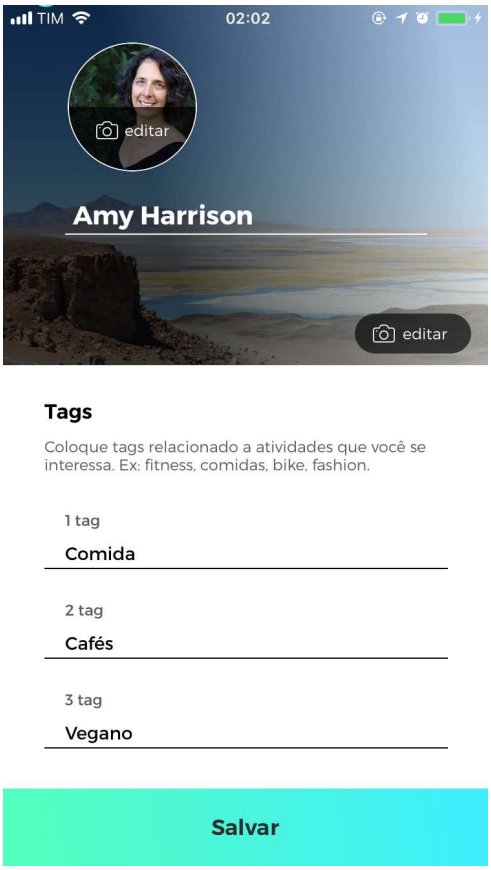
Fonte: Elaborada pelos autores.

Na tela de perfil, é onde se visualiza todos os achados de um perfil. Estes podendo ser exibidos em forma de lista ou distribuídos pelo mapa, conforme a Figura 9 a e b. Ao tocar em um achado, via mapa ou lista, o aplicativo direciona o usuário à tela de detalhe do achado descrita na seção 5.1.3. Outras informações disponíveis nessa tela são: nome, fotos de perfil e capa, número de achados publicados, número de amigos, número de favoritos e as *tags* (palavras-chave que representam os principais interesses desse perfil).

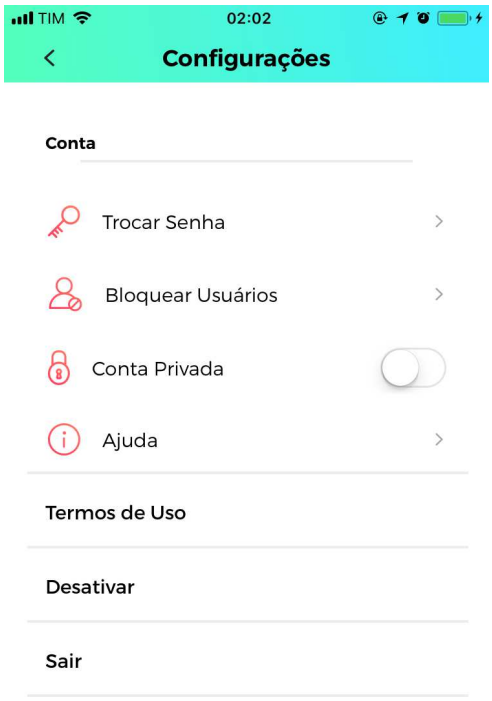
Caso o perfil visualizado seja o perfil do usuário autenticado no aplicativo, as opções para edição e configurações estarão disponíveis na parte superior da tela conforme a Figura 9 a. Na Figura 10 a, é onde altera-se as fotos de perfil e capa, nome e *tags* do usuário (edição), já na Figura 10 b modifica-se algumas configurações em gerais da conta.

Figura 10 – Telas de edição e configuração.

(a) Edição



(b) Configuração



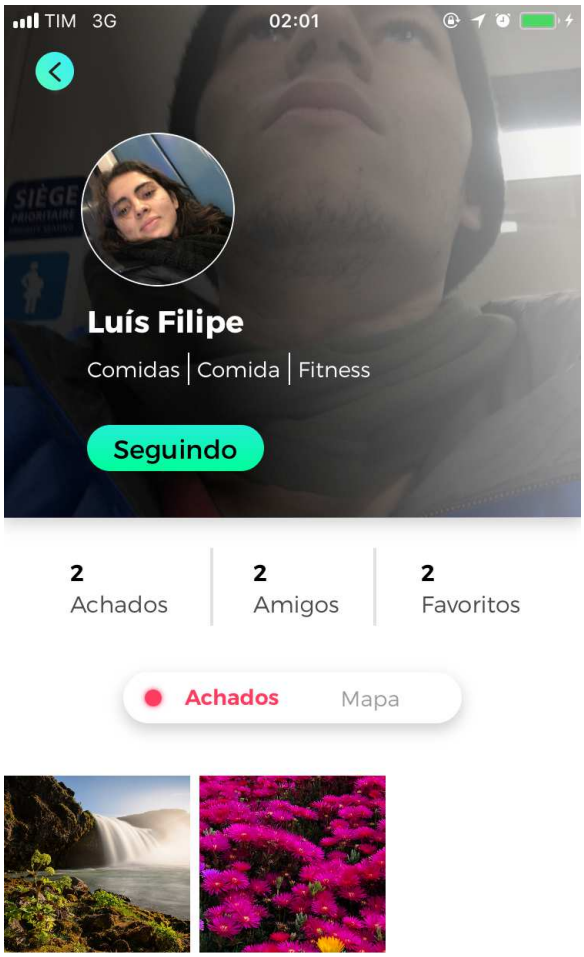
Fonte: Elaborada pelos autores.

Caso o perfil visualizado não seja o perfil autenticado, as opções de configuração e edição não aparecem e a opção de seguir ou deixar de seguir aparecerá, conforme mostrado na Figura 11.

5.1.5 Perfil estabelecimento

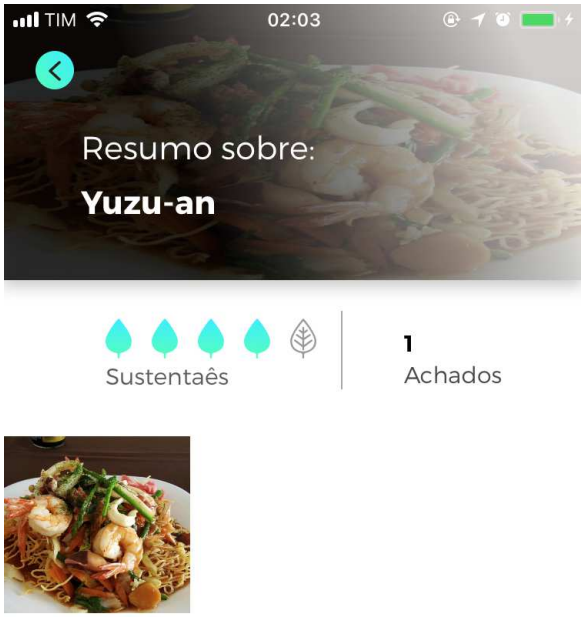
A tela de perfil de um estabelecimento é composto por todos os achados publicados no aplicativo referentes à esse estabelecimento. A foto pricipal e o nome são referentes ao primeiro achado publicado e a nota sustentei é a média das notas de todas as publicações. A tela pode ser vista na Figura 12.

Figura 11 – Perfil amigo.



Fonte: Elaborada pelos autores.

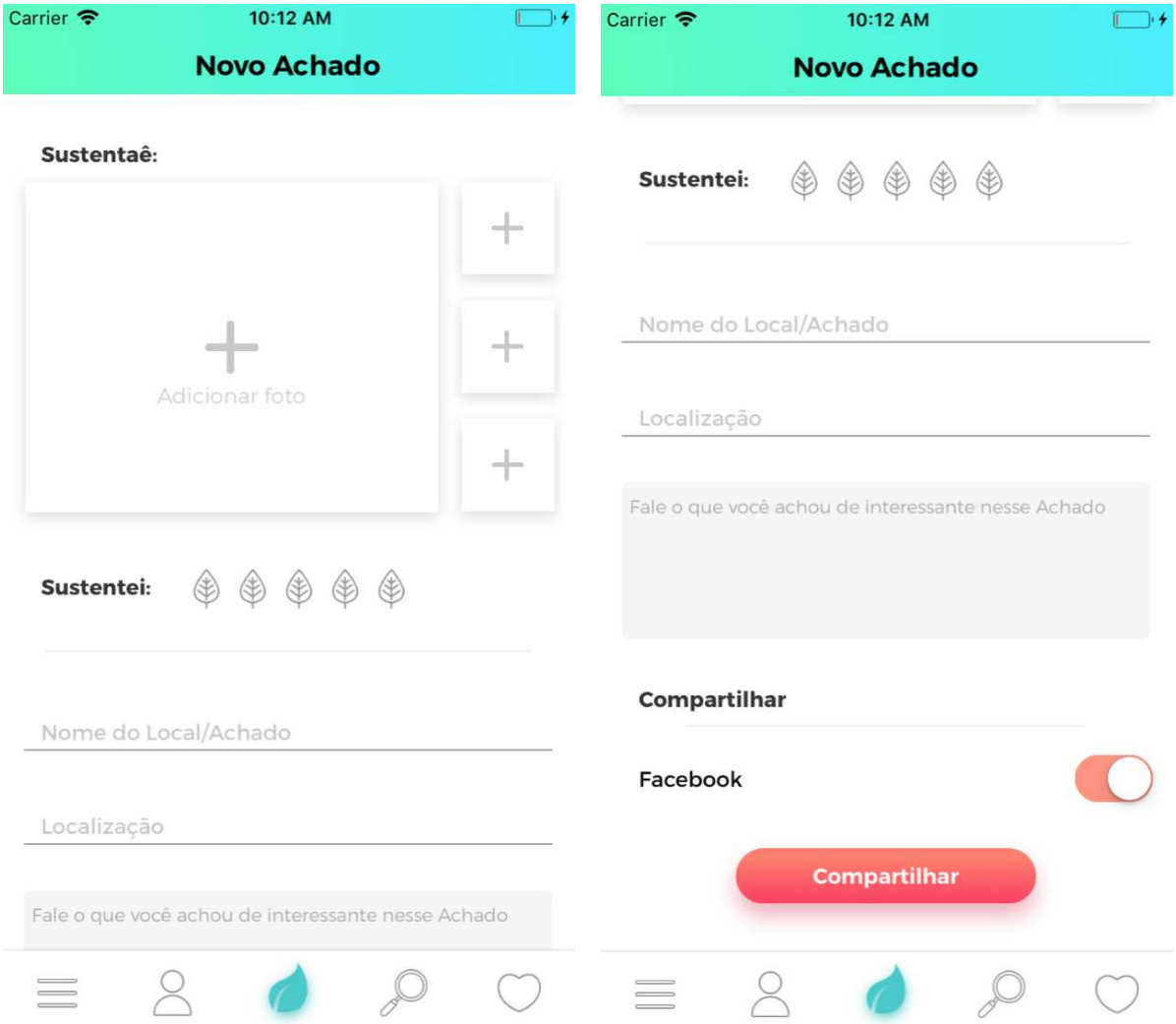
Figura 12 – Perfil do estabelecimento.



5.1.6 Publicação de novo achado

Figura 13 – Telas de criação de novo achado.

(a) Início do Formulário (b) Final do Formulário

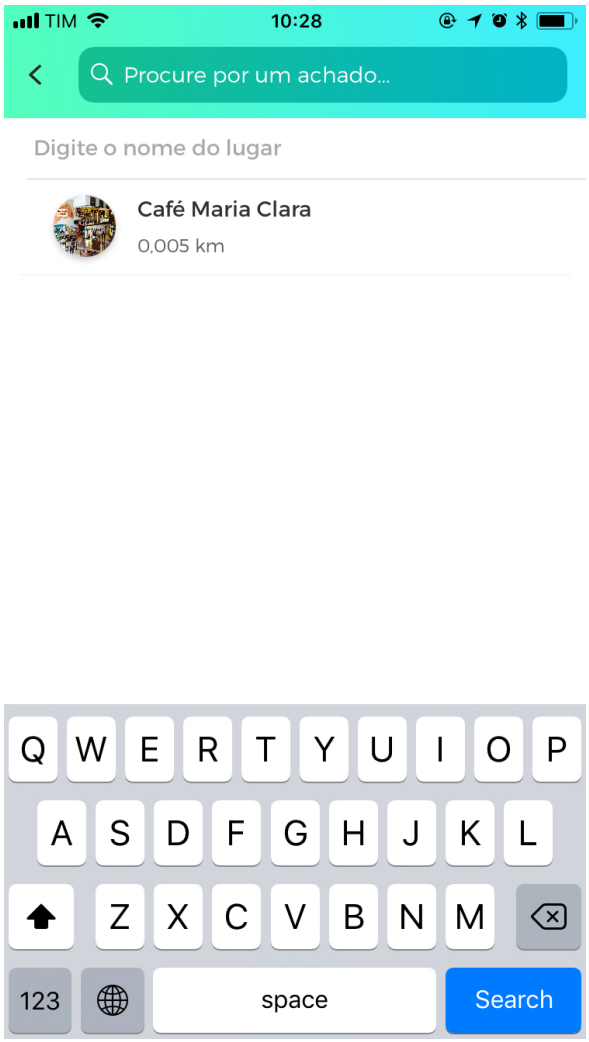


Fonte: Elaborada pelos autores.

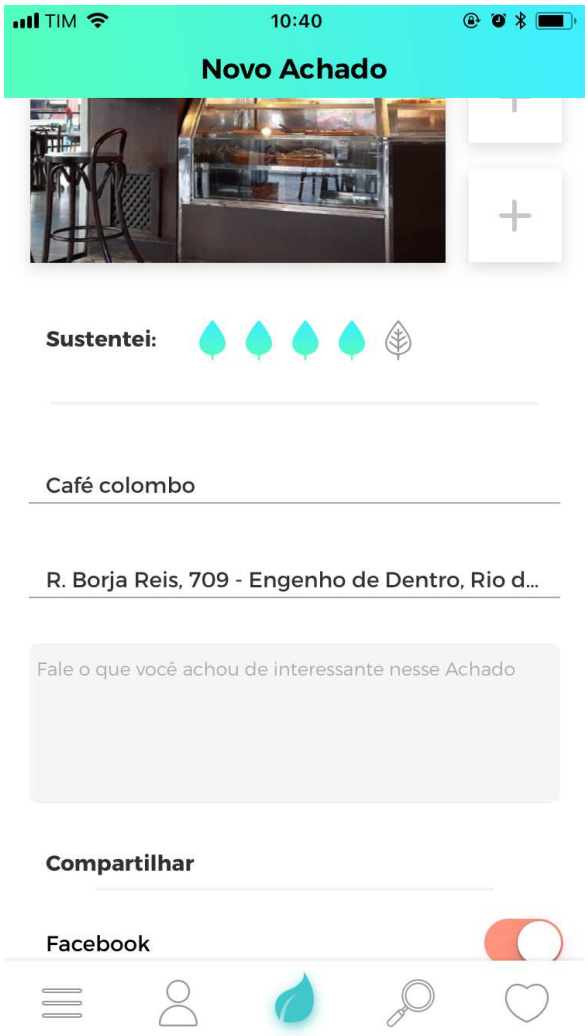
A tela de publicação de novo achado, Figura 13 a e b, é composta pela seção de *upload* de fotos, da nota sustentei, um campo para a descrição do estabelecimento, nome e endereço. Contudo o campo de nome e endereço não são preenchidos de maneira direta, pois existe uma necessidade de relacionar achados de um mesmo estabelecimento. Dessa forma, ao selecionar o campo nome, uma nova tela é aberta com a lista de estabelecimentos próximos à localização atual já cadastrados no aplicativo. Caso o local que se deseja publicar já exista, deve-se selecionar este estabelecimento, e caso não exista, há a opção de criar esse novo estabelecimento. Ao selecionar um estabelecimento pré-cadastrado ou ao criar um novo, o aplicativo utiliza a localização para preencher o endereço automaticamente como pode ser visto na Figura 14 a e b.

Figura 14 – Seleção de estabelecimento.

(a) Lista de estabelecimentos



(b) Endereço preenchido automaticamente



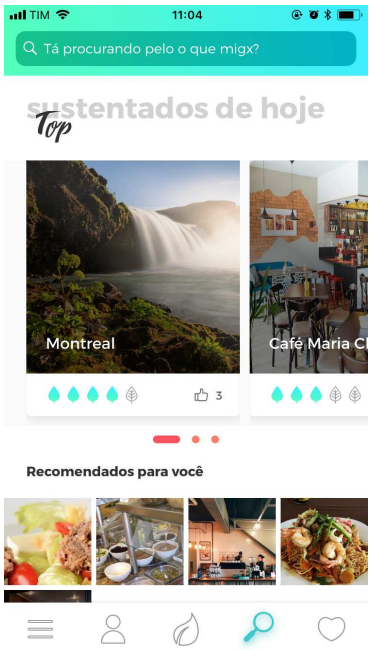
Fonte: Elaborada pelos autores.

5.1.7 Pesquisa

A tela incial da pesquisa apresenta os achados mais populares do dia, isto é, aqueles que possuem mais curtidas no dia. Essa mesma tela apresenta os itens recomendados para o usuário autenticado. Esta recomendação baseia-se em achados publicados por perfis que não são seguidos por esse usuário ordenados pelo número de curtidas como mostrado na Figura 15.

Ao digitar na barra de pesquisa, as opções de filtro aparecem, sendo possível pesquisar por todos (perfil ou estabelecimento), por pessoa ou pelo local conforme as Figuras 16 e 17.

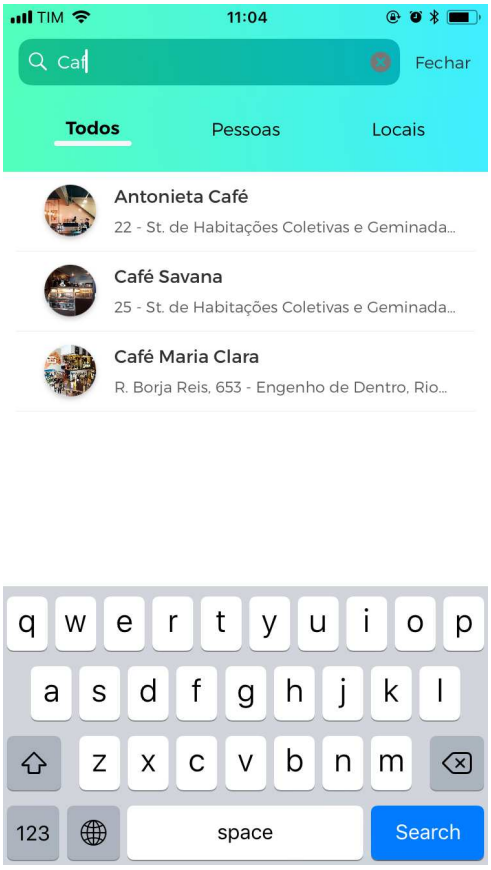
Figura 15 – Página inicial da pesquisa.



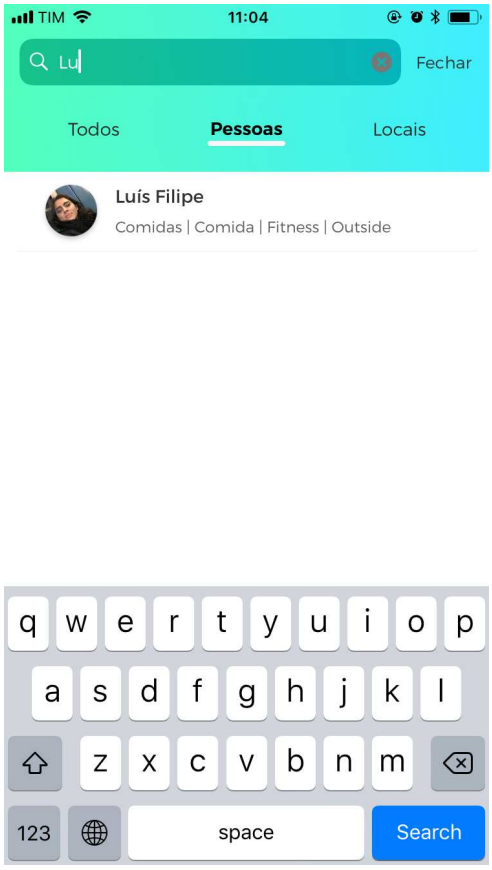
Fonte: Elaborada pelos autores.

Figura 16 – Filtros de pesquisa.

(a) Pesquisa por todos



(b) Pesquisa por pessoa



Fonte: Elaborada pelos autores.

Figura 17 – Pesquisa por locais.

Fonte: Elaborada pelos autores.

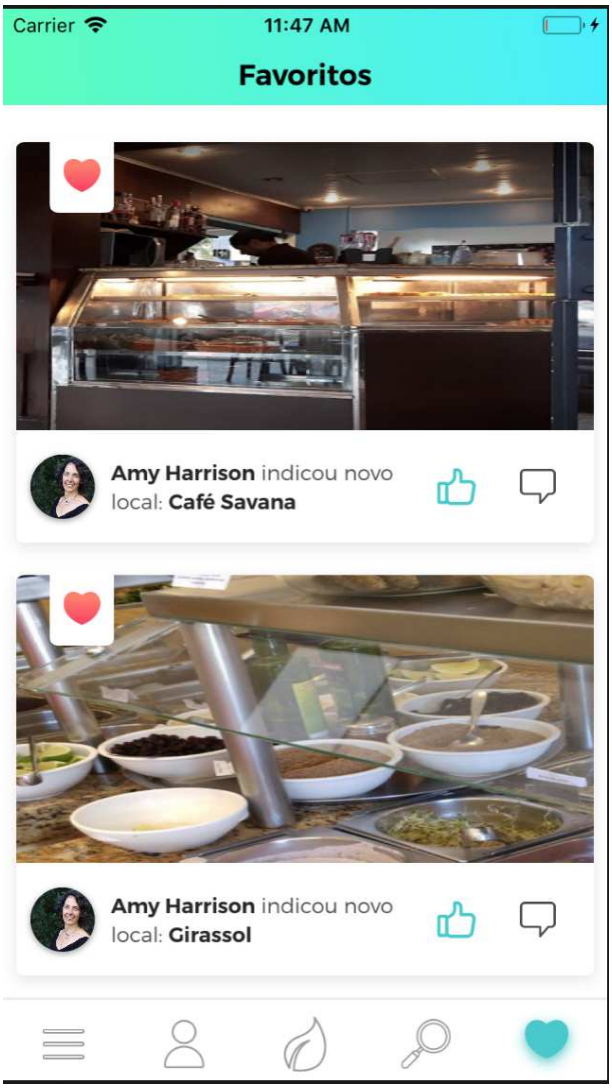
Na pesquisa por todos, o texto digitado é comparado com o nome de estabelecimentos e de perfis. Já na pesquisa por pessoa, o texto é apenas comparado com o nome de perfis.

A pesquisa por local, além de comparar o texto informado com o nome dos estabelecimentos, calcula a distância de um estabelecimento até a localização atual do usuário. Dessa forma, os resultados são ordenados por essa distância de forma crescente e são adicionados ao mapa em forma de marcações, como pode ser visto na Figura 17.

5.1.8 Favoritos

A Figura 18 mostra a tela de favoritos. Essa tela é semelhante a tela de *feed* (seção 5.1.2) porém apenas achados favoritos são listados.

Figura 18 – Tela de favoritos.



Fonte: Elaborada pelos autores.

5.2 Requisitos

Os requisitos do aplicativo foram levantados juntamente com a criadora do conceito *Sustentaê*. Na primeira etapa do projeto, TCC 1, foi realizado uma entrevista com a criadora com o intuito de reafirmar as ideias propostas em seu trabalho de conclusão de curso e a ata da reunião pode ser encontrada no apêndice A.

Os requisitos foram escritos em forma de *User Stories*, uma vez que a metodologia de desenvolvimento adotada foi o *Scrum*. A Tabela 4 trás os requisitos previamente elicitados na parte inicial do projeto.

Tabela 4 – *User Stories* (Histórias de Usuário) previamente elicitados do projeto.

#	*User Story* (História de Usuário)
US01	Eu como usuário, gostaria de me cadastrar no aplicativo para ter acesso ao *login*.
US02	Eu como usuário, gostaria de realizar o *login* no aplicativo para utilizar as funcionalidades do mesmo.
US03	Eu como usuário, gostaria de visualizar os achados das pessoas que sigo em um *feed* para ver as atividades das pessoas que sigo.
US04	Eu como usuário, gostaria de adicionar um achado aos favoritos para poder vê-lo posteriormente.
US05	Eu como usuário, gostaria de curtir a publicação de um achado para demonstrar meu interesse.
US06	Eu como usuário, gostaria de visualizar mais detalhes sobre um achado específico para saber mais acerca do mesmo.
US07	Eu como usuário, gostaria de visualizar a localização exata do achado para pode ir visitá-lo.
US08	Eu como usuário, gostaria de comentar em um achado para interagir com outros usuários.
US09	Eu como usuário, gostaria de visualizar os comentários em um achado para me informar melhor.
US10	Eu como usuário, gostaria de visualizar o meu perfil para poder acompanhar minhas publicações.
US11	Eu como usuário, gostaria de editar o meu perfil para poder mantê-lo atualizado.
US12	Eu como usuário, gostaria de acessar as configurações do aplicativo para poder personalizá-lo conforme meu gosto.
US13	Eu como usuário, gostaria de tornar minha conta privada para minha privacidade.
US14	Eu como usuário, gostaria de desativar a minha conta para encerrar minha participação no aplicativo.
US15	Eu como usuário, gostaria de realizar o *logout* do aplicativo para sair.
US16	Eu como usuário, gostaria de acessar os termos de uso para me manter informado sobre.
US17	Eu como usuário, gostaria de seguir outros usuários para visualizar suas publicações em meu *feed*.
US18	Eu como usuário, gostaria de publicar um novo achado para compartilhá-lo no aplicativo.
US19	Eu como usuário, gostaria de visualizar meus achados favoritos para encontrá-los facilmente.
US20	Eu como usuário, gostaria de visualizar todas as minhas notificações em uma tela para poder acompanhar todas as interações dentro do aplicativo.
US21	Eu como usuário, gostaria de visualizar achados mais curtidos do dia no aplicativo para poder segui-los.
US22	Eu como usuário, gostaria de visualizar achados recomendos a mim para poder segui-los.
US23	Eu como usuário, gostaria de pesquisar um achado para poder segui-lo.
US24	Eu como usuário, gostaria de pesquisar um achado próximo a mim para poder segui-lo.
US25	Eu como usuário, gostaria de pesquisar outro usuário para visualizar seu perfil.

Ainda nessa etapa foi realizado uma prova de conceito com o objetivo de validar a arquitetura geral proposta para o projeto, desde a arquitetura interna da aplicação móvel e sua comunicação com a API, quanto a própria arquitetura proposta para a construção da API.

Para atingir esse objetivo os seguintes requisitos foram implementados em uma única *sprint*, a *sprint* 0:

- US01 - Eu como usuário, gostaria de me cadastrar no aplicativo para ter acesso ao *login*;
- US02 - Eu como usuário, gostaria de realizar o *login* no aplicativo para utilizar as funcionalidades do mesmo.

Com a execução da *sprint* 0 foi atingido o objetivo de validação da arquitetura do projeto, visto que a aplicação móvel já realizava a comunicação com o API, construída por meio dos módulos do *Parse* e hospedados na ferramenta do *Heroku*. Internamente, também foi possível aplicar a arquitetura MVC na aplicação móvel.

Já na segunda etapa, TCC 2, com o desenvolvimento do projeto surgiu a necessidade de realizar mudanças nos requisitos pertencentes ao escopo do projeto. Foram adicionados dois novos requisitos e removidos outros dois. Os requisitos retirados do escopo foram o **US 13** e o **US 20** e os adicionados foram os seguintes:

- US 26 - Eu como usuário, gostaria de criar um novo local para poder publicar achados sobre ele.
- US 27 - Eu como usuário, gostaria de visualizar todos os achados relacionados a um local para poder me informar melhor sobre ele.

5.2.1 Definição das *Sprints*

Após a definição do *backlog* e a execução da *sprint* 0, as histórias de usuário foram priorizadas pela equipe de desenvolvimento juntamente com o *Product Owner* em três estados diferentes: Alta, Média ou Baixa. Em seguida foi utilizada a técnica de *planning poker* para estimar o esforço necessário para realizar cada história de usuário e representá-los em pontos de estimativa.

Por fim, baseado nas prioridades e no esforço estimado, as histórias de usuário foram organizadas em 5 *sprints*, em que cada *sprint* teve duração de duas semanas. A Tabela 5 mostra informações mais detalhadas de cada uma das *sprints*: as datas de início e fim, as histórias de usuário desenvolvidas com suas prioridades e pontos estimados.

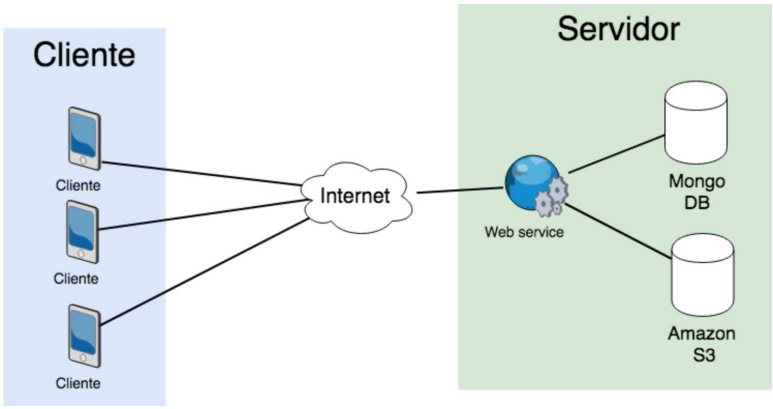
Tabela 5 – Relatório de execução das *Sprints*.

#	Prioridade	Pontos de Estimativa
	Sprint 1 (13/08 - 26/08)	
US18	Alta	21
US03	Alta	13
	Sprint 2 (27/08 - 09/09)	
US10	Alta	21
US06	Alta	8
US05	Média	3
	Sprint 3 (10/09 - 23/09)	
US26	Média	13
US17	Média	5
US07	Média	3
US08	Baixa	5
US04	Baixa	3
US09	Baixa	3
	Sprint 4 (24/09 - 07/10)	
US27	Baixa	8
US12	Baixa	5
US19	Baixa	5
US21	Baixa	5
US22	Baixa	5
US15	Baixa	3
	Sprint 5 (08/10 - 22/10)	
US11	Baixa	8
US23	Baixa	8
US24	Baixa	5
US25	Baixa	5
US14	Baixa	3
US16	Baixa	3

5.3 Arquitetura

O produto desenvolvido por esse trabalho é um aplicativo de rede social. Dessa forma, a interação entre usuários é essencial, sendo assim, houve uma necessidade de desenvolver dois módulos distintos capazes de interagir um com o outro. O primeiro foi denominado **Cliente** e corresponde ao aplicativo *iOS*. Ele é o canal no qual os usuários interagem entre si e com o conteúdo do aplicativo. O segundo módulo é denominado **Servidor**. Este é o responsável por disponibilizar uma interface capaz de gerenciar o armazenamento de dados e por toda a lógica necessária para o funcionamento do aplicativo. A Figura 19 mostra a estrutura geral da solução em que: o módulo **Cliente** possui os dispositivos móveis com o aplicativo em execução e o módulo **Servidor** compreende o *Web Service* que conta com o *MongoDB* como base de dados e uma instância da *Amazon S3* para armazenamento de arquivos binários.

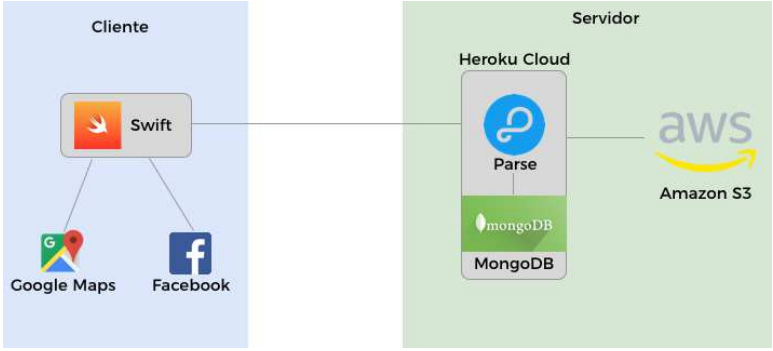
Figura 19 – Estutura geral da solução.



Fonte: Elaborada pelos autores.

5.3.1 Tecnologias Utilizadas

Figura 20 – Tecnologias da arquitetura.



Fonte: Elaborada pelos autores.

As tecnologias utilizadas no Módulo do **Cliente** foi a linguagem de programação *Swift* para o desenvolvimento do aplicativo juntamente com os módulos de autenticação do *Facebook*, utilizado no processo de autenticação e cadastro no aplicativo, e o módulo do *Google Maps*(GOOGLE, 2018) para as interações com o mapa dentro do aplicativo. Na parte **Servidor** foi utilizado o *framework Parse*, sendo disponibilizado via *Heroku Cloud*, integrado com o banco de dados *MongoDB* e o serviço de armazenamento de arquivos binários da *Amazon S3*. As próximas duas seções abordam com mais detalhes cada módulo desenvolvido da solução.

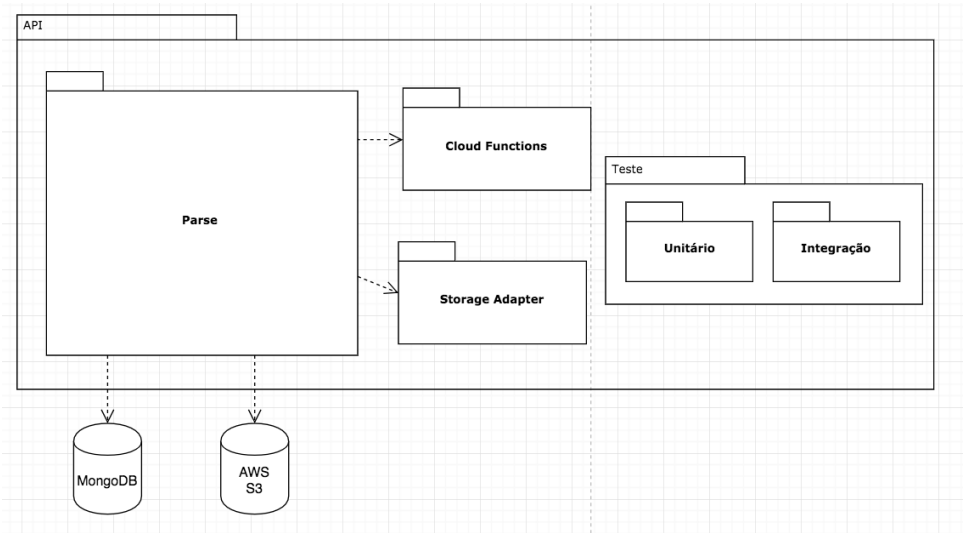
5.3.2 Arquitetura API

O *webservice* desenvolvido para o funcionamento do aplicativo é capaz de gerenciar relações de amizade, sistema de *feed*, sistema de busca, criação de novas postagens, autenticação de usuários. Sendo desenvolvido utilizando o *framework open-source Parse*.

De maneira geral, o *Parse* foi utilizado como um *backend as a service* (modelo que visa disponibilizar funcionalidade de software como serviço). Os sistemas de amizade, curtir, favoritos e criação de postagem foram utilizados a partir da própria implementação disponibilizada pelo *Parse*. Já os itens de pesquisa (por pessoa, local ou ambos) e recomendação de postagens foram desenvolvidos. Essas novas funcionalidades introduzidas no serviço do *Parse* denominou-se *cloud functions* (funçoes de nuvem).

A forma como foi utilizado o *framework Parse* está representado conforme o diagrama de pacotes representado na Figura 21.

Figura 21 – Diagrama de pacotes - **Servidor**.



Fonte: Elaborada pelos autores.

A API está dividida em quatro pacotes principais: teste, *Router, Cloud Functions,*

Parse e *Storage Adapters*. O primeiro pacote de teste contém os testes unitários e de integração da API subdivididos em dois pacotes específicos para cada um deles. O pacote *Cloud Functions* possui as *cloud functions* implementadas - a sessão 5.3.2.1 mostra um exemplo que foi implementado.

Por fim, o pacote *Parse* inclui todas as funções já implementadas pela plataforma do *Parse* utilizando-se do *MongoDB* como banco de dados principal e o pacote *Storage Adapters* realiza a conexão com o serviço *Amazon S3* a fim de armazenar os dados do tipo imagem.

5.3.2.1 *Cloud Functions*

O código-fonte a seguir mostra a *cloud function* responsável por selecionar os itens mais relevantes(*treading*) no momento. De forma geral, a *cloud function* seleciona os quatro achados mais curtidos do último dia.

```
1  Parse.Cloud.define('treading', async (req, res) => {
2    // 24 hours ago
3    var startDate = new Date(new Date().getTime() - (24 * 60 * 60 * 1000))
         ;
4
5    const query = new Parse.Query("Discoveries");
6
7    query.greaterThanOrEqualTo("createdAt", startDate);
8    query.limit(4)
9    query.include("profile")
10   query.include("establishment")
11
12   let todayDiscoveries = await query.find();
13   let likesDiscoveriesMap = await mapDiscoveryByLike(todayDiscoveries)
14
15   res.success(likesDiscoveriesMap.sort(sortLikesRank))
16 })
```

5.3.2.2 Banco de dados

Como exemplificado na seção 3.2.4, o *MongoDB* é um banco do tipo *NoSQL* orientado a documentos, portanto a representação dos dados é feita de forma diferente ao modelo tradicional de entidade-relacionamento. As Figuras 22 a, b e c mostram uma forma de organizar os dados nesse tipo de banco de dados utilizando *JSON* como esquemas e representam como os dados foram organizados na plataforma do *Parse*.

Figura 22 – Modelo de dados.

(a) Organização *users, profiles, comments*

```
/* User */
{
  "objectId":"String",
  "profile":"Pointer<Profile>",
  "authData" : {
    "facebook": {
      "id": "String",
      "access_token": "String",
      "expiration_date": "Date"
    }
  }
}

/* Profile */
{
  "objectId":"String",
  "name":"String",
  "user":"Pointer<User>",
  "cover":"Pointer<Image>",
  "avatar":"Pointer<Image>",
  "tags": ["String"]
}

/* Comments */
{
  "objectId":"String",
  "createdAt":"Date",
  "content":"String",
  "discovery":"Pointer<Discovery>",
  "profile":"Pointer<Profile>"
}
```

(b) Organização *Favorites, Follow, Likes, Images*

```
/* Favorites */
{
  "objectId":"String",
  "discovery":"Pointer<Discoveries>",
  "profile":"Pointer<Profile>"
}


/* Follow */
{
  "objectId":"String",
  "to":"Pointer<Profile>",
  "from":"Pointer<Profile>"
}

/* Likes */
{
  "objectId":"String",
  "discovery":"Pointer<Discoveries>",
  "profile":"Pointer<Profile>"
}

/* Images */
{
  "objectId":"String",
  "content":"File"
}
```

(c) Organização *Discoveries, Establishments*

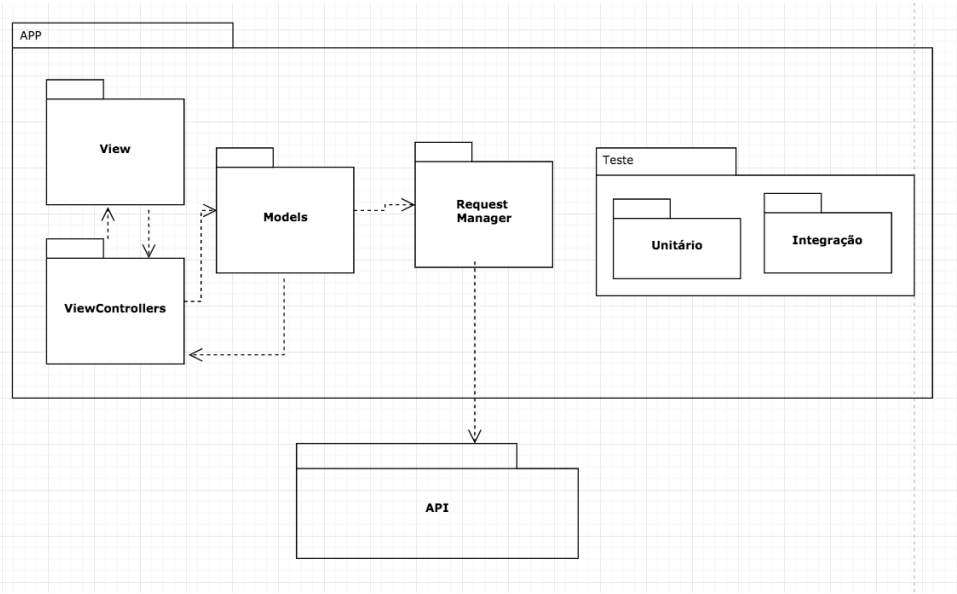
```
/* Discoveries */
{
  "objectId":"String",
  "profile":"Pointer<Profile>",
  "score":"Number",
  "establishment":"Pointer<Establishments>",
  "images": [{
    "__type": "Pointer",
    "className": "Images",
    "objectId": "String"
    }
  ],
  "content":"String"
}

/* Establishments */
{
  "objectId":"String",
  "name":"String",
  "location":"GeoPoint"
}
```

Fonte: Elaborada pelos autores.

5.3.3 Arquitetura do aplicativo

A arquitetura utilizada no aplicativo será o MVC. Este tipo de arquitetura foi selecionada por ser o padrão usado em aplicativos para a plataforma *iOS* com utilização da ferramenta *XCode*. Definida esta arquitetura, a criação dos pacotes *ViewControllers, Views e Models* fica evidente. Para o gerenciamento de dados foi necessário a criação de um pacote de *Models* tornando acessível os dados via servidor, por meio do pacote *Request Manager* que comunica-se com o pacote da API. Dessa forma, o diagrama de pacotes na Figura 23 representa uma visão geral da arquitetura desenvolvida, além do pacote de *Teste* contendo dois subpacotes voltados aos testes unitários e de integração.

Figura 23 – Diagrama de pacotes - Aplicativo.



Fonte: Elaborada pelos autores.

5.4 Qualidade de Software

Atualmente, 4 das 5 empresas mais valiosas do mundo estão no mercado de desenvolvimento de software e os recursos destinados a tecnologia da informação nas empresas cresceu em 49%. Paralelamente, um conjunto de normas e modelos para assegurar a qualidade se estabeleceram. Dessa forma, a indústria de software passou a investir mais recursos na garantia da qualidade de software, levando a um grande número de padrões de qualidade, isto é, práticas que ajudam as organizações a alcançarem melhores níveis de garantia de qualidade (MUNOZ; MEJIA; IBARRA, 2018).

Dessa maneira, fez-se necessário aderir as convenções de código para as duas linguagens a serem utilizadas na implementação do software para aumentar a qualidade e manutenabilidade do código. Convenções de código são uma maneira de padronizar o for-

mato do código escrito, definindo espaçamentos, identações, quebras de linha, tamanho máximo de linhas, nomeclaturas e etc. Abaixo segue os links para acesso das convenções de código utilizadas para cada linguagem de programação do projeto:

- *Swift* - <https://github.com/raywenderlich/swift-style-guide>
- *JavaScript* - <https://github.com/airbnb/javascript>

5.4.1 Testes Automatizados

Foram desenvolvidos testes automatizados (unitários e integração) com o intuito de validar que o código desenvolvido está de acordo com os resultados esperados.

O código da aplicação como também dos testes podem ser encontrados no seguinte *link*: https://goo.gl/ZWQA6V. Um exemplo de teste unitário e de teste de integração está disponível no apêndice E.

5.4.2 Integração e Entrega Contínua

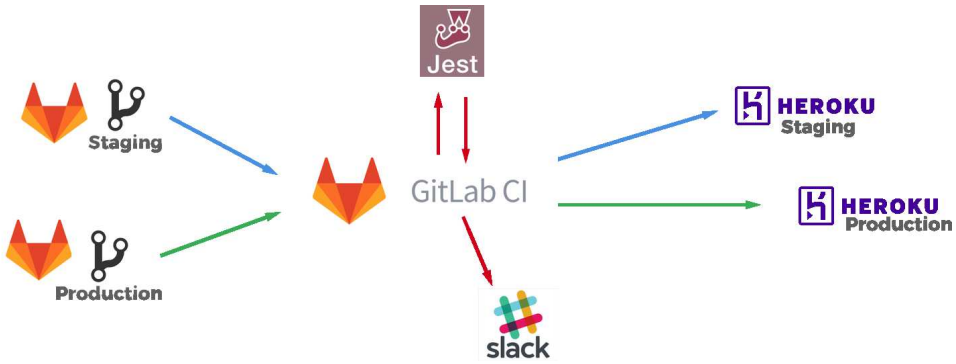
A implementação da integração e entrega contínua foi baseada em três ambientes diferentes: *development, staging* e *production*. O ambiente de *development* (desenvolvimento) é voltado para o desenvolvimento do projeto e pode conter erros e funcionalidades experimentais, já o de *staging* ou de preparo, é utilizado para testes em fase *beta* - *beta*, *bugs expected to appear* ou erros esperados que apareçam é a versão de um produto que ainda não foi finalizado - que foram distribuídos para teste. Por fim, o ambiente de *production* (produção), é onde a aplicação estará sendo utilizada pelos usuários finais.

Primeiramente na API, o processo de integração contínua utilizou a ferramenta do próprio sistema que realiza o controle de versão (*GitLab*), o *GitLabCI*. Após um *merge request* ser revisado e aceito pela equipe de desenvolvimento na *branch staging* ou *production*, o *GitLabCI* executa toda a suíte de testes da aplicação, utilizando o *framework Jest*, para verificar a ocorrência de algum erro. Caso não ocorram erros, o *GitLabCI* criará uma nova *build* e informará o resultado do processo de integração à equipe responsável por meio da ferramenta de comunicação *Slack*.

Após todo o processo de integração, é a vez do processo de entrega contínua. Isto é, o processo de disponibilização do produto de software será executado e distribuirá o produto para o ambiente correto após o processo de integração. Caso o processo de integração tenha ocorrido na *branch staging*, essa nova versão será disponibilizada no ambiente de *staging*, caso tenha ocorrido na *branch production* essa nova versão será disponibilizada no ambiente de *production*.

A Figura 24 exemplifica o processo realizado na API.

Figura 24 – Processo de integração contínua da API.

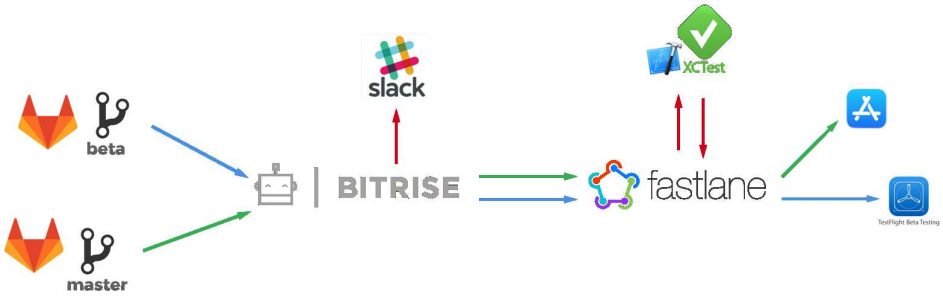


Fonte: Elaborada pelos autores.

No aplicativo *iOS* o processo segue o mesmo padrão visto na API, porém com ferramentas diferentes. Depois da aprovação de um *merge request* na *branch beta* ou *master*, a ferramenta *Bitrise* realiza o *download* do código da *branch* em questão em uma máquina virtual *MacOS*. Após isso, o *Bitrise* utiliza outra ferramenta, o *Fastlane*. Esta por sua vez, realiza a compilação do projeto na máquina virtual e executa toda a suíte de testes contidos no aplicativo, utilizando o *framework XCTest* da própria *Apple*. Caso todos os testes tenham sucesso, a ferramenta criará uma nova *build* do aplicativo.

Se esse processo tiver se iniciado na *branch beta*, o *Fastlane* realiza o *deploy* na ferramenta de testes da *Apple*, o *TestFlight*, caso for disparado pela *branch master*, a ferramenta faz o *deploy* na *App Store*. Ao fim de todo o processo, o *Bitrise* manda as informações da *build* para o *Slack*. A Figura 25 mostra o processo realizado no aplicativo *iOS*.

Figura 25 – Processo de integração contínua do aplicativo *iOS*.



Fonte: Elaborada pelos autores.

A integração do *GitLab* com as ferramentas *Bitrise*, *Fastlane*, *TestFlight*, e *Slack* foi concluída com sucesso. Entretanto, o processo de integração contínua no aplicativo *iOS* não foi totalmente finalizado. Visto que durante a construção desse trabalho a *Apple* realizou uma notável atualização na *IDE Xcode*, que é utilizada para compilar e empacotar um aplicativo, lançando uma nova versão (10.1) a qual ainda não possui suporte na

ferramenta *Bitrise* e portanto, até a finalização das *sprints* desse projeto, não foi possível realizar a integração. Apesar desse acontecimento, a compilação automatizada do projeto pode ser disparada manualmente através da ferramenta *Fastlane*.

5.5 Estudo de Usabilidade

Os testes executados para a avaliação do produto tiveram como foco as principais funcionalidades do aplicativo. Isto é, analisar como é o comportamento e o sentimento do usuário ao utilizá-las. São elas:

- Seguir outro perfil e visualizar as postagens desse perfil;
- Avaliar um local, criando-o como um novo "achado";
- Visualizar as informações de um "achado" e realizar um comentário a respeito.

Para esse teste de usuabilidade realizado, utilizou-se a metodologia somativa. Segundo Tullis e Albert (2008), essa metodologia consiste em avaliar o quanto um software ou uma parte de um software consegue cumprir com seus objetivos depois de construído através de um indicador numérico. Sendo assim, foram definidas tarefas a serem executadas pelos participantes do teste. Para cada tarefa, um valor foi atribuído conforme o sucesso de sua execução(realizada ou não) e um outro para medir se houve necessidade de ajuda externa para que a tarefa fosse executada. Com os dois indicadores foi possível identificar quais tarefas os usuários estão conseguindo realizar e em quais há mais dificuldades.

Ao final de cada teste, um questionário foi aplicado visando identificar como foi a experiência do participante ao utilizar o aplicativo. O detalhamento das tarefas, suas atribuições, análises e questionário são relatados a seguir.

5.5.1 Execução e análise de tarefas

O critério estabelecido para que uma tarefa seja considerada executada com sucesso é que ela tenha um início e fim bem definidos. Isto é, o participante deve informar quando começou e quando terminou a tarefa.

A métrica utilizada para analisar as tarefas foi atribuir um valor numérico para o sucesso ou fracasso na sua execução. Caso seja executada com sucesso o valor atribuído foi de 1, e em caso de fracasso 0. Com essa atribuição de valores, é possível calcular a

porcentagem de sucesso para cada atividade, que é dada pelo valor médio do somatório de pontos de cada atividade conforme a fórmula 5.1.

$$(\sum_{P=1}^{N} i)/N \tag{5.1}$$

Onde P é o número do participante, N o número total de participantes e i o resultado da execução.

Para a análise da necessidade de ajuda externa, a qual nomeou-se nível de sucesso, buscou-se identificar o quão fácil foi finalizar uma determinada tarefa. Para realizar o cálculo desse nível foi atribuído um valor numérico para o desempenho de cada participante ao executar uma tarefa. Os valores são listados a seguir:

- executada com facilidade: 1;
- executada com pouca ajuda: 0,75;
- executada com alguma ajuda: 0,5;
- executada com muita ajuda: 0,25.

Com essa métrica foi possível analisar, dentro da porcentagem de conclusão, a facilidade em que uma tarefa foi concluída.

5.5.2 Tarefas do Estudo de Usabilidade

As tarefas elicitadas para o teste de usabilidade são:

- **Tarefa 1**: Autentique-se no aplicativo através da sua conta do *Facebook*(FACEBOOK, 2018);
- **Tarefa 2**: Procure pelo usuário Gabriel Araújo, veja seu perfil e siga-o;
- **Tarefa 3**: Vá ao seu *feed* e verifique as postagens. Escolha uma postagem para curtir, visualizar mais informações e realizar um comentário sobre;
- **Tarefa 4**: Vá ao seu *feed* e salve uma postagem como favorito, depois verifique a sua lista de favoritos;
- **Tarefa 5**: Crie uma nova postagem para o local que você se encontra com as seguintes informações:
  - Avaliação: 4 folhas
  - Descrição: Minha casa é o lugar mais sustentável

- Imagem: Tire uma foto
- Nome do achado: A casa do [seu nome]

A tarefa 1 é essencial, uma vez que representa o primeiro passo para a utilização do aplicativo, que é a autenticação. A tarefa 2 valida o sistema de busca além da integração entre os usuários do aplicativo. A tarefa 3 e 4 validam a interação principal do aplicativo, que é a descoberta de novos achados e por fim, a tarefa 5 valida a criação de um novo achado no aplicativo.

O roteiro do teste de usabilidade pode ser encontrado no apêndice C.

5.5.3 Experiência de usabilidade

Nessa etapa do teste de usabilidade o elementro central a ser analisado é o usuário. Saber o que ele pensa e como se sente em relação ao aplicativo avaliado é a base dessa análise. Independente se houve dificuldade ou não ao utilizar o aplicativo o que se espera avaliar nessa etapa é a satisfação ao final da experiência (TULLIS; ALBERT, 2008).

A métrica que foi utilizada é um questionário que avalia a experiência do usuário após utilizar o aplicativo. Esse questionário descreve a visão do usuário em relação ao aplicativo.

O questionário adotado foi o *System Usability Scale* (SUS) desenvolvido por Brooke (1996). Esse questionário é formado por 10 afirmações, em que metade é positiva e a outra metade negativa, com escala de 5 níveis (1-5). A pontuação geral é a média da pontuação dos participantes multiplicada por 2,5. A pontuação de cada participante é calculada através da soma dos pontos obtidos em cada afirmação na qual o participante respondeu, em que a pontuação de cada afirmaçao é contabilizada da seguinte forma:

- as afirmações de números 1, 3, 5, 7 e 9, a pontuação é a posição da escala marcada pelo participante subtraída do número 1.
- as afirmações de números 2, 4, 6, 8 e 10, a pontuação é o número 5 subtraído da posição da escala marcada.

5.5.3.1 Questionário SUS

Tabela 6 – Questionário SUS.

	Discordo Totalmente(1)	Discordo(2)	Neutro(3)	Concordo(4)	Concordo Totalmente(5)
1. Acho que gostaria de usar este sistema com freqüência.					
2. Achei o sistema desnecessariamente complexo					
3. Achei o sistema fácil de usar.					
4. Achei que seria necessário o apoio de um técnico para poder usar este sistema.					
5. As funções deste sistema estavam bem integradas.					
6. Achei este sistema muito inconsistente.					
7. Imagino que a maioria das pessoas aprenderiam a usar este sistema rapidamente					
8. Achei o sistema muito complicado de usar					
9. Eu me senti muito confiante com o sistema					
10. Eu preciso aprender um monte de coisas antes de continuar usando este sistema					

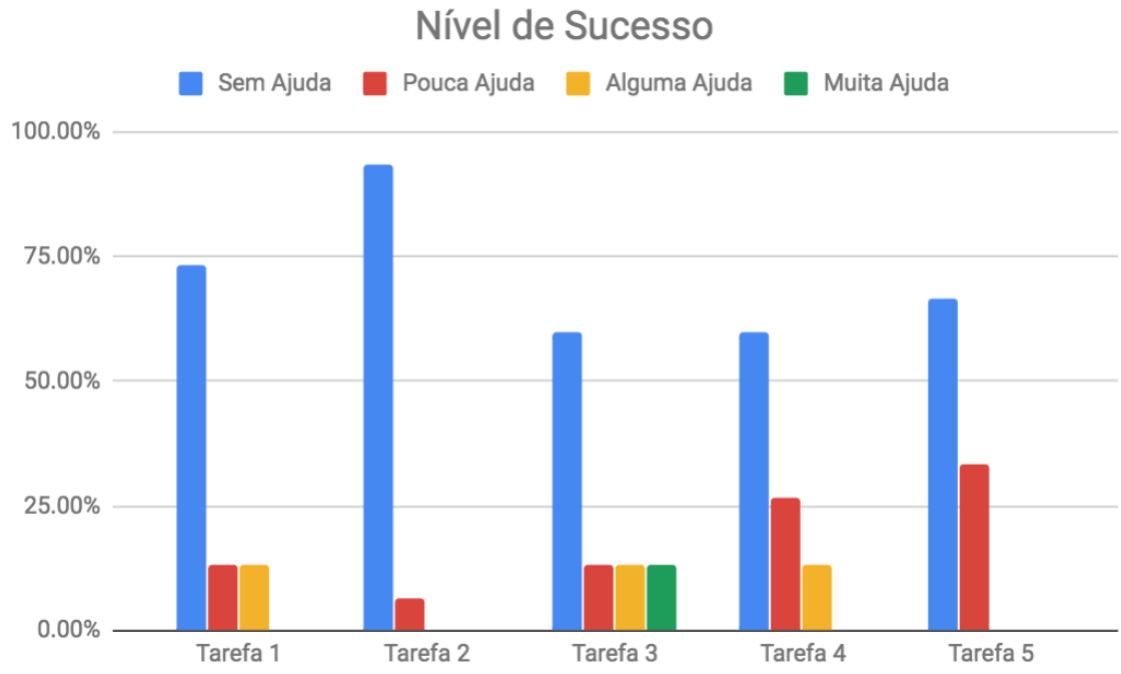
Segundo Tullis e Albert (2008), o SUS possui um rendimento considerável e consistente para testes realizados com um pequeno número de usuários. Com 8 participantes, pode-se identificar preferências e problemas com 80% de precisão, uma vez que há a utilização de afirmações positivas e negativas.

5.5.4 Resultados e Análises

5.5.4.1 Índices de sucesso e nível de sucesso

Os 15 participantes executaram todas as tarefas com sucesso (o resuldado individual se encontra no apêndice D). Observando somente esse indicador, tem-se um rendimento de 100%. O que indica que todas as tarefas foram executadas com sucesso, ou seja, que o aplicativo apresenta uma boa arquitetura no sentido da organização das telas, botões e texto. Contudo o nível de succeso expõe dados que revelam que para todas as tarefas serem executadas houve necessidade de ajuda externa conforme pode-se ver na Figura 26.

Figura 26 – Gráfico do nível de sucesso das tarefas.



Fonte: Elaborada pelos autores.

Os índices exibidos na Figura 26 mostram que apenas um participante necessitou de ajuda para executar a tarefa 2, que está relacionada ao sistema de busca. Entretanto, foi observado que essa dificuldade gerada na execução da tarefa, está relacionada ao uso do sistema operacional *iOS*. Uma vez que a mecânica para retornar à tela anterior em

dispositivos *iOS* é difirente dos dispositivos *Android*, aparelho no qual o participante é habituado a utilizar.

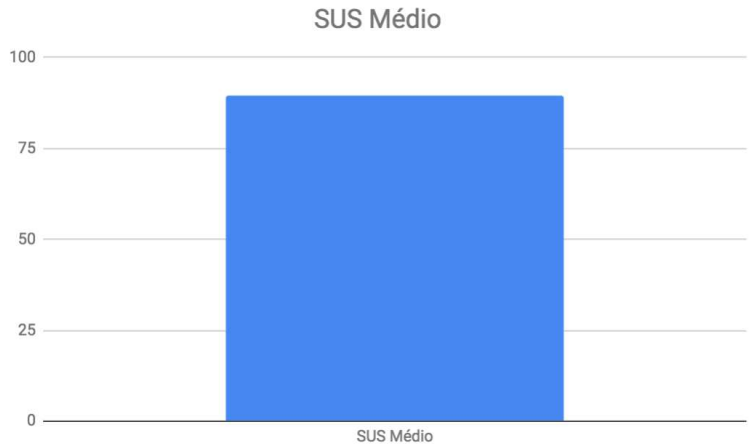
É possível identificar através da Figura 26 que no sistema de autenticação, relacionado à tarefa 1, houve algumas dúvidas. Aproximadamente 26,66% dos participantes tentaram efetuar o acesso através do *Facebook* utilizando as credenciais da rede social nos campos de usuário e senha e não através do botão de "Entrar com Facebook". A explicação dos participantes que cometeram esse engano foi de que o botão não tem destaque maior se comparado com o formulário com *email* e senha.

As tarefas 3, 4 e 5 apresentam altos índices de ajuda conforme a Figura 26. Nas tarefas 3 e 4, o ponto em que houve intensa necessidade de ajuda foi o de curtir *versus* favoritar. No aplicativo foi utilizado o ícone de "joinha" para curtir e o "coração" para salvar como favorito. Contudo os participantes que usam frequentemente a rede social *Instagram*(INSTAGRAM, 2018) utilizam o botão de "coração" para curtir, uma vez que esse é o ícone utilizado pelo *Instagram* que simboliza a ação de curtir. Já na tarefa 5, houve a necessidade de ajuda para localizar a tela de criação de achados e em alguns casos os participantes entraram tela por tela até acharem a correta. Um fator que se destacou foi o ícone utilizado para a página (uma folha de árvore), este pouco sugestível para a indicação de uma página de criação. Pode-se relacionar isso ao fato de que os participantes ainda não se familiarizaram com a ideia da "nota sustentei" (representada pelo mesmo ícone) e obrigatória para a criação de um novo "achado".

As atividades de melhoria que surgiram após os testes estão na seção 6.1, onde é relatado os trabalhos futuros.

5.5.4.2 Experiência de usabilidade

Figura 27 – SUS Médio.



Fonte: Elaborada pelos autores.

A pontuação média do SUS obtida nesse estudo foi de 89,5%, conforme apresentado na Figura 27 e as respostas de cada participante estão no apêndice D. Segundo Brooke (1996), pontuações do SUS abaixo de 60% indicam que o sistema possui uma experiência moderadamente pobre e insatisfação do usuário, e pontuações acima de 80% indicam uma experiência muito boa com alto índice de satisfação dos usuários. Dessa forma, os resultados dos testes aplicados mostraram que os usuários estão satisfeitos com o aplicativo e este possui um bom nível de experiência.

5.5.5 Considerações Finais

De maneira geral, segundo os índices obtidos na experiência de usabilidade e através das tarefas, os usuários apresentaram satisfação ao utilizar o aplicativo. Relatando uma experiência de usabilidade agradável e intuitiva. Contudo há alguns conceitos que não foram assimilados no primeiro momento, como o conceito da "nota sustentei" e da dificuldade de compreender qual botão utilizar para curtir ou favoritar, como relatado na seção 5.5.4.1 deste trabalho.

6 Conclusão

O objetivo principal desse trabalho, conforme descrito na seção 1.3.1 do capítulo 1, foi a construção do aplicativo *Sustentaê*, uma rede social que dá suporte à sustentabilidade compartilhada, e sua posterior avaliação por usuários. Esse objetivo foi alcançado, visto que todos os requisitos elicitados no escopo do projeto foram implementados e a avaliação de usabilidade foi executada como explanado na seção 5.5.4 deste trabalho.

Do ponto de vista do produto final desenvolvido nesse tabalho, os resultados superaram as expectativas dos envolvidos. O aplicativo demonstrou que é capaz de desempenhar o papel de plataforma de divulgação do locais sustentáveis, uma vez que reuni essas informações em um mesmo lugar e de maneira estruturada, além de que o teste de usabilidade apontou índices positivos (sessão 5.5.4).

A autora do conceito Sustentaê, Vanessa Raulino, ficou satisfeita com o resultado. Segundo seu relato, ver o seu trabalho de conceituação sendo executado e de forma funcional é gratificante.

Diversos conceitos, métodos e práticas da Engenharia de Software foram utilizados para atingir esse objetivo, como: engenharia de requisitos, metodologia de desenvolvimento ágil, padrões de projeto, arquitetura de software, testes de software automatizados, integração e entrega contínua.

A ferramenta fundamental para o desenvolvimento desse projeto foi o *framework open-source Parse*. Com ele foi possível desenvolver uma pequena rede social de forma rápida e eficiente, uma vez que a plataforma é estável e possui uma grande e ativa comunidade.

6.1 Trabalhos Futuros

Para trabalhos futuros, podem ser considerados os seguintes itens:

1. ***Push-notifications***: a implementação de notificações do tipo *push* é bastante válida a ser agregada ao produto, visto que uma rede social pode ter a sua experiência melhorada com esse tipo de interação, em que o usuário é notificado quando alguma ação relativa a ele ocorre dentro do aplicativo;

2. **Testes de desempenho**: a fim de garantir e testar a escalabilidade do aplicativo, testes de desempenho podem ser realizados, como testes de *stress*, de volume e de carga;

3. **Perfil empresa**: Desenvolver, no aplicativo, a funcionalidade "Perfil empresa" que consiste em habilitar que o responsável por um estabelecimento "achado" possa gerenciar um perfil composto pelas publicações dos usuários (perfil dinâmico).

Referências Bibliográficas

ALVES, D. E.; SILVA, C. A. da; FERREIRA, A. M. G. Environmental communication. In: *Proceedings of the 26th Annual ACM International Conference on Design of Communication*. New York, NY, USA: ACM, 2008. (SIGDOC '08), p. 279–280. ISBN 978-1-60558-083-8. Disponível em: <http://doi-acm-org.ez54.periodicos.capes.gov.br/10.1145/1456536.1456598>. Citado 2 vezes nas páginas 23 e 24.

AMAZON. *versão 01/03/2006 da API*. Amazon.com, 2018. Disponível em: <https://aws.amazon.com/pt/s3/>. Citado na página 45.

AMBIENTAL, S. P. E. S. do Meio Ambiente / Coordenadoria de P. *Economia Verde: desenvolvimento, meio ambiente e qualidade de vida no Estado de São Paulo*. 2010. Citado na página 28.

APPLE. *versão 9.2*. Apple Inc., 2018. Disponível em: <https://developer.apple.com/xcode/>. Citado na página 44.

ASADI, M.; RAMSIN, R. Method engineering process patterns. In: *Proceedings of the 2nd India software engineering conference*. ACM, 2009. p. 143–144. Disponível em: <https://dl.acm.org/citation.cfm?id=1506249>. Citado na página 34.

ATOM. *versão 1.24*. GitHub Inc., 2018. Disponível em: <https://atom.io/>. Citado na página 47.

BARBOSA, E. F. *Introdução ao Teste de Software*. 2009. Disponível em: <http://www.labes.icmc.usp.br/site/sites/default/files/NotaDidatica65.pdf>. Acesso em: 4 Maio 2018. Citado na página 39.

BECK, K. *Extreme Programming Explained: Embrace Change*. [S.l.]: Addison-Wesley Professional, 2004. ISBN 0321278658. Citado na página 40.

BERKMAN-CHARDON, A. et al. Scenario-based programming for mobile applications. In: *Proceedings of the International Conference on Mobile Software Engineering and Systems*. New York, NY, USA: ACM, 2016. (MOBILESoft '16), p. 161–172. ISBN 978-1-4503-4178-3. Disponível em: <http://doi-acm-org.ez54.periodicos.capes.gov.br/10.1145/2897073.2897080>. Citado na página 35.

BERNERS-LEE, F. *Uniform Resource Identifier (URI): Generic Syntax*: Rfcs. 2005. Disponível em: <http://www.rfc-base.org/rfc-2616.html>. Acesso em: 4 Maio 2018. Citado na página 38.

BESSGHAIER, N.; SOUII, M. Towards usability evaluation of hybrid mobile user interfaces. In: *2017 IEEE/ACS 14th International Conference on Computer Systems and Applications (AICCSA)*. [S.l.: s.n.], 2017. Citado na página 36.

BITRISE. Bitrise ltd., 2018. Disponível em: <https://www.bitrise.io/>. Citado na página 46.

BONITA. *versão 7.6*. Bonitasoft, 2018. Disponível em: <https://www.bonitasoft.com/>. Citado na página 44.

BOONCHUAY, K. Design and implementation a rest api for association rule mining. In: *2017 14th International Conference on Electrical Engineering/Electronics, Computer, Telecommunications and Information Technology (ECTI-CON)*. [S.l.: s.n.], 2017. Citado na página 38.

BROOKE, J. Sus - a quick and dirty usability scale. *Redhatch Consulting Ltd.*, 1996. Citado 2 vezes nas páginas 78 e 82.

BRUNDTLAND, G. et al. *Our Common Future ('Brundtland report')*. Oxford University Press, USA, 1987. Paperback. (Oxford Paperback Reference). Disponível em: <http://www.bne-portal.de/fileadmin/unesco/de/Downloads/Hintergrundmaterial_international/Brundtlandbericht.File.pdf?linklisted=2812>. Citado na página 27.

CAIRNCROSS, F. *Meio ambiente: custos e benefícios*. São Paulo: Nobel, 1992. Citado na página 24.

CARBONZ. *versão 1.6.1*. Carbon Z., 2018. Disponível em: <http://carbonzapp.com.br/>. Citado na página 30.

FACEBOOK. *versão 197.1*. Facebook Inc., 2018. Disponível em: <https://www.facebook.com/>. Citado na página 77.

FASTLANE. *versão 2.89*. Google, Inc, 2018. Disponível em: <https://fastlane.tools/>. Citado na página 46.

FERREIRA, A. B. *Novo Dicionário Aurélio da Língua Portuguesa*. 2004. Citado na página 23.

FIELDING, G. *Hypertext Transfer Protocol – HTTP/1.1. Internet*: Rfcs. 1999. Disponível em: <http://www.rfc-base.org/rfc-2616.html>. Acesso em: 4 Maio 2018. Citado na página 38.

FIELDING, R. T. *Architectural Styles and the Design of Network-based Software Architectures*. Tese (Doutorado), 2000. AAI9980887. Citado na página 38.

FORTUNATO, D.; BERNARDINO, J. Progressive web apps: An alternative to the native mobile apps. In: *2018 13th Iberian Conference on Information Systems and Technologies (CISTI)*. [S.l.: s.n.], 2018. Citado 2 vezes nas páginas 35 e 36.

GAMMA, E. *Design Patterns: Elements of Reusable Object-Oriented Software*. Pearson Education, 1994. Disponível em: <http://www.uml.org.cn/c++/pdf/DesignPatterns.pdf>. Citado na página 36.

GARLAN, D.; SHAW, M. Software architecture: reflections on an evolving discipline. ACM Press, 2011. Disponível em: <http://dl.acm.org/citation.cfm?doid=2025113.2025116>. Citado na página 36.

GIL, A. C. *Como elaborar projetos de pesquisa*. São Paulo: Atlas, 2002. Citado na página 49.

GIT. *versão 2.17*. Git, 2018. Disponível em: <https://git-scm.com/>. Citado na página 46.

GITLAB. GitLab Inc, 2018. Disponível em: <https://about.gitlab.com/>. Citado na página 46.

GOOGLE. *versão 2.7.0*. Google Inc., 2018. Disponível em: <https://cloud.google.com/maps-platform/>. Citado na página 70.

GRANERO, A. E.; COUTO, T. C. Consumo no ciberespaço: a explosão de aplicativos de dispositivos móveis que ajudam a controlar a vida na palma da mão. *Revista GEMInIS*, v. 4, n. 2, p. 89–105, dec 2013. ISSN 2179-1465. Disponível em: <http://www.revistageminis.ufscar.br/index.php/geminis/article/view/147>. Citado na página 23.

GUANGCHUN, L.; LU, W.; HANHONG, X. A novel web application frame developed by mvc. *SIGSOFT Softw. Eng. Notes*, ACM, New York, NY, USA, v. 28, n. 2, p. 7–, mar. 2003. ISSN 0163-5948. Disponível em: <http://doi-acm-org.ez54.periodicos.capes.gov.br/10.1145/638750.638779>. Citado na página 37.

HEROKU. Salesforce.com, 2018. Disponível em: <https://www.heroku.com/>. Citado na página 46.

HU, G. Market and social welfare analysis for hybrid sustainable manufacturing industry. *Int. J. Sustainable Manufacturing*, v. 2, n. 4, p. 338–355, 2012. Citado na página 23.

HUMBLE, D. F. J. *Continuous Delivery: Reliable Software Releases through Build, Test, and Deployment Automation*. [S.l.]: Addison-Wesley Professional;, 2010. ISBN 0321601912. Citado na página 40.

INSTAGRAM. *versão 71.0*. Instagram Inc., 2018. Disponível em: <https://www.instagram.com/>. Citado na página 81.

JEST. *versão 23.0*. Facebook Inc., 2018. Disponível em: <https://jestjs.io/>. Citado na página 45.

KUMAR, L.; SUREKA, A. An empirical analysis on web service anti-pattern detection using a machine learning framework. In: *2018 IEEE 42nd Annual Computer Software and Applications Conference (COMPSAC)*. [S.l.: s.n.], 2018. ISSN 0730-3157. Citado na página 37.

KUN, Y. Underlying techniques for web services: A survey. *Journal of software*, 2004. Citado na página 38.

LATEX. *versão 3.14*. LaTeX3 Project, 2018. Disponível em: <https://www.latex-project.org/>. Citado na página 47.

MACHADO, F. N. R. *Análise e Gestão de requisitos de software: onde nascem os sistemas*. São Paulo, Brasil: Editora Érica, 2011. Citado na página 33.

MALAVOLTA, I. Beyond native apps: Web technologies to the rescue! (keynote). In: *Proceedings of the 1st International Workshop on Mobile Development*. New York, NY, USA: ACM, 2016. (Mobile! 2016), p. 1–2. ISBN 978-1-4503-4643-6. Disponível em: <http://doi-acm-org.ez54.periodicos.capes.gov.br/10.1145/3001854.3001863>. Citado na página 35.

MARTIN, R. C. *Design Principles and Design Patterns*. Object Mentor, 2000. Disponível em: <http://www.objectmentor.com/resources/articles/Principles_and_Patterns.pdf>. Citado na página 36.

MAURER, F. A case study on the impact of scrum on overtime and customer satisfaction. *Agile Development Conference*, IEEE Cumputer Society, 2005. Citado na página 34.

MIKHAILOVA, I. Sustentabilidade: Evolução dos conceitos teóricos e os problemas da mensuração prática. *Revista Economia e Desenvolvimento*, v. 16, p. 23–24, 2004. Disponível em: <http://w3.ufsm.br/depcie/arquivos/artigo/ii_sustentabilidade.pdf>. Citado na página 27.

MINAYO, M. C. de S. *O desafio do conhecimento*. São Paulo, Brasil: Hucitec, 1993. Citado na página 49.

MONGODB. *versão 4.0*. MongoDB Inc., 2018. Disponível em: <https://www.mongodb.com/>. Citado na página 45.

MORESI, E. *Metodologia da Pesquisa*. 2003. Disponível em: <https://goo.gl/Nm4Ybz>. Acesso em: 29 mai. 2018. Citado na página 49.

MUNOZ, M.; MEJIA, J.; IBARRA, S. Tools and practices to software quality assurance: A systematic literature review. In: *2018 13th Iberian Conference on Information Systems and Technologies (CISTI)*. [S.l.: s.n.], 2018. Citado na página 73.

NIELSEN, J. *Why You Only Need to Test with 5 Users*. 2000. Disponível em: <https://www.nngroup.com/articles/why-you-only-need-to-test-with-5-users/>. Acesso em: 5 de Julho de 2018. Citado na página 53.

PARSE. Facebook, 2018. Disponível em: <https://github.com/parse-community>. Citado na página 45.

PEZZE, M. *Teste e Análise de Software - Processos, Princípios e Técnicas*. [S.l.]: ABOOKMAN, 2008. ISBN 8577802620. Citado na página 39.

PNUMA. *Caminhos para o Desenvolvimento Sustentável e a Erradicação da Pobreza – Síntese para Tomadores de Decisão*. 2011. Disponível em: <www.unep.org/greeneconomy>. Acesso em: 13 abr. 2018. Citado na página 27.

PRODANOV, C. C.; FREITAS, E. C. *Metodologia do trabalho cientiífico: Métodos e Técnicas da Pesquisa e do Trabalho Acadêmico*. Rio Grande do Sul: Feevale, 2013. ISSN 978-85-7717-158-3. Citado 2 vezes nas páginas 49 e 50.

QUE, P.; GUO, X.; ZHU, M. A comprehensive comparison between hybrid and native app paradigms. In: *2016 8th International Conference on Computational Intelligence and Communication Networks (CICN)*. [S.l.: s.n.], 2016. Citado na página 35.

RAULINO, V. D. *Sustentaê: o aplicativo móvel em prol da sustentabilidade*. 67 p. Monografia (Monografia (Graduação em Publicidade e Propaganda)) — Universidade Católica de Brasília, 2017. Citado 5 vezes nas páginas 24, 28, 29, 30 e 55.

RAYWENDERLICH. *Model-View-Controller (MVC) in iOS: A Modern Approach*: Site. 2016. Disponível em: <https://www.raywenderlich.com/132662/mvc-in-ios-a-modern-approach>. Acesso em: 27 abr. 2018. Citado na página 37.

RISING, L. The scrum software development process for small teams. IEEE, 2000. Citado na página 34.

RODRIGUES, W. C. *Metodologia Cientiífica.* Paracambi: [s.n.], 2007. Citado na página 50.

RUIZ, J. *Metodologia Cientiífica: guia para eficiência nos estudos*. São Paulo: Atlas, 1996. Citado 2 vezes nas páginas 49 e 50.

SCHUMACHER, E. *Small is beautiful*. United States: Hartly, Marks Publishhers, 1999. Citado na página 23.

SCHWABER, K.; SUTHERLAND, J. *Um guia definitivo para o Scrum : As regras do jogo*. 2013. Disponível em: <https://www.scrumguides.org/docs/scrumguide/v1/Scrum-Guide-Portuguese-BR.pdf>. Acesso em: 27 abril de 2018. Citado 3 vezes nas páginas 34, 35 e 52.

SHOVE, E. Social theory and climate change. *Theory, Culture e Society*, v. 27, n. 2-3, p. 277–288, 2010. Disponível em: <https://doi.org/10.1177/0263276410361498>. Citado na página 23.

SLACK. *versão 2.8*. Slack Technologies, 2018. Disponível em: <https://slack.com/>. Citado na página 44.

SOMMERVILLE, I. *Software Engineering (6th Ed.)*. Redwood City, CA, USA: Addison Wesley Longman Publishing Co., Inc., 2000. ISBN 0-201-42765-6. Citado 3 vezes nas páginas 33, 36 e 39.

SPRINTS, Z. Zoho Corporation Pvt. Ltd, 2018. Disponível em: <https://www.zoho.com/sprints/>. Citado na página 43.

TAVARES, A. *Gerência de Projetos com PMBOK e SCRUM – Um estudo de caso*. Monografia (TCC) — Faculdade Cenecista Senhora dos Anjos, 2008. Citado na página 34.

TESTFLIGHT. Apple, Inc, 2018. Disponível em: <https://developer.apple.com/testflight/>. Citado na página 47.

TRODO, L. *Uso de Métricas nos Testes de Software*. Monografia (TCC) — Universidade Federal do Rio Grande do Sul, 2009. Citado na página 39.

TULLIS, T.; ALBERT, W. *Measuring the User Experience: Collecting, Analyzing, and Presenting Usability Metrics*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2008. ISBN 0123735580, 9780123735584. Citado 3 vezes nas páginas 76, 78 e 80.

ZOTERO. *versão 5.0.43*. George Mason University, 2018. Disponível em: <https://www.zotero.org/>. Citado na página 47.

Apêndices

APÊNDICE A – Ata de Reunião 001

Ata de Reunião 001

Data: 15/01/2018 **Hora:** 21h ás 22h **Local:** *Skype*

Participantes:

- Vanessa Raulino - Autora do TCC inspirador acerca do conceito do *Sustentaê*
- Gabriel Araújo - Autor desse TCC acerca do desenvolvimento do aplicativo *Sustentaê*
- Luís Resende - Autor desse TCC acerca do desenvolvimento do aplicativo *Sustentaê*

Relatório:

Fluxo de cadastro de novo achado (localização?):

- Localização igual do *Instagram*. Pesquisa com *auto-complete*.
- Achado de um achado já achado. Funciona como uma *hashtag*. Tudo fica no mesmo lugar.
- Estilo pesquisa, se for nova criar esse novo. Se não for incrementar valor de achados para esse achado.

Fluxo de cadastro de um novo perfil já existente na plataforma (Perfil dinâmico):

- Foi discutido mas não há uma decisão sobre o que deve ser feito. Vanessa irá ver e retornar. Ideia: coletâneas.

Perfil empresa (falta informações de quantos visitaram, quantos indicaram, quantos eu sigo)

- Ideia das fotos para autenticação (manual) ou por localização. Vanessa vai pesquisar sobre como funciona o processo de autenticação do *Instagram*.

Perfil empresa (na aba de postagem, são postagens somente do local? ou postagens de indicação também? aonde vemos as indicações do local, e total de numero de curtidas) Adicionar filtro de todas as indicações da empresa?

- Número de visitas do *post*: quantos foram para o lugar a partir dessa postagem. Colocar o total de visitas no perfil do lugar.

Feed (Tela principal do aplicativo) de empresa (O que vejo?)

- Postagens do lugar. Onde vejo as postagens do lugar: "fotos com você do *Instagram*" e na busca.
- *Feed*: mostrar as postagens dos usuários sobre a própria empresa
- Empresa não vai seguir ninguém

Dúvida na funcionalidade de mapa dentro do *feed* e da postagem (como funciona?)

- *Swipe* para mostrar um mapa. Imagem do mapa figurativa. Ao clicar abre o mapa para o local do *post*;

Dúvida na funcionalidade de busca de locais

- Postagens dos usuários, lugar no mapa e perfil do local

Tela de visualização de postagem da empresa

- Vanessa: "Vai ser bem parecido com esse [post de usuário]"

Criação de postagem da empresa (o que é o campo localização?)

- Remove campo de localização do *post* de empresa

Dúvida da funcionalidade de bloquear usuários (como funcionaria?)

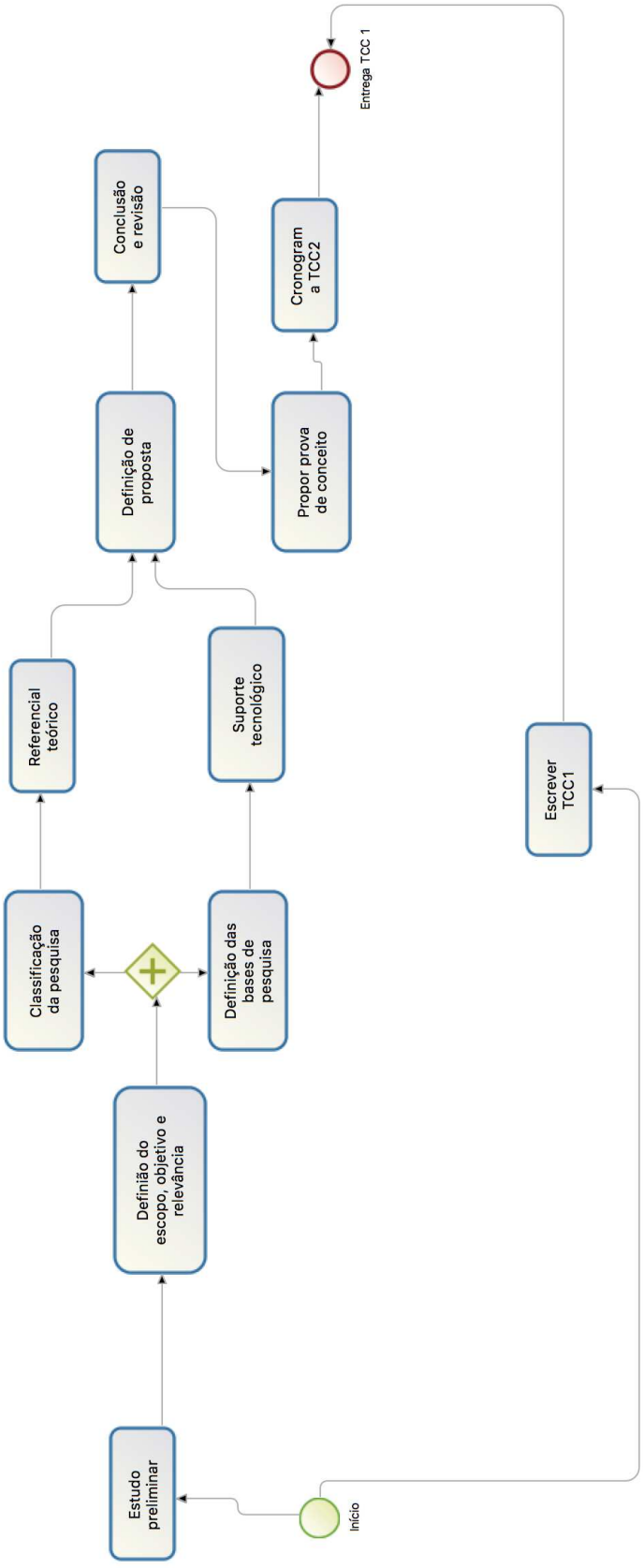
- Bloqueia um perfil selecionado: bloqueio funciona nas duas vias. Não ver *post* de nenhum nem outro.

Itens fora do escopo:

- Marcação nos comentários
- Perfil empresa

APÊNDICE B – Fluxo de atividades TCC 1

Figura 28 – Fluxo de atividades - TCC 1.



Fonte: Elaborada pelos autores.

APÊNDICE C – Roteiro do teste de usabilidade

Este roteiro foi criado para obter um melhor entendimento sobre as dificuldades que os participantes apresentam ao utilizar o aplicativo Sustentaê.

A seguir serão apresentadas atividades para você completar que contribuíram para obter esse entendimento.

Enquanto você estiver executando cada atividade, é importante saber o que está se passando em sua mente. Dessa forma, ao executar cada atividade, pedimos que diga para onde você está olhando, em que está pensando e o que lhe parece confuso.

TAREFA 1 Autentique-se no aplicativo através da sua conta do facebook

TAREFA 2 Procure pelo usuário Gabriel Araújo, veja seu perfil e siga-o

TAREFA 3 Vá ao seu *feed* e verifique as postagens. Escolha uma postagem para curtir, visualizar mais informações e comentar

TAREFA 4 Vá ao seu *feed* e salve uma postagem como favorito, depois verifique a sua lista de favoritos

TAREFA 5 Crie uma nova postagem para o local que você se encontra com as seguintes informações

- Avaliação: 4 folhas
- Descrição: Minha casa é o lugar mais sustentável
- Image: Tira uma foto
- Nome do achado: A casa do [seu nome]

CONSENTIMENTO

Afirmo que sou maior de 18 anos e desejo participar do estudo de usabilidade do aplicativo para iPhone Sustentaê. Todas as informações coletadas neste estudo são confidenciais, e meu nome não será identificado em momento algum. Estou ciente que posso fazer perguntas ou desistir da colaboração em qualquer momento, sem qualquer tipo de penalidade.

Assinatura do Participante: _____

Data:

APÊNDICE D – Resultado do teste de usabilidade

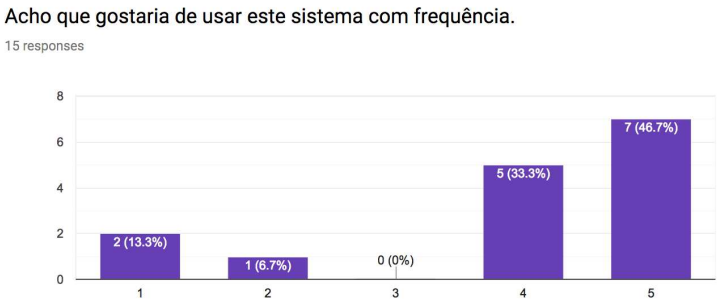
Tabela 7 – Resultado da execução das tarefas.

Usuário	Tarefa 1	Tarefa 2	Tarefa 3	Tarefa 4	Tarefa 5
1	1	1	1	1	1
2	1	1	1	1	1
3	1	1	1	1	1
4	1	1	1	1	1
5	1	1	1	1	1
6	1	1	1	1	1
7	1	1	1	1	1
8	1	1	1	1	1
9	1	1	1	1	1
10	1	1	1	1	1
11	1	1	1	1	1
12	1	1	1	1	1
13	1	1	1	1	1
14	1	1	1	1	1
15	1	1	1	1	1

Tabela 8 – Resultado do sucesso de execução das tarefas.

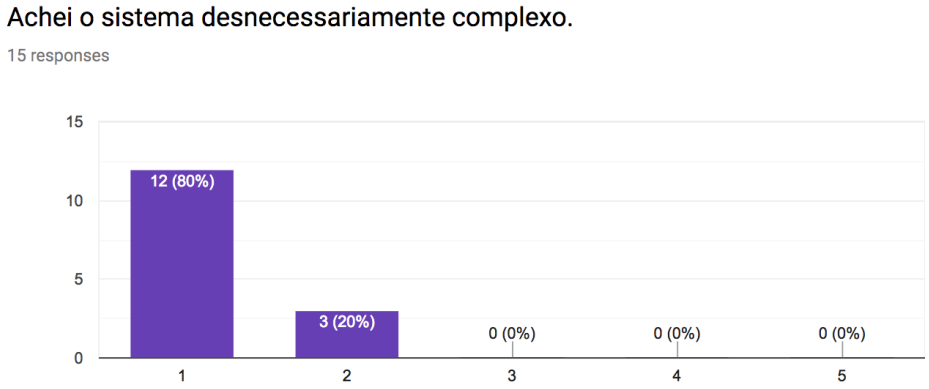
Usuário	Tarefa 1	Tarefa 2	Tarefa 3	Tarefa 4	Tarefa 5
1	1	1	1	1	1
2	1	1	1	1	1
3	0,5	0,75	0,5	0,75	0,75
4	0,75	1	0,25	0,75	0,75
5	1	1	1	1	1
6	1	1	1	1	1
7	1	1	1	1	1
8	1	1	1	1	1
9	0,5	1	0,25	1	0,75
10	1	1	0,5	0,75	0,75
11	1	1	1	1	1
12	1	1	1	0,75	1
13	1	1	1	1	1
14	0,75	1	1	0,5	1
15	1	1	0,75	1	1

Figura 29 – Resultado SUS - Pergunta 1.



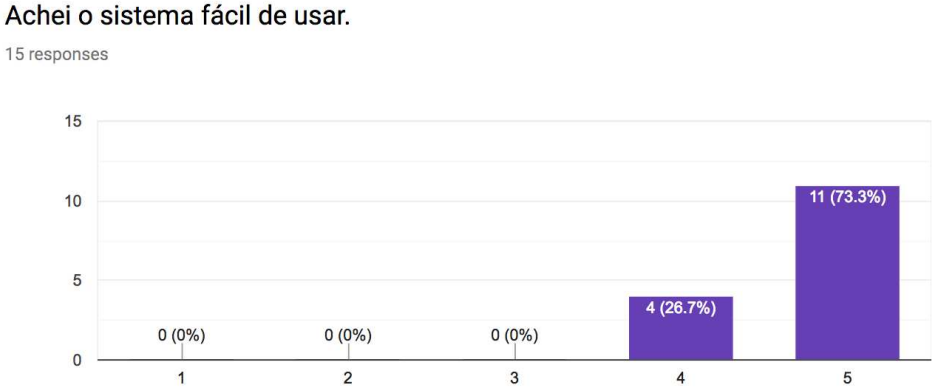
Fonte: Elaborada pelos autores.

Figura 30 – Resultado SUS - Pergunta 2.



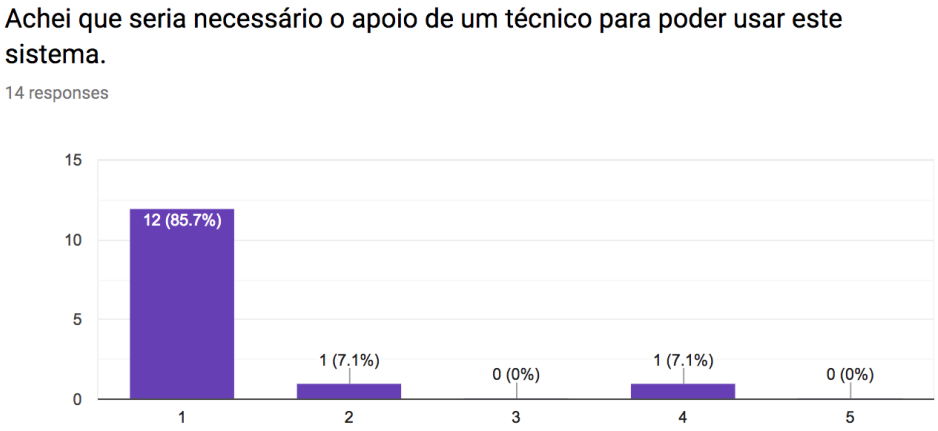
Fonte: Elaborada pelos autores.

Figura 31 – Resultado SUS - Pergunta 3.



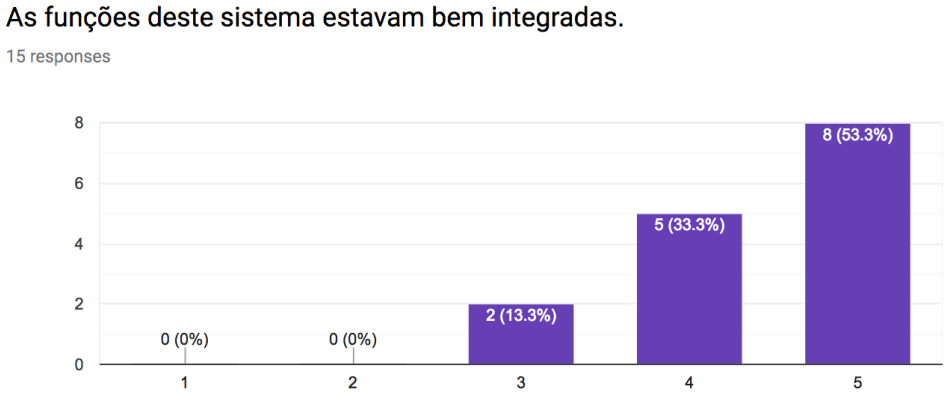
Fonte: Elaborada pelos autores.

Figura 32 – Resultado SUS - Pergunta 4.



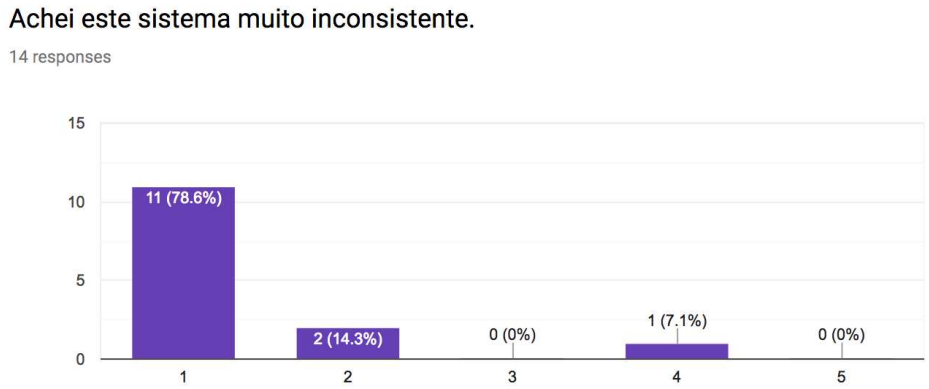
Fonte: Elaborada pelos autores.

Figura 33 – Resultado SUS - Pergunta 5.



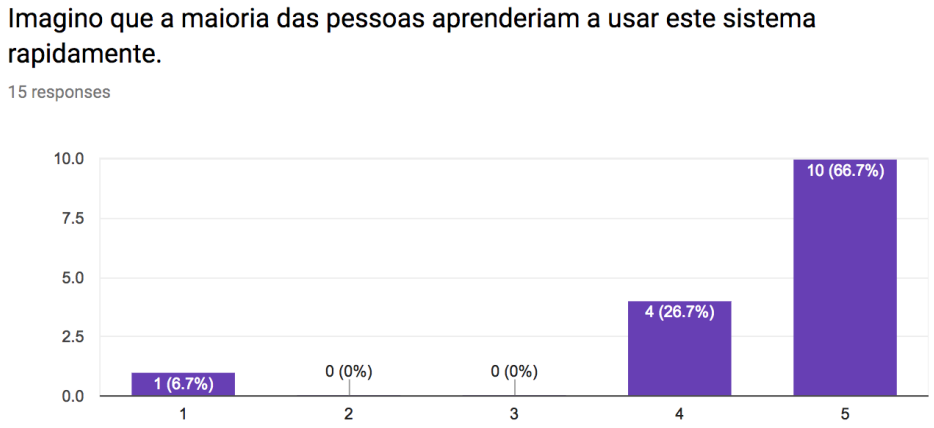
Fonte: Elaborada pelos autores.

Figura 34 – Resultado SUS - Pergunta 6.



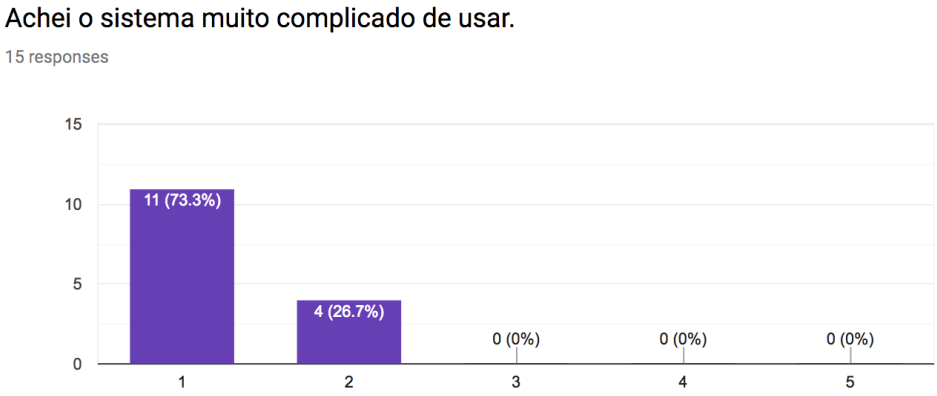
Fonte: Elaborada pelos autores.

Figura 35 – Resultado SUS - Pergunta 7.



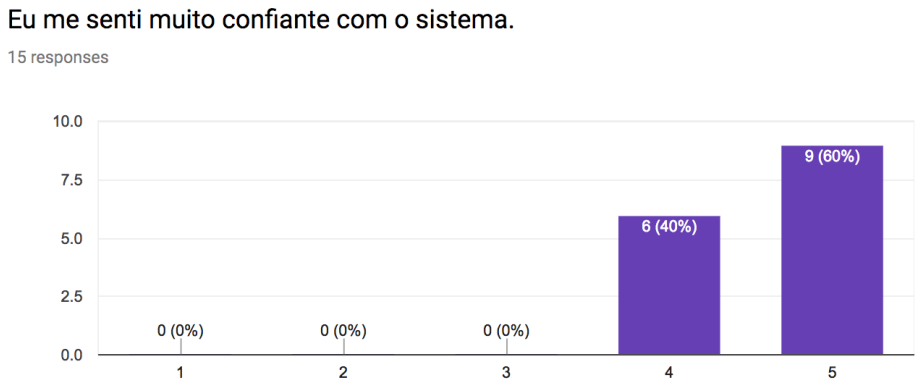
Fonte: Elaborada pelos autores.

Figura 36 – Resultado SUS - Pergunta 8.



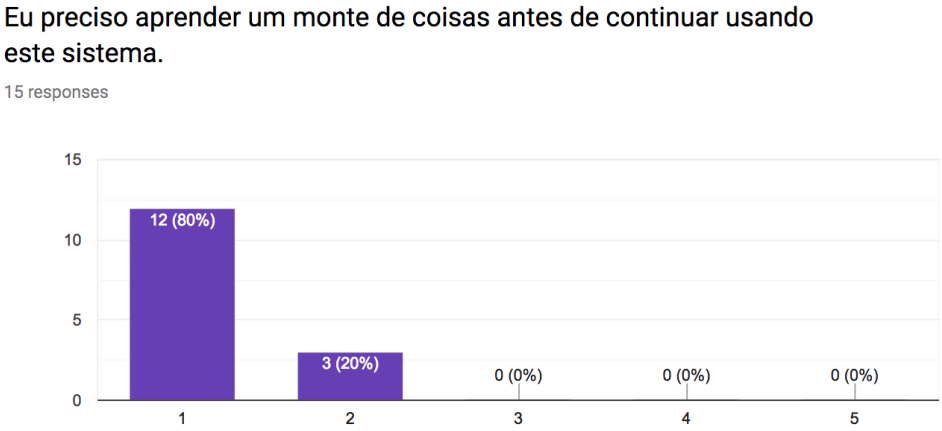
Fonte: Elaborada pelos autores.

Figura 37 – Resultado SUS - Pergunta 9.



Fonte: Elaborada pelos autores.

Figura 38 – Resultado SUS - Pergunta 10.



Fonte: Elaborada pelos autores.

APÊNDICE E – Exemplo de teste automatizado

Esse teste unitário visa validar que o mapeamento de uma model do tipo *Discovery* para um model do tipo *Post* é realizado de forma correta e que os dados antes do mapeamento continuam iguais após o mapeamento.

```
1
2  import XCTest
3  @testable import Sustentae
4
5  class SustentaeTests: XCTestCase {
6
7      var discovery: Discovery?
8
9      override func setUp() {
10         super.setUp()
11         // Put setup code here. This method is called before the
               invocation of each test method in the class.
12
13         let establishment = Establishment(name: "LocalTest",
14                                           location: PFGeoPoint(latitude:
                                                 0, longitude: 0))
15         let profile = Profile(withoutDataWithObjectId: "testId")
16         let images = [UIImage(named: "testUpload")!]
17
18         discovery = Discovery(content: "content",
19                               score: 10,
20                               establishment: establishment,
21                               profile: profile,
22                               images: images)
23     }
24
25     func testDiscoveryMapToPost() {
26         if let post = discovery?.getPostModel() {
27             assert(discovery?.content == post.text)
28             assert(discovery?.score == post.rating)
29             assert(discovery?.establishment.name == post.establishment?.
                   name)
30             assert(discovery?.establishment == post.establishment)
31             assert(discovery?.images.count == post.images?.count)
32             assert(discovery?.profile.objectId == post.profile?.objectId
                   )
```

```
        }
    }
}
```

Esse teste de integração visa validar a navegação principal do aplicativo através dos botões na parte debaixo da tela.

```
import XCTest

class SustentaeUITests: XCTestCase {

    var app = XCUIApplication()
    var tabBarsQuery: XCUIElementQuery?

    override func setUp() {
        super.setUp()
        continueAfterFailure = false

        XCUIApplication().launch()
        tabBarsQuery = app.tabBars
    }

    func testTabBarOpenFavorites() {
        tabBarsQuery?.buttons.element(boundBy: 4).tap()
        let labelElement = app.staticTexts["Favoritos"]
        XCTAssertEqual(labelElement.label, "Favoritos")
    }

    func testTabBarOpenSearch() {
        tabBarsQuery?.buttons.element(boundBy: 3).tap()
        let labelElement = app.staticTexts["sustentados de hoje"]
        XCTAssertEqual(labelElement.label, "sustentados de hoje")
    }

    func testTabBarOpenCreatePost() {
        tabBarsQuery?.buttons.element(boundBy: 2).tap()
        let labelElement = app.staticTexts["Novo Achado"]
        XCTAssertEqual(labelElement.label, "Novo Achado")
    }

    func testTabBarOpenFeed() {
        tabBarsQuery?.buttons.element(boundBy: 0).tap()
        let labelElement = app.staticTexts["Dashboard"]
        XCTAssertEqual(labelElement.label, "Dashboard")
    }

}
```