UNIVERSIDADE DE SÃO PAULO
ESCOLA DE ARTES, CIÊNCIAS E HUMANIDADES
PROGRAMA DE PÓS-GRADUAÇÃO EM SISTEMAS DE INFORMAÇÃO

ANDRÉ RAMOS MOREIRA

Arquitetura e modelo de ciclo de vida para aplicações de curta duração

São Paulo

2019

ANDRÉ RAMOS MOREIRA

Arquitetura e modelo de ciclo de vida para aplicações de curta duração

Dissertação apresentada à Escola de Artes, Ciências e Humanidades da Universidade de São Paulo para obtenção do título de Mestre em Ciências pelo Programa de Pós-graduação em Sistemas de Informação.

Área de concentração: Metodologia e Técnicas da Computação

Versão corrigida contendo as alterações solicitadas pela comissão julgadora em 24 de abril de 2019. A versão original encontra-se em acervo reservado na Biblioteca da EACH/USP e na Biblioteca Digital de Teses e Dissertações da USP (BDTD), de acordo com a Resolução CoPGr 6018, de 13 de outubro de 2011.

Orientador: Prof. Marcos Lordello Chaim

São Paulo

2019

Autorizo a reprodução e divulgação total ou parcial deste trabalho, por qualquer meio convencional ou eletrônico, para fins de estudo e pesquisa, desde que citada a fonte.

CATALOGAÇÃO-NA-PUBLICAÇÃO

(Universidade de São Paulo. Escola de Artes, Ciências e Humanidades. Biblioteca)
CRB-8 7012

Moreira, André Ramos

Arquitetura e modelo de ciclo de vida para aplicações de curta duração / André Ramos Moreira ; orientador, Marcos Lordello Chaim. – 2019.
92 f.

Dissertação (Mestrado em Ciências) - Programa de Pós-Graduação em Sistemas de Informação, Escola de Artes, Ciências e Humanidades, Universidade de São Paulo.
Versão corrigida

1. Engenharia de software. 2. Computação móvel. 3. Ciclo de vida de desenvolvimento de software. I. Chaim, Marcos Lordello, orient. II. Título.

CDD 22.ed.– 005.1

Dissertação de autoria de André Ramos Moreira, sob o título “**Arquitetura e modelo de ciclo de vida para aplicações de curta duração**”, apresentada à Escola de Artes, Ciências e Humanidades da Universidade de São Paulo, para obtenção do título de Mestre em Ciências pelo Programa de Pós-graduação em Sistemas de Informação, na área de concentração Metodologia e Técnicas da Computação, aprovada em 24 de abril de 2019 pela comissão julgadora constituída pelos doutores:

Prof. Dr. Marcos Lordello Chaim

Escola de Artes, Ciências e Humanidades - Universidade de São Paulo

Presidente

Prof. Dr. Auri Marcelo Rizzo Vincenzi

Universidade Federal de São Carlos

Prof. Dr. Luciano Vieira de Araújo

Escola de Artes, Ciências e Humanidades - Universidade de São Paulo

Prof. Dr. Márcio Katsumi Oikawa

Universidade Federal do ABC

Dedico à minha esposa linda Ricelli Moreira Silva Ramos

Agradecimentos

Estou imensamente feliz com a execução desse trabalho. Quando iniciei essa pesquisa há quatro anos não tinha visão de que o resultado seria tão satisfatório.

Gostaria de agradecer especialmente ao meu orientador que, além das horas de trabalho, me guiou durante essa pesquisa, na qual pude desenvolver diversas habilidades e adquirir conhecimentos valiosos.

Agradeço a Deus, à família, aos amigos, aos colegas e aos professores. A todos os que me apoiaram e me ajudaram das mais diversas formas o meu muito obrigado!

Por fim agradeço à Universidade de São Paulo na figura de todos os seus colaboradores, que, em defesa de uma sociedade melhor, se dedicam à nobre causa de transformar o mundo em um lugar mais justo e igualitário, pois só com a educação essa mudança é possível.

“Não é possível refazer este país, democratizá-lo, humanizá-lo, torná-lo sério, com adolescentes brincando de matar gente, ofendendo a vida, destruindo o sonho, inviabilizando o amor. Se a educação sozinha não transformar a sociedade, sem ela tampouco a sociedade muda.”

(Paulo Freire)

Resumo

MOREIRA, André Ramos. **Arquitetura e modelo de ciclo de vida para aplicações de curta duração**. 2019. 92 f. Dissertação (Mestrado em Ciências) – Escola de Artes, Ciências e Humanidades, Universidade de São Paulo, São Paulo, 2019.

Aplicativos simples e utilizados durante um curto período de tempo são cada vez mais comuns nos dispositivos móveis. Eles perdem a utilidade uma vez que seu objetivo específico é atingido. Porém, consomem memória e processamento do dispositivo, além de seu desenvolvimento consumir recursos humanos e tecnológicos. Para minimizar esses problemas, este trabalho apresenta uma nova abordagem para geração e controle desse tipo de aplicativo por meio de uma nova arquitetura e modelo de ciclo de vida para aplicações de curta duração. A abordagem permite que os aplicativos sejam gerados automaticamente e possam ser excluídos ou gerenciados de acordo com a vontade do usuário ou por outros critérios pré-estabelecidos (e.g., por tempo transcorrido ou por localização). A arquitetura e modelos desenvolvidos foram utilizados em três aplicativos típicos com funcionalidades básicas, *layout* trivial e processamento leve. Essa validação demonstrou que a abordagem atende às necessidades de aplicativos em diferentes domínios (restaurantes, *shoppings* e eventos) e pode ser aplicado em situações com diferentes restrições de tempo e localização.

Palavras-chaves: Aplicativos de curta duração. Geração automática de aplicativos. Gestão do ciclo de vida de aplicativos.

Abstract

MOREIRA, André Ramos. **Architecture and lifecycle model for short-lived apps**. 2019. 92 p. Dissertation (Master of Science) – School of Arts, Sciences and Humanities, University of São Paulo, São Paulo, 2019.

Simple applications that are used over a short period of time are becoming increasingly popular on mobile devices. They lose their usefulness once their specific goal is achieved. However, they consume memory and processing of the device; furthermore, its development consume human and technological resources. To minimize these problems, this work presents a new approach for generation and control of this type of application by means of a new architecture and life cycle model for short lived applications. The approach allows applications to be automatically generated, excluded or managed according to the user's wishes or other pre-established criteria (e.g., by time elapsed or by location). The architecture and models developed were used in three typical applications with basic functionalities, trivial layout and light-weight processing. This validation demonstrated that the approach addresses the needs of applications in different domains (restaurants, malls and events) and can be applied in situations with different time and location constraints.

Keywords: Short-lived apps. Automatic generation of applications. Application lifecycle management.

Lista de figuras

Figura 1 – Mapa mental	30
Figura 2 – Ciclo de vida de um aplicativo	40
Figura 3 – Arquitetura de alto nível	42
Figura 4 – JSON <i>Barra Azul Shopping</i>	46
Figura 5 – Lista de aplicativos	48
Figura 6 – Telas iniciais do aplicativos.	52
Figura 7 – Telas de geração do aplicativo.	53
Figura 8 – Tela inicial com aplicativo gerado	54
Figura 9 – Telas do aplicativo <i>shopping</i>	55
Figura 10 – Telas do ciclo de vida do aplicativo	56
Figura 11 – Aplicativo <i>Rackthon - 2019</i>	60
Figura 12 – JSON <i>Rackthon - 2019</i>	61
Figura 13 – Aplicativo <i>Saveur Restaurant</i>	62
Figura 14 – JSON <i>Saveur Restaurant</i>	64

Lista de tabelas

Tabela 1 – Atributos e dados para descrição de um aplicativo	45
--	----

Sumário

1	Introdução	14
1.1	<i>Justificativa</i>	15
1.2	<i>Objetivos</i>	17
1.3	<i>Hipótese</i>	17
1.4	<i>Organização</i>	18
2	Conceitos básicos	19
2.1	<i>Projeto de software</i>	19
2.2	<i>Conceitos e princípios de projeto de software</i>	20
2.2.1	Modularidade	20
2.2.2	Coesão	20
2.2.3	Acoplamento	21
2.3	<i>Princípios</i>	21
2.3.1	Abstração de dados e funções	21
2.3.2	Ocultamento da informação	22
2.3.3	Alta coesão, baixo acoplamento	22
2.3.4	Reúso	22
2.4	<i>Padrões de projeto</i>	23
2.5	<i>Estilos arquiteturais</i>	23
2.6	<i>Arquiteturas de referência</i>	24
2.7	<i>Arcabouços de software</i>	24
2.8	<i>Ecossistemas de software</i>	25
2.9	<i>Linhas de produtos de software</i>	26
2.10	<i>Considerações finais</i>	28
3	Trabalhos relacionados	29
3.1	<i>Seleção dos temas relevantes</i>	29
3.2	<i>Linhas de produto de aplicativos para dispositivos móveis</i>	31
3.3	<i>Metodologias e arcabouços para aplicativos de dispositivos móveis</i>	33
3.4	<i>Variações de “App Stores”</i>	34
3.5	<i>Outras formas de distribuição de software para dispositivos móveis</i>	34

3.6	<i>Barreiras, dificuldades e desafios das “App Stores”</i>	36
3.7	<i>Evolução do software baseada em seu uso e em comentários de usuários na “App Store”</i>	37
3.8	<i>Diversidade em software para dispositivo móvel</i>	38
3.9	<i>Considerações finais</i>	39
4	Modelo e arquitetura de ciclo de vida para aplicativos de curta duração	40
4.1	<i>Modelo de ciclo de vida</i>	40
4.2	<i>Arquitetura para aplicações de curta duração</i>	41
4.3	<i>Aspectos do desenvolvimento</i>	43
4.3.1	<i>Especificação do aplicativo</i>	43
4.4	<i>Leitor e tradutor de especificações</i>	47
4.4.1	<i>Gerenciador de aplicativos</i>	48
4.4.2	<i>Gerenciamento de ciclo de vida</i>	49
4.4.3	<i>Construção do aplicativo gerenciador</i>	49
4.5	<i>Aplicativo Barra Azul Shopping</i>	51
4.6	<i>Considerações finais</i>	57
5	Estudo de caso da aplicação do modelo	58
5.1	<i>Implementação de três exemplos</i>	58
5.1.1	<i>Exemplo 1: Barra Azul Shopping</i>	58
5.1.2	<i>Exemplo 2: Rackthon - 2019</i>	59
5.1.3	<i>Exemplo 3: Saveur Restaurant</i>	62
5.1.4	<i>Resultados da construção dos exemplos</i>	63
5.2	<i>Desafios e decisões de implementação</i>	65
5.2.1	<i>Funcionalidades exploradas</i>	66
5.3	<i>Por que foi escolhida a plataforma Android</i>	66
5.3.1	<i>Recursos nativos explorados</i>	66
5.4	<i>Discussão</i>	67
5.5	<i>Limitações</i>	69
5.6	<i>Considerações finais</i>	70
6	Conclusão	71

6.1	<i>Síntese do trabalho</i>	71
6.2	<i>Contribuições</i>	72
6.3	<i>Trabalhos futuros</i>	73
	Referências¹	74
	Apêndice A – Especificações dos exemplos utilizados nesse trabalho no formato JSON.	78

¹ De acordo com a Associação Brasileira de Normas Técnicas. NBR 6023.

1 Introdução

O uso dos *smartphones* da forma como conhecemos atualmente, segundo Martin *et al.* (2016), surgiu com o lançamento das lojas de aplicativos. Esse lançamento ocorreu em 2008, iniciado pela “App Store” da Apple e, em seguida, pela “Google Play Store”. Desde então ambas acumularam mais de 4 milhões¹ de aplicativos para *download* no terceiro trimestre de 2018.

Diversas aplicações nativas e comerciais eram utilizadas antes do lançamento das lojas de aplicativos, de forma pequena e pouco estruturada, porém, com o surgimento dessas lojas, a disseminação do uso de aplicativos foi incentivada por suas características positivas para o usuário final, como disponibilidade, compatibilidade, facilidade de uso e variedade no catálogo. Só a partir de 2008 os usuários puderam realmente explorar o poder de computação de seus dispositivos móveis e a versatilidade dos diversos tipos de aplicativos (MARTIN *et al.*, 2016).

Com o mundo ao alcance dos dedos, dispositivos móveis como *tablets* e *smartphones* tornaram-se uma das ferramentas mais necessárias no dia-a-dia das pessoas e desempenham um papel fundamental na vida do indivíduo (RAHMAN *et al.*, 2018). Essa tendência de uso tem crescido continuamente, com os indivíduos aumentando todos os anos o tempo gasto utilizando aplicativos².

Embora exista pouca informação pública sobre o comportamento do uso de aplicativos (BÖHMER *et al.*, 2011), pesquisas realizadas em 2017 mostram que mais 65% da população mundial utiliza dispositivos móveis³ e que um usuário médio gasta mais de três horas por dia com esses aparelhos⁴. Dessas três horas de uso diário, quase 90% do tempo é gasto em aplicativos⁵. No entanto, os dados mostram que o uso do dispositivo é monopolizado por alguns aplicativos chave (PEREZ, 2015), enquanto outros aplicativos são usados por um curto período ou, ocasionalmente, para atingir objetivos específicos.

Os aplicativos de curta duração podem ser utilizados para configurar um eletrodoméstico como ar-condicionado, geladeira ou aparelho de TV em um quarto de hotel, verificar uma lista de itens em um menu de restaurante ou ajudar turistas a localizarem-se

¹ Statista - disponível em goo.gl/6Fi6Vx acessado em 26.12.2018

² eMarketer - disponível em goo.gl/TRUKhr acessado em 22.05.2018

³ Digital in 2017: Global overview disponível em goo.gl/SABhPe acessado em 22.05.2018

⁴ Global Digital Future in Focus 2018 - disponível em goo.gl/51Zp1z acessado em 22.05.2018

⁵ Mobile Marketing Statistics disponível em goo.gl/Cpo96G acessado em 22.05.2018

em um parque. Assim, esses aplicativos perdem a utilidade uma vez que seu objetivo específico é atingido. No entanto, geralmente eles são mantidos no dispositivo, causando consumo de rede, uso desnecessário de espaço de armazenamento e possível comprometimento da segurança do dispositivo.

1.1 Justificativa

Os aplicativos de curta duração são simples e visam prover um serviço determinado durante um curto período de tempo. Muitas aplicações podem ser classificadas como de curta duração; porém, para esclarecer esse conceito, são apresentados a seguir exemplos típicos.

Exemplo 1. Em um evento de grande porte, como o *Hacktown*⁶, é necessário divulgar a programação que descreve os palestrantes, os locais, os horários e outras informações como hospedagens, transporte, alimentação, etc. Um aplicativo com essa funcionalidade é útil, pois facilita a participação e o acesso à informação, utiliza recursos simples e tem ciclo de vida limitado à duração do evento.

Exemplo 2. Em um restaurante, a consulta ao cardápio, a realização de pedidos e o pagamento podem ser feitos de forma automatizada via aplicativo. Depois da refeição, o aplicativo não se faz mais necessário e pode ser descartado.

Exemplo 3. Em um *shopping center*, pode-se utilizar um aplicativo para listar as lojas, sua descrição e localização. Além disso, o aplicativo pode divulgar ofertas, promoções, novos serviços oferecidos, entre outras informações de interesse dos clientes. Esse aplicativo pode ser limitado à região em que o estabelecimento está localizado e suas proximidades.

As situações apresentadas são hipotéticas, porém, comuns ao dia-a-dia de um indivíduo. A criação das aplicações, sua utilização e posterior exclusão pode ser feita de forma simples e automatizada de forma a economizar recursos do dispositivo móvel. Esses aplicativos possuem características comuns: trivialidade de funções, ciclo de vida limitado, conjunto de dados pré-existent, botões relacionados a ações e um *layout* corriqueiro. Com isso, é possível definir uma categoria de aplicativos que atendem situações similares e que podem utilizar os recursos oferecidos pela solução aqui proposta.

Na situação do exemplo 1, o aplicativo pode ser dividido da seguinte maneira: as funções incluem consulta à programação do evento e a informações de transporte,

⁶ Hacktown 2018 - <https://hacktown.com.br/> acessado em 08.01.2019

hospedagem, alimentação, etc.; o ciclo de vida é limitado à duração do evento; os dados consistem de datas, horários e programação do evento, bem como as informações de transporte, hospedagem e alimentação; os botões fazem buscas conforme demanda do usuário; e o *layout* é simples e pode ser reaproveitado para diversos eventos, com pequenas mudanças como, por exemplo, a inserção do logotipo do evento.

O aplicativo do exemplo 2 também pode ser dividido de forma semelhante: as funções permitem consulta às opções do cardápio e preços; o ciclo de vida é limitado à refeição; os dados são pratos, ingredientes, bebidas, preços e formas de pagamento; os botões realizam consultas conforme demanda do usuário; e o *layout* é simples e pode ser reaproveitado para diversos estabelecimentos, com pequenas mudanças como, por exemplo, a inserção de fotos dos pratos, imagem do restaurante, entre outros.

O aplicativo do exemplo 3 inclui consulta à localização das lojas, assim como a ofertas, serviços oferecidos, etc.; seu ciclo de vida é limitado à permanência no *shopping*; os dados são as informações das lojas, localização, promoções e serviços; os botões realizam buscas conforme demanda do usuário; e o *layout* é simples e pode ser reaproveitado para diversos *shoppings*, com pequenas mudanças como, por exemplo, a inserção do logotipo do estabelecimento ou de seus mapas.

Portanto, observa-se que os exemplos apresentados descrevem uma classe de aplicativos com características comuns que podem ser abstraídas em uma arquitetura para implementá-los e um modelo de ciclo de vida para gerenciá-los.

Além disso, existem diversos serviços para criação de aplicativos de funcionalidades simples, e não necessariamente com ciclo de vida de curta duração, que seguem o conceito de *click-and-drag* (clica e arrasta) para que os aplicativos sejam desenvolvidos pelo usuário comum, sem a necessidade de conhecimento de alto nível de programação, por exemplo, FlipGo⁷ e AppInventor⁸. Esses serviços diferem-se do modelo aqui proposto em dois aspectos fundamentais: (i) o gerenciamento de ciclo de vida do aplicativo automatizado e independente do usuário e (ii) a especificação que gera o aplicativo automaticamente, sem a necessidade de outras ações.

⁷ FlipGo - disponível em <https://comocriarumaplicativo.com.br/> acessado em 08.01.2019

⁸ AppInventor - disponível em <http://appinventor.mit.edu/explore/> acessado em 08.01.2019

1.2 Objetivos

O objetivo desta dissertação é apresentar uma arquitetura e um modelo de ciclo de vida para gerar aplicativos de curta duração de maneira automática, rápida e independente das lojas de aplicativos. Dessa maneira, pretende-se reduzir o consumo de memória e de rede, bem como a exposição a ameaças associadas aos aplicativos ativos de forma transparente ao usuário.

A arquitetura proposta permite que os aplicativos sejam gerados de maneira simplificada a partir de uma especificação concisa, traduzida internamente ao dispositivo no aplicativo desejado. O ciclo de vida proposto possibilita que os aplicativos gerados possam ser excluídos automaticamente ou gerenciados de acordo com a vontade do usuário.

Como objetivos específicos, buscou-se:

- Elaborar um modelo de ciclo de vida para aplicações de curta duração;
- Desenvolver uma arquitetura baseada nos conceitos de linhas de produto, arcabouços (*frameworks*) e ecossistemas de software para apoiar a geração de aplicativos de curta duração;
- Implementar o modelo de ciclo de vida e a arquitetura em um ambiente de execução de aplicativos em dispositivos móveis;
- Validar o modelo e a arquitetura por meio do desenvolvimento de aplicativos de curta duração.

1.3 Hipótese

A hipótese investigada neste projeto é a de que aplicativos cujo ciclo de vida é de curta duração, sendo úteis e necessários em determinados momentos, podem ser desenvolvidos de maneira rápida e gerenciados de modo seguro utilizando a arquitetura e o modelo de ciclo de vida para aplicações de curta duração.

Essa abordagem pode trazer diversos benefícios, tanto ao usuário final quanto ao desenvolvedor de aplicativos para dispositivos móveis. Para o usuário, os principais benefícios são (i) acesso rápido ao aplicativo, (ii) aplicativo específico para sua demanda, (iii) redução do consumo de banda de dados, (iv) redução do consumo de memória do dispositivo, (v) menor exposição a riscos de segurança e (vi) confiabilidade e qualidade do

aplicativo. Para o desenvolvedor, as principais vantagens são (i) rapidez no desenvolvimento por meio do reúso de código, (ii) facilidade na inclusão e teste de novas funcionalidades, (iii) independência de “App Store” para distribuição do aplicativo e (iv) maior alcance do público alvo do aplicativo, uma vez que ele teria a informação do código para geração do aplicativo nos ambientes em que ele é necessário.

1.4 Organização

Esse trabalho está organizado da seguinte forma. O Capítulo 2 apresenta os conceitos fundamentais: arquitetura de software, linhas de produto, arcabouços reutilizáveis e ecossistemas de software. O Capítulo 3 apresenta a revisão da literatura, com uma breve descrição dos artigos relacionados ao tema de pesquisa. O Capítulo 4 apresenta a arquitetura e o modelo de ciclo de vida para aplicações de curta duração. O Capítulo 5 apresenta um estudo de caso da aplicação da arquitetura e do modelo desenvolvidos. Por fim, o Capítulo 6 as conclusões, com a síntese do trabalho, as contribuições desenvolvidas e propostas para trabalhos futuros.

2 Conceitos básicos

Neste capítulo, os principais conceitos de arquitetura de software aplicáveis ao modelo e arquiteturas propostos são descritos. A arquitetura e o modelo de ciclo de vida para aplicações de curta duração baseiam-se nos conceitos de princípios e projetos de software, arquitetura, linhas de produtos de software, arcabouços (*frameworks*) e ecossistemas de software. Esses conceitos podem ser aplicados em alguma escala a qualquer projeto de software, independente das técnicas e tecnologias utilizadas. Eles são discutidos brevemente a seguir.

2.1 Projeto de software

Durante o projeto, procura-se detalhar a organização do software de forma a permitir que ele seja desenvolvido. Essa organização envolve a definição da estrutura do software, também conhecida como arquitetura. Durante o projeto ocorre também o detalhamento dos subsistemas e módulos, a definição das estratégias de armazenamento, gerenciamento e persistência dos dados, a interação humano-computador e, por fim, a implementação dos algoritmos que realizam a aplicação (BEZERRA, 2007).

Todos os métodos de desenvolvimento de software procuram fazer a transição dos requisitos definidos na análise para a criação de um conjunto de subsistemas e módulos que efetivamente implementam o software. Porém, a criação da arquitetura associada ao projeto é baseada em princípios que norteiam o desenvolvedor nessa atividade. Conceitos como reuso, modularidade, coesão e acoplamento são úteis para um desenvolvimento de aplicativos mais eficientes e com menores custos. Estilos arquiteturais, arcabouços (*frameworks*), linhas de produtos e ecossistemas de software são abordagens que permitem que ideias e soluções já provadas em aplicações anteriores possam ser reutilizadas com baixo custo em novos softwares.

Este capítulo inicialmente apresenta os conceitos de projetos e princípios na elaboração de um software. Em seguida são apresentados os estilos arquiteturais, as arquiteturas de referência, arcabouços e linhas de produto de software. Esses princípios e modelos foram utilizados na elaboração do projeto apresentado nesta dissertação.

2.2 Conceitos e princípios de projeto de software

A elaboração de um bom projeto de software não é uma atividade trivial, principalmente, porque somente se consegue desenvolver um projeto bom a partir da experiência prática. No entanto, há alguns conceitos subjacentes que ajudam a elaborar um projeto de software. Esses conceitos dão origem a princípios que servem como guia para identificar os principais componentes do projeto. Os conceitos e princípios descritos a seguir são válidos para diferentes níveis de abstração – subsistemas, módulos ou procedimentos – de um projeto de software.

2.2.1 Modularidade

O conceito de software modular surge em oposição ao de software monolítico. No limite, um software monolítico é aquele composto de um único módulo; porém, modernamente esse conceito foi estendido. Ele inclui programas que podem possuir módulos, mas nos quais não há acesso disciplinado aos dados, por exemplo, por meio de compartilhamento de memória, nem funções claras para utilizar os módulos separadamente, ou seja, reaproveitá-los. Em suma, software para o qual não há interfaces e acesso bem definidos entre as unidades lógicas do software (FOX *et al.*, 2015).

O conceito de modularidade já era praticado pelos desenvolvedores de software nos anos 60. Ele foi formalizado, porém, no início dos anos 70 com o trabalho de Parnas (1972). Esse trabalho preconiza que o software deve ser dividido em módulos com interfaces e funções bem definidas. A subdivisão do sistema em módulos permite que a manutenção do software seja facilitada, pois módulos com interfaces e funções claras são fáceis de entender e de alterar.

2.2.2 Coesão

A coesão indica o quão especializadas são as funções atribuídas a um módulo. O módulo deve ser construído de modo a conter um conjunto de funções que façam sentido e têm um objetivo comum. Os módulos com alta coesão, isto é, com um conjunto de funções altamente relacionadas é mais fácil de ser compreendido pelo desenvolvedor e de ser reutilizado em outros sistemas (BEZERRA, 2007).

Um módulo com baixa coesão possui dois ou mais grupos de atributos ou operações com alta correlação dentro de cada grupo, mas baixa correlação entre grupos. Uma melhor solução seria subdividi-los em dois módulos contendo cada um deles os grupos diferentes de atributos ou operações (BEZERRA, 2007). Módulos com baixa coesão são mais complexos e, por isso, menos reutilizáveis e mais difíceis de manter.

2.2.3 Acoplamento

O acoplamento é uma medida de quão fortemente um módulo está conectado a outros módulos, tem conhecimento ou depende de outros módulos. Um módulo com acoplamento fraco não depende de muitos outros. Por outro lado, um módulo com acoplamento forte é menos inteligível isoladamente e menos reutilizável. Além disso, um módulo com alto acoplamento é mais sensível a mudanças, quando é necessário modificar as classes das quais ele depende (BEZERRA, 2007).

2.3 Princípios

Os princípios fundamentais para a elaboração de um projeto de software são *abstração de dados e funções*; *ocultamento da informação*; e *alta coesão, baixo acoplamento*. Esses itens são apresentados brevemente a seguir.

2.3.1 Abstração de dados e funções

O princípio de *abstração de dados e funções* indica que os módulos de um projeto devem encapsular tanto os dados como as funções. Ou seja, o acesso aos dados deve ser controlado pelas funções dos módulos.

As linguagens de programação fornecem recursos para a implementação do princípio de abstração de dados e funções como as estruturas de classes e de aspectos disponíveis nas linguagens orientadas a objeto e a aspectos. Mais recentemente, as arquiteturas baseadas em serviços (SOA – *Service Oriented Architecture*) são fundamentadas nesse princípio ao definir serviços que encapsulam os dados e as funções.

2.3.2 Ocultamento da informação

Esse princípio é semelhante ao de abstração de dados e funções. Ele estabelece que os módulos devem controlar o acesso aos dados. Dessa forma, todo o acesso a qualquer dado deve ter uma interface de controle. O propósito é impedir que os dados de diferentes módulos sejam compartilhados entre eles.

2.3.3 Alta coesão, baixo acoplamento

A criação do projeto envolve a definição dos módulos que irão compor o software e também a relação entre eles. Os módulos devem ser altamente coesos, isto é, possuir um conjunto de funcionalidades relacionadas e focadas em tarefas específicas. Além disso, devem se relacionar com outros módulos para atingir o objetivo do sistema. No entanto, os relacionamentos entre módulos devem ser limitados, isto é, o acoplamento deve ser baixo. Isso porque um módulo que colabora com muitos módulos é de difícil compreensão e reutilização (GAMMA *et al.*, 1995).

Se um módulo possui alta coesão e baixo acoplamento pode ser entendido como um “mini sistema”, e pode ser desenvolvido separadamente por um grupo de programadores menor. Dessa maneira, é possível construir os módulos separados e em paralelo, reduzindo o tempo de desenvolvimento de um projeto de software.

2.3.4 Reúso

Um princípio de elaboração de projeto de software aplicado fortemente neste projeto é o reúso de soluções anteriores. Soluções que se mostraram efetivas e eficazes em projetos passados podem ser reutilizadas em vários níveis: no nível de subsistemas ou módulos, e no nível de classes.

O reúso pode ocorrer de várias maneiras, por exemplo, utilizando padrões de projeto para a definição das classes (GAMMA *et al.*, 1995), estilos arquiteturais para criar a estrutura mais geral do software (SHAW; GARLAN, 1996) e *frameworks* (MYLLÄRNIEMI *et al.*, 2018), linhas de produtos de software (CLEMENTS; NORTHROP, 2002) e ecossistemas de software (MYLLÄRNIEMI *et al.*, 2018) para reusar partes de código já implementadas.

O princípio de reúso, portanto, inclui os demais princípios de projeto. Ele é particularmente efetivo na definição das classes que fazem parte de um projeto e da arquitetura do sistema. A seguir, descreveremos sucintamente soluções já desenvolvidas para a definição da arquitetura, conhecidas como padrões de projeto, estilos arquiteturais, e soluções parcialmente prontas para o projeto de software como arcabouços, linhas de produto e ecossistemas de software.

2.4 *Padrões de projeto*

Os padrões de projeto de software surgiram embasados em padrões de projetos arquitetônicos de construções. Eles propõem utilizar soluções conhecidas e já aplicadas para problemas no projeto arquitetural. A maioria dos problemas nesses projetos são recorrentes e, portanto, soluções aplicadas anteriormente podem ser reusadas, sem a necessidade da criação de novas soluções.

Em Gamma *et al.* (1995) é utilizado o conceito de reúso para resolver problemas semelhantes que aparecem frequentemente durante o desenvolvimento de software. Os autores catalogaram 23 padrões de projeto divididos nas seguintes categorias:

Criacionais: separam a operação de uma aplicação de como os seus objetos são criados;

Estruturais: definem uma estrutura de classes geral que pode ser estendida no futuro; e

Comportamentais: utilizam herança para distribuir o comportamento entre subclasses, ou agregação e composição para construir comportamento complexo a partir de componentes mais simples.

2.5 *Estilos arquiteturais*

A *arquitetura de software* descreve como os subsistemas que compõem uma parte do software estão interconectados para atenderem aos requisitos funcionais e não-funcionais da aplicação (FOX *et al.*, 2015).

Um estilo arquitetural define uma família de sistemas em termos de um padrão organizacional dos seus módulos. Mais especificamente, um estilo arquitetural define um vocabulário de componentes e de tipos de conectores, bem como um conjunto de limitações no modo como são conectados (SHAW; GARLAN, 1996).

Difícilmente, o projeto de um software baseia-se em um único estilo arquitetural. Na prática, a arquitetura é composta por vários estilos arquiteturais. Alguns desses estilos são filtros e tubos, chamada-retorno, repositório, *broadcast*, camadas, cliente-servidor, entre outros (SHAW; GARLAN, 1996).

2.6 Arquiteturas de referência

A arquitetura de um sistema é o conjunto das principais decisões tomadas ao projetar um sistema. Tais decisões abrangem preocupações estruturais como: componentes, conectores e configurações; a implantação do sistema; suas propriedades não-funcionais; e os padrões de evolução do sistema, incluindo a adaptação em tempo de execução. As arquiteturas de software consideradas particularmente úteis para famílias de sistemas geralmente são codificadas em padrões arquitetônicos, estilos arquitetônicos e arquiteturas de referência reutilizáveis e parametrizadas (MEDVIDOVIC; TAYLOR, 2010).

As arquiteturas de referência são estilos arquiteturais para domínios específicos, como o aeroespacial e o automobilístico. São padrões de arquitetura que abordam os requisitos específicos para determinados tipos de sistemas. Esses estilos incluem módulos e conexões próprios para esses domínios que podem ser reutilizados em uma classe de sistemas (SHAW; GARLAN, 1996).

2.7 Arcabouços de software

Arcabouços de software, também denominados *frameworks* de desenvolvimento de aplicações, especificam um projeto da arquitetura reutilizável e a implementação parcial de uma aplicação. Os arcabouços geralmente não são utilizáveis por ou visíveis aos usuários finais, pois existem principalmente para atender aos desenvolvedores facilitando a construção de aplicativos (MYLLÄRNIEMI *et al.*, 2018).

Atualmente, os arcabouços de software são usados extensivamente para desenvolver eficientemente diferentes tipos de aplicativos de software. Por exemplo, o uso de arcabouços como Spring¹, Django², Node.js³ ou Angular.js⁴, é a abordagem padrão para o desenvolvi-

¹ Spring Framework - disponível em <http://spring.io/projects/spring-framework> acessado em 02.01.2019

² Django Project - disponível em <https://www.djangoproject.com> acessado em 02.01.2019

³ Node.js - disponível em <https://nodejs.org/en> acessado em 02.01.2019

⁴ Angular.js - disponível em <https://angularjs.org> acessado em 02.01.2019

mento de software da web. Da mesma forma, os aplicativos para dispositivos móveis são desenvolvidos usando uma estrutura de aplicativo; exemplos incluem PhoneGap⁵ e Ionic⁶.

Selecionar um arcabouço é uma decisão crítica de projeto em qualquer desenvolvimento de aplicativo. Isso ocorre porque a estrutura selecionada ditará a arquitetura da aplicação (MYLLÄRNIEMI *et al.*, 2018).

Os arcabouços de software também apresentam desvantagens. Como são estruturas complexas e envolvem projetos específicos, elas são difíceis de aprender e podem exigir um extenso treinamento prático. A adequação de uma estrutura para uma aplicação específica pode não ser aparente até que algum investimento em aprendizagem já tenha ocorrido (MYLLÄRNIEMI *et al.*, 2018).

2.8 Ecossistemas de software

Um ecossistema de software é definido como “um conjunto de negócios funcionando como uma unidade e interagindo com um mercado compartilhado de software e serviços, juntamente com os relacionamentos entre eles. Nesse contexto, os relacionamentos são frequentemente sustentados por uma plataforma ou mercado tecnológico comum” (MYLLÄRNIEMI *et al.*, 2018).

Assim, um ecossistema de software existe em torno de uma tecnologia central que é muitas vezes referida como uma plataforma de software. Dentro dos ecossistemas de software, a plataforma é um recurso tecnológico chave. A plataforma pode ser, por exemplo, uma aplicação de usuário final, um sistema operacional, uma arquitetura de sistema padrão ou um protocolo de comunicação.

A plataforma pode ser usada diretamente por usuários finais (por exemplo, o ecossistema do “Google Drive”⁷), mas também pode ser usada principalmente por outros desenvolvedores de software (por exemplo, o ecossistema de *plug-ins* do Eclipse⁸ ou o ecossistema de *plug-ins* do Apache⁹). Às vezes, a plataforma se assemelha a uma infraestrutura utilizada durante o desenvolvimento de software em diferentes domínios. Uma propriedade importante da plataforma de software é que ela é extensível além das

⁵ Adobe PhoneGap - disponível em <https://phonegap.com> acessado em 02.01.2019

⁶ Ionic Framework - disponível em <https://ionicframework.com/> acessado em 02.01.2019

⁷ Google Drive - disponível em drive.google.com/ acessado em 11.01.2019

⁸ Eclipse - disponível em <https://www.eclipse.org/> acessado em 11.01.2019

⁹ Apache - disponível em <https://www.apache.org/> acessado em 11.01.2019

fronteiras da organização que a controla e que é importante para a interação dentro do ecossistema.

O proprietário da plataforma precisa fornecer recursos para que desenvolvedores externos possam criar aplicativos no topo da plataforma de software. Esses recursos são chamados de recursos de limite de plataforma. Os recursos de limite da plataforma são uma maneira de o proprietário da plataforma controlar e influenciar o que acontece no ecossistema de software, enquanto se beneficia da geração de desenvolvimento de aplicativos externos. Além disso, os recursos de limite da plataforma minimizam o esforço de coordenação entre os desenvolvedores de aplicativos e o proprietário da plataforma. Por fim, os recursos de limite da plataforma permitem a coordenação entre os desenvolvedores de aplicativos. Isso é especialmente crucial quando as empresas que criam aplicativos semelhantes querem colaborar diretamente entre si (MYLLÄRNIEMI *et al.*, 2018).

Existem três tipos diferentes de recursos de limite de plataforma. Os recursos de limite de aplicativos expõem os serviços da plataforma aos aplicativos e são usados pelos aplicativos em tempo de execução. Exemplos incluem APIs (*Application Programming Interface* - Interface de Programação de Aplicativos), bibliotecas e binários. Os recursos de limite de desenvolvimento são as ferramentas fornecidas aos desenvolvedores para apoiá-los no desenvolvimento de aplicativos. Os exemplos incluem ambientes de desenvolvimento integrados (IDEs), kits de desenvolvimento de software (SDKs), depuradores, compiladores, editores e repositórios de aplicativos. E os recursos de limite social transferem as informações necessárias para o desenvolvimento dos aplicativos e explicam a plataforma aos desenvolvedores. Os exemplos incluem documentação, material de treinamento, eventos, diretrizes de desenvolvimento, direitos de propriedade intelectual, acordos entre o proprietário da plataforma e desenvolvedores de aplicativos, mídia social e fóruns *on-line*. Recursos de fronteira social também permitem interação dentro da comunidade de desenvolvimento; desenvolvimento de software é, em última instância, uma atividade social (MYLLÄRNIEMI *et al.*, 2018).

2.9 Linhas de produtos de software

Uma linha de produtos de software (LPS) é um grupo de sistemas que compartilham um conjunto comum e gerenciado de recursos que satisfazem necessidades particulares

de um mercado ou de uma missão e desenvolvidos de maneira prescrita (BASS *et al.*, 1997; CLEMENTS; NORTHROP, 2002). As LPSs também podem ser definidas como famílias de sistemas de software que possuem funcionalidades comuns e funcionalidades variáveis (SHEN; PENG; ZHAO, 2012). As principais vantagens tangíveis e intangíveis para o uso do LPS são redução do *time-to-market*, redução de defeitos, satisfação do cliente, e comprometimento dos recursos.

Os *core assets* (CAs) formam a base de uma LPS. Eles incluem a arquitetura, componentes de software reusáveis, requisitos, especificações, documentação, modelos de desempenho, planos de teste, casos de teste, entre outros artefatos. Os CAs podem ser desenvolvidos para a LPS ou podem ser extraídos a partir de produtos existentes. O desenvolvimento de *core assets* é chamado de engenharia de domínio, pois uma LPS visa atender um conjunto de produtos para um dado domínio de aplicação.

O desenvolvimento de produtos recebe como entrada os requisitos do produto individual, o escopo da LPS e seus CAs e um plano de produção. A partir deles, os CAs são ajustados para criar o produto desejado. Esse ajuste pode requerer alterações dos CAs atuais ou mesmo a criação de código adicional para requisitos específicos do novo produto. O plano de produção define como esse processo de derivação do novo produto a partir da LPS irá ocorrer. Há um laço de realimentação nesse processo, pois o novo produto pode levar à criação de novos CAs, que podem alterar o escopo da LPS e a própria especificação do novo produto (JAAKSI, 2002).

As LPSs apresentam várias vantagens: aumentam a qualidade, ganham escala ao reutilizar vários recursos (os CAs) comuns a vários produtos, aumentam o *time-to-market* de produtos, reduzem a necessidade de mão de obra qualificada, entre outras. As desvantagens principais são o custo inicial de gerar a LPS, o risco de não identificar os CAs adequados e a possível inibição de novas soluções para problemas conhecidos. As LPSs têm sido aplicadas com sucesso em produtos no domínio militar, de telecomunicação, de impressoras, entre outros.

2.10 Considerações finais

Projetar um software — e, em particular, sua arquitetura — não é uma tarefa trivial. Em geral, produzir bons projetos de software requer experiência prática e o conhecimento dos conceitos e princípios de projeto de software.

A arquitetura e o modelo de ciclo de vida de software propostos nesta pesquisa baseiam-se nos conceitos de arquitetura de software, linhas de produtos, arcabouços (*frameworks*) e ecossistemas de software.

Por isso, eles foram apresentados sucintamente neste capítulo juntamente aos principais conceitos e princípios associados ao projeto e arquitetura de software. No próximo capítulo, é apresentada a revisão de literatura de temas relacionados com esta dissertação.

3 Trabalhos relacionados

A pesquisa abrange diferentes temas, a saber, linhas de produtos e arcabouços para aplicativos para dispositivos móveis e mecanismos de distribuição como lojas de aplicativos e outras formas de distribuição. Por isso, a revisão bibliográfica apresentada a seguir foi organizada pelos temas relacionados à proposta de pesquisa.

Inicialmente, na Seção 3.1 será descrito o processo que levou à seleção dos temas relevantes para a condução da pesquisa e, em seguida, são apresentados os trabalhos mais expressivos em cada tema. O capítulo é concluído com a Seção 3.9 que apresenta uma discussão sobre a relação entre esses estudos e o projeto de pesquisa.

3.1 Seleção dos temas relevantes

As primeiras pesquisas realizadas para a revisão bibliográfica apresentaram um grande desafio para a definição dos termos de pesquisa. A natureza abrangente dessa pesquisa, que unifica diferentes áreas de engenharia de software e ciência da computação, contribuiu para, inicialmente, uma pesquisa nas bases tradicionais retornasse uma quantidade excessiva de estudos. Ao lê-los, percebeu-se que eram pouco relacionados ao tema, o que levantou o questionamento se os termos de pesquisa eram os mais adequados.

Para determinar quais termos seriam adequados para a pesquisa foi adotado o uso de uma técnica chamada mapa mental. Nessa técnica, é desenvolvido um tipo de diagrama que objetiva a gestão de informações, de conhecimento e de capital intelectual, também a compreensão e solução de problemas, entre outros vários possíveis usos. (BUZAN; BUZAN, 2010).

Os mapas mentais procuram representar com o máximo de detalhe o relacionamento conceitual existente entre informações fragmentadas, difusas e pulverizadas. É uma ferramenta para ilustrar ideias e conceitos, dar-lhes forma e contexto, traçar os relacionamentos de causa, efeito, simetria ou similaridade, e torná-las mais palpáveis e mensuráveis (DEBASTIANI, 2016).

A construção do mapa deu-se a partir da pesquisa dos termos relacionados inicialmente utilizados na pesquisa. Após a análise e leitura dos principais artigos retornados,

novos termos e relacionamentos foram incluídos no mapa. Com isso, foi possível gerar o mapa da Figura 1.

Figura 1 – Mapa mental

Fonte: André Ramos Moreira, 2019

Na Figura 1 estão mapeados os principais termos ligados à pesquisa e seus relacionamentos. Os termos principais, destacadas nas caixas cinzas, foram elencados por meio de pesquisas exploratórias nos artigos relacionados ao buscar o termo da caixa cinza anterior,

por exemplo, o termo *Plataforma* foi obtido com base na leitura dos artigos que retornaram da pesquisa do termo *Distribuição de Software*. Adicionalmente estão representados nessa figura termos, problemas e artigos identificados que conduziram a pesquisa pelas outras áreas de menor relevância. Esses termos estão destacados nas arestas da representação e os artigos estão identificados na figura.

Embasados no mapa apresentado na Figura 1, chegou-se a um conjunto de termos de pesquisa expandido, o que garantiu que nenhum termo importante ficasse de fora da pesquisa. Com base nos resultados obtidos pelo mapa mental, foi possível determinar os termos principais relacionados a essa pesquisa.

Os termos pesquisados foram: “Dedicated App Store” (Loja de aplicativos dedicada), “Software distribution App Store” (Distribuição de software em loja de aplicativos) e “Software distribution product line” (Linha de produtos de software e distribuição). Esses termos foram combinados, submetidos aos motores de busca de artigos tradicionais e trouxeram como resultados vários trabalhos diretamente (ou indiretamente) relacionados ao projeto aqui proposto. Esses artigos foram analisados e serão brevemente discutidos nas sessões seguintes. A opção pela discussão breve dos trabalhos pesquisados dá-se em função da limitada relação entre os trabalhos e o projeto aqui proposto, assunto explorado na Seção 3.9.

3.2 Linhas de produto de aplicativos para dispositivos móveis

Em Jaaksi (2002), é apresentada a utilização de uma linha de produto de software para a empresa Nokia com o objetivo de desenvolver navegadores para dispositivos móveis. A tecnologia foi desenvolvida inicialmente para os aparelhos dos autores e, em seguida, distribuída como um produto de software. A LPS aumentou os custos em várias funções de suporte, trabalho arquitetônico e gerenciamento. Os autores sugerem, portanto, que uma organização considere iniciar uma linha de produtos somente quando ela objetiva a reutilização sistemática e o atendimento de uma base de clientes heterogênea com um domínio comum. Com base nas experiências de Jaaksi (2002), é possível conseguir níveis bastante altos de reutilização sem desenvolver e manter uma linha de produtos inteira, por exemplo, usando componentes de código independentes. É possível atender melhor uma base de clientes homogênea, adaptando um único produto para diferentes fins.

Em Pascual *et al.* (2015), explora-se o uso de algoritmos evolutivos multiobjetivos (AEMO) para gerar configurações otimizadas em tempo de execução das Linhas de Produtos de Software Dinâmico (LPSD) de acordo com critérios diferentes. O problema de otimização é formalizado em termos de um modelo de características (MC) que mapeia as variabilidades. São avaliados seis AEMOs existentes, aplicando-os a 12 MCs diferentes, para otimização de três objetivos (usabilidade, consumo de bateria e uso de memória). Os resultados são discutidos de acordo com os requisitos particulares de um serviço de reconfiguração dinâmico (SRD) para aplicações para dispositivos móveis, mostrando que PAES (KNOWLES; CORNE, 2000) e NSGA-II (DEB *et al.*, 2002) são os algoritmos mais adequados para ambientes móveis.

Mizouni *et al.* (2014) propõem uma nova estrutura para desenvolvimento de aplicações para dispositivos móveis. Baseado em conceitos de modelagem de recursos e linhas de produtos de software (LPS), o *framework* proposto guia a modelagem de adaptabilidade em tempo de projeto e oferece apoio ao contexto e adaptabilidade em tempo de execução. É um metamodelo de características que incorpora, além dos conceitos de LPS, as prioridades das funções de aplicação para impulsionar a adaptabilidade. Uma ferramenta, baseada nesse metamodelo, é apresentada para modelar os recursos do aplicativo móvel e para derivar os membros da LPS. Também é descrita uma estrutura móvel, construída em cima do *framework* proposto, para adaptar dinamicamente a aplicação em tempo de execução. Mizouni *et al.* (2014) mostram que a especificação de requisitos de contexto, além de modelagem de recursos, pode ser usada para melhorar a seleção de recursos em tempo de execução.

Greenfield e Short (2003) combinam desenvolvimento baseado em componentes, desenvolvimento orientado a modelos e linhas de produtos de software para formar uma abordagem de desenvolvimento de aplicativos baseada no conceito de fábricas de software. Esta abordagem promete maiores ganhos de produtividade e previsibilidade do que aqueles produzidos por melhorias incrementais para o paradigma atual de orientação a objeto, que não acompanharam a inovação na tecnologia da plataforma. As fábricas de software prometem tornar a montagem das aplicações mais rentável por meio da reutilização sistemática, possibilitando a formação de cadeias de suprimentos e permitindo a customização em massa.

Fang *et al.* (2015) apresentam uma abordagem baseada em modelo para a derivação de sistemas em um dado domínio. A arquitetura proposta da infraestrutura de derivação

permite que a configuração de características, topologias e processos seja integrada no processo de derivação multietapas. O trabalho descreve um protótipo para provar a viabilidade e a melhoria da eficiência da derivação. Foi feita uma avaliação com entrevistas com as partes interessadas do domínio. Os resultados mostram um alto potencial para melhorar a eficiência da derivação, adotando a abordagem na prática.

Quinton *et al.* (2012) propõem a utilização da Configuração de Engenharia de Linha de Produto de Software (ELPS) e do Modelo de Características (MC) para desenvolver uma ferramenta de suporte à decisão. Usando essas técnicas de modelagem e automação, essa ferramenta leva em consideração os requisitos técnicos da aplicação, bem como os requisitos de qualidade do usuário para fornecer um resultado exato entre as soluções em nuvem que melhor atendam a todos os requisitos.

3.3 Metodologias e arcabouços para aplicativos de dispositivos móveis

Abrahamsson *et al.* (2004) apresentam uma abordagem de desenvolvimento ágil chamada Mobile-D. A abordagem Mobile-D é brevemente descrita e as experiências adquiridas com quatro estudos de caso são discutidas. Ao aplicar a abordagem Mobile-D na prática, as seguintes observações positivas foram feitas: maior visibilidade do progresso, identificação precoce e resolução de problemas técnicos (como dificuldades com as plataformas J2ME (KEOGH; TOPLEY, 2003) e Symbian (HARRISON, 2003)), responsabilidade dividida, compartilhamento eficiente de informações, alta coerência de práticas de processos, baixa frequência de defeito em produtos lançados e ritmo de desenvolvimento constante.

Biegel e Cahill (2004) desenvolveram um arcabouço (*framework*) que facilita a construção de aplicações para dispositivos móveis, compatíveis com o contexto. A estrutura permite aos desenvolvedores fundir dados de sensores diferentes, representar o contexto da aplicação e planejar o software de maneira eficiente em relação ao contexto, sem a necessidade de escrever um código complexo. É incorporado um paradigma de comunicação baseado em eventos projetado especificamente para ambientes sem fio *ad-hoc*, que dá apoio ao acoplamento entre sensores, atuadores e componentes de aplicativos. O *framework* foi aplicado com sucesso no desenvolvimento de uma série de aplicativos como prova de conceito, incluindo uma aplicação sensível que dirige e obedece aos sinais de trânsito de forma autônoma.

No trabalho de Francese, Risi e Tortora (2015) descreve-se como os “MicroApps” – aplicativos para dispositivos móveis criados pelo usuário, projetados compondo as funcionalidades oferecidas pelas várias aplicações do telefone – são gerenciados pela “MicroApp Store”, uma plataforma de distribuição digital para “MicroApps”, com o objetivo de capacitar os usuários finais, permitindo que eles não apenas criem novas aplicações, mas também compartilhem com outros usuários. Assim, cada usuário pode baixar os modelos de aplicativos gráficos projetados por outros e reutilizá-los para criar novas aplicações personalizadas, sem a exigência de habilidades de programação.

3.4 *Variações de “App Stores”*

O trabalho de Kudla *et al.* (2015) apresenta um processo de desenvolvimento que permite desenvolvedores independentes escrever software para o domínio militar. Com isso, reduz-se a barreira para a entrada nesse mercado, mantendo-se os altos padrões de segurança. Wenzel (2014) realizou um estudo de caso comparativo de duas “Apps Stores Empresariais” (ASE), uma interna e outra externa. Com esse estudo, foram identificadas diversas recomendações para uso corporativo para melhor atender às expectativas das empresas que as utilizarão. Concluiu-se que ambas “App Stores Empresariais”, externa e interna, possuem vantagens e desvantagens sendo que a recomendação de uma ou outra depende do perfil da empresa.

O trabalho de Beimborn e Palitza (2013) explora o conceito de “App Stores Cooperativas” (ASC) e desenvolve um quadro de benefícios que pode servir de base para a conceitualização e a investigação empírica do valor de ASCs e a adoção por outras empresas. Nesse trabalho, é demonstrada a importância do uso de dispositivos móveis e softwares, assim como de suas formas de distribuição, para o desenvolvimento das companhias, e como fazê-lo utilizando ASCs para reduzir os custos, melhorar os processos e ser mais eficiente.

3.5 *Outras formas de distribuição de software para dispositivos móveis*

Al-Akkad e Raffelsberger (2014) questionam a distribuição de aplicativos sem acesso a serviços na nuvem. Enquanto na vida cotidiana as pessoas adquirem aplicativos através de lojas de aplicativos, o acesso a essas lojas pode ser severamente dificultado em

desastres. Foram investigados cinco mecanismos potenciais de implantação de aplicativos de forma *ad-hoc*. Cada abordagem é avaliada de acordo com seis critérios e considera que o dispositivo móvel é a plataforma de implantação minimamente necessária. Este artigo conclui com a observação de que os provedores atuais de sistemas operacionais móveis não fornecem mecanismos de implantação “sem internet”, embora plataformas como o Android nativo permitam esse tipo de implantação.

Pahl e Carle (2013) propõem fornecer aos usuários ferramentas para o desenvolvimento comunitário de serviços de *drivers* e orquestração. São analisados os requisitos para uma estrutura de *middleware* que permita o desenvolvimento distribuído. São apresentadas as extensões necessárias para a construção do software baseada na comunidade: (i) um repositório para definições de interface, (ii) recursos de “App Store” e “App Manager”, e (iii) classificações multidimensionais que contemplam diferentes dimensões dos softwares desenvolvidos. Finalmente, mostra-se como o desenvolvimento inteligente de software espacial pode ser facilitado usando o sistema de orquestração de espaço inteligente distribuído (*Distributed Smart Space Orchestration System* - DS2OS), uma arquitetura para desenvolvimento de software comunitário para espaços inteligentes.

Ahmet e Holmquist (2010) realizaram estudos sobre dois serviços com foco nos aspectos sociais do compartilhamento de aplicativos entre usuários. Eles fizeram um protótipo de aplicativo de compartilhamento de fotos chamado *Portrait Catalog* e um aplicativo de bate papo chamado “Hanashi”. Eles diferem não só pela funcionalidade e *design*, mas também pela sua disponibilidade para o público, bem como os meios de distribuição que eles oferecem. Enquanto os usuários do “Hanashi” podem procurar contatos de suas próprias listas telefônicas e enviar convites via SMS, os usuários do *Portrait Catalog* podem convidar amigos e familiares pela prática clássica do boca-a-boca, compartilhando uma sugestão ou um *link* entre duas pessoas fisicamente próximas ou encaminhando um *link* para que a outra pessoa baixe o aplicativo via SMS ou e-mail. O trabalho apresenta as informações iniciais sobre como os usuários compartilham serviços móveis entre si em um dispositivo (telefone) que não inclui a distribuição de aplicativos como parte da experiência do usuário. Observou-se que fatores como os hábitos dos usuários de baixar e testar novas aplicações, seus entendimentos sobre o serviço que estão usando e os meios de distribuição que os serviços oferecem, afetam a forma como os serviços foram compartilhados. O trabalho conclui que o compartilhamento de um aplicativo com outros usuários não se dá apenas pela prática do boca-a-boca, mas o aplicativo também precisa

ser compatível com vários aparelhos diferentes para serem compartilhados em uma escala maior. Uma boa distribuição de aplicativos deve ser compatível não apenas com redes sociais estendidas, mas também com compartilhamento casual, por exemplo, no metrô para compartilhar uma foto de um irmão. Quando um serviço é executado apenas em tipos específicos de aparelhos, a quantidade de grupos de usuários diminui. Isso cria um canal de comunicação fechado entre amigos que têm a possibilidade de executar o aplicativo.

3.6 Barreiras, dificuldades e desafios das “App Stores”

As lojas de aplicativos atuais são repositórios de software que fornecem aplicativos depois de solicitações de usuários. No entanto, quando um aplicativo possui um *bug*, a loja continua fornecendo aplicativos defeituosos até que o desenvolvedor carregue uma versão corrigida, impactando assim a reputação do desenvolvedor de lojas e aplicativos. Gomez *et al.* (2015) propõem uma nova geração de lojas de aplicativos cujo objetivo é: (a) reduzir a intervenção humana para manter aplicativos nos dispositivos móveis; e (b) aprimorar os serviços de loja com funcionalidades inteligentes e autônomas para aumentar automaticamente a qualidade dos aplicativos entregues. Uma loja de aplicativos inteligente é capaz de automaticamente detectar aplicativos defeituosos e fornecer diferentes *patches* para evitar *bugs*. Depois de implantar os *patches*, a loja inteligente aprende quais deles são efetivos e melhoram sua estratégia de reparo. Gomez *et al.* (2015) apresentaram uma implementação protótipo com essas características e seu uso em um aplicativo defeituoso real da “Google Play Store”.

A ausência da “Google Play” na China possibilitou a distribuição de aplicativos para Android através de lojas de aplicativos de terceiros. Wang *et al.* (2016) apresentam um método para analisar a confiabilidade das lojas mais populares de aplicativos Android na China. Os resultados indicam que o nível de confiança delas melhorou em média 24%. Sugere-se que as mudanças positivas estejam relacionadas ao desenvolvimento do mercado móvel da China, à melhoria do sistema Android e às políticas introduzidas. Embora o nível de confiança das lojas de aplicativos Android na China ainda seja baixo, prevê-se que ele melhore no futuro.

Vishwanath, Harrison e Ng (2016) analisaram mais de 200 *sites* de lojas de aplicativos de consumo e de negócios. Eles foram avaliados com o objetivo de verificar em

que medida o *design* dos *sites* afeta a aceitação e a utilização pelos usuários finais. Os resultados indicam que os *sites* de software de negócios tendem a fornecer comparativamente mais informações sobre como o produto pode melhorar os negócios e a imagem da empresa e dedicam uma porcentagem proporcionalmente maior do espaço de seu *site* para proporcionar facilidade de uso e informações de influência social, enquanto os *sites* de software de consumo se concentram mais na compatibilidade do software com os diferentes dispositivos e fornecem mais informações de utilização para o usuário final.

Bastani, Anand e Aiken (2015) propõem um processo para a produção de aplicativos certificados para serem livres de ataques maliciosos. Nessa abordagem, o desenvolvedor fornece testes que especificam o código que pode ser acessado durante o uso do aplicativo. O aplicativo hospedado na loja é tratado para impor a especificação fornecida de forma que a execução do código não testado termina o aplicativo. Para tanto, usa-se a inferência abdutiva que, essencialmente, permite calcular as especificações mínimas simultaneamente consistentes com as funções obrigatórias do aplicativo e em concordância com a política das lojas de aplicativos. Para minimizar o tratamento necessário, interage-se com o desenvolvedor para impedir a execução do código inacessível. Foi demonstrada a eficácia da abordagem na verificação de um corpus de 77 aplicativos Android visto que o processo de verificação interativa efetivamente controlou 11 dos 12 ataques que aconteceram.

3.7 *Evolução do software baseada em seu uso e em comentários de usuários na “App Store”*

Ruiz *et al.* (2017) extraíram as classificações de lojas de mais de 10.000 aplicativos na “Google Play”, por um ano inteiro, para examinar se a classificação da loja de um aplicativo é capaz de capturar a mudança dos níveis de satisfação dos usuários em relação às novas versões dos aplicativos. Identificou-se que muitos aplicativos aumentam sua classificação de versão para versão, enquanto alguns outros têm classificações resilientes às flutuações, uma vez que esses aplicativos reuniram um número substancial de avaliadores. Portanto, concluiu-se que a atual classificação de lojas de aplicativos não é dinâmica o suficiente para capturar a mudança dos níveis de satisfação dos usuários associados à evolução da natureza dos aplicativos. Esta resiliência é um grande problema que pode desencorajar os desenvolvedores a melhorar a qualidade de seu aplicativo.

Guzman, Bhuvanagiri e Bruegge (2014) apresentam uma visualização interativa do *feedback* dos usuários. Ela exibe avaliações de aplicativos segundo quatro pontos de vista diferentes: geral, baseado em análise, baseado em recursos e baseado em tópicos. Realizou-se um estudo que visualizou 2009 avaliações do aplicativo Dropbox disponível na “App Store”. Os participantes consideraram a abordagem útil para as tarefas de evolução do software, pois descobriram que poderia auxiliar os desenvolvedores e os analistas a obter uma visão geral dos recursos mais e menos populares dos aplicativos e priorizar seu trabalho. Ao usar diferentes estratégias para encontrar informações relevantes durante o estudo, a maioria dos participantes chegou às mesmas conclusões sobre as críticas dos usuários e as tarefas atribuídas.

3.8 *Diversidade em software para dispositivo móvel*

A ideia de diversidade automática de software tem pelo menos duas décadas de idade (LARSEN *et al.*, 2014). As deficiências das defesas atuais e a transição para a distribuição de software *on-line* (o modelo “App Store”) reviveu o interesse pela diversidade automática de software. A diversidade oferece várias propriedades únicas. Ao contrário de outras defesas, ela introduz incerteza no alvo. O conhecimento preciso do software alvo permite uma ampla gama de ataques. Isso torna a diversidade um mecanismo de defesa difuso e não focalizado. Em segundo lugar, a diversidade oferece proteção probabilística semelhante à criptografia, pois os ataques terão sucesso por acaso. As implementações devem oferecer alta entropia de forma que o espaço de *design* das transformações variáveis do programa seja grande. Como resultado, os pesquisadores propuseram múltiplas abordagens de diversidade de software que variam em relação aos modelos de ameaça, segurança, desempenho e praticidade.

Bures, Hnetyinka e Plasil (2014) propõem um método de modelagem que transfere os benefícios de arquiteturas explícitas de componentes hierárquicos para o projeto de sistemas ciberfísicos inteligentes. Especificamente, o método foi baseado em sistemas de modelagem como arquiteturas de referência de LPS. Contrariamente à LPS tradicional, uma abordagem em tempo de *design*, foram criadas configurações de LPS em tempo de execução. Isso foi feito de forma descentralizada, traduzindo o processo de configuração para o processo de estabelecimento de conjuntos de componentes.

Larsen *et al.* (2014) estudaram sistematicamente o estado da arte na diversidade de software e destacaram os *trade-offs* fundamentais entre diferentes abordagens automatizadas. Também se indicam áreas abertas e desafios não resolvidos. Estes incluem “soluções híbridas”, relatórios de erros, *patches* e ataques na implementação em softwares diversificados.

3.9 Considerações finais

A pesquisa proposta abrange temas associados a soluções arquitetônicas e gerenciamento de aplicativos de curta duração para dispositivos móveis. Por isso, a revisão bibliográfica requereu a criação de um mapa mental para organizar os temas relacionados à pesquisa. A partir deste mapa, foram selecionados trabalhos relacionados aos diferentes temas.

A revisão bibliográfica permitiu identificar que há poucos estudos sobre modelos de ciclo de vida de aplicações e arquitetura de aplicações de ciclo de vida curto. Foram identificados alguns estudos que trabalham com LPS e arcabouços para softwares para dispositivos móveis, desde trabalhos antigos como o de Jaaksi (2002) e o de Abrahamsson *et al.* (2004), até recentes como os trabalhos de Mizouni *et al.* (2014) e Pascual *et al.* (2015). Ainda assim, nenhum desses trabalhos aborda especificamente um modelo de ciclo de vida para software de curta duração para dispositivos móveis.

Da mesma forma, há muitos trabalhos que avaliam e propõem melhorias nas “App Stores” atuais, como em Wang *et al.* (2016), Vishwanath, Harrison e Ng (2016), Gomez *et al.* (2015) e Bastani, Anand e Aiken (2015), e formas de melhorar o software baseadas nos comentários de usuários na “App Store”, como em Ruiz *et al.* (2017) e Guzman, Bhuvanagiri e Bruegge (2014). Também há outros trabalhos que propõem “App Stores” diferentes para domínios específicos, como Kudla *et al.* (2015), Francese, Risi e Tortora (2015), Wenzel (2014) e Beimborn e Palitza (2013). E ainda há algumas propostas para a distribuição de software sem o uso de “App Stores”, como em Pahl e Carle (2013) e Ahmet e Holmquist (2010), porém nenhum desses trabalhos focam em soluções e gerenciamento de aplicativos de curta duração para dispositivos móveis. No próximo capítulo, utilizando os conceitos apresentados no capítulo 2 e o conhecimento adquirido com os trabalhos relacionados apresentados neste capítulo, serão apresentados uma arquitetura e um modelo de ciclo de vida para aplicativos de curta duração.

4 Modelo e arquitetura de ciclo de vida para aplicativos de curta duração

Nesse capítulo são apresentados o modelo e a arquitetura de ciclo de vida para aplicativos de curta duração para dispositivos móveis. Na seção 4.1 é apresentado o modelo de ciclo de vida para esse tipo de aplicativo. Em seguida, na seção 4.2, é apresentada a arquitetura proposta para aplicações de curta duração. A seção 4.3 apresenta o detalhamento técnico da solução proposta neste trabalho. A seção 4.4 apresenta o modelo do leitor e tradutor de especificações da proposta desenvolvida. A seção 4.5 apresenta a aplicação da arquitetura e modelo de ciclo de vida desenvolvidos a um aplicativo exemplo de um *shopping center*. Por fim, a seção 4.6 apresenta as considerações finais sobre os temas discutidos no capítulo.

4.1 Modelo de ciclo de vida

Nesta seção é apresentado o modelo de ciclo de vida para os aplicativos de curta duração e a arquitetura proposta para esse modelo de aplicativos.

A figura 2 apresenta o modelo de ciclo de vida de um aplicativo utilizável no padrão aqui proposto. Esse ciclo é composto de algumas etapas, sendo elas a *especificação*, a *disponibilização* (aplicativo disponível), o *arquivamento* e a *exclusão* do aplicativo.

Figura 2 – Ciclo de vida de um aplicativo

Fonte: André Ramos Moreira, 2019

A primeira etapa na figura 2 consiste na geração do aplicativo a partir da especificação que descreve suas funcionalidades e o tipo de *layout* necessário. No quadro seguinte é representado o aplicativo gerado a partir da especificação. O aplicativo é utilizado e reutilizado tantas vezes quantas necessárias, de acordo com seu ciclo de vida.

Em seguida, na figura 2, há duas possibilidades: arquivamento ou exclusão do aplicativo. Quando um aplicativo é arquivado, ele pode ser recuperado para usos posteriores ou pode ser excluído no futuro. O modelo proposto contempla todas as fases desde a concepção do aplicativo, o que compreende sua geração, uso, arquivamento e recuperação, concluindo com a exclusão, ou seja, do início ao fim da vida útil do aplicativo. Sendo assim, para facilitar o entendimento e tratamento, dividiu-se o problema em subproblemas, sendo eles: (i) Concepção do aplicativo, (ii) Transmissão da especificação do aplicativo, (iii) Geração do aplicativo e (iv) Ciclo de vida do aplicativo no dispositivo.

No primeiro item, concepção do aplicativo, está compreendida a descrição de um aplicativo de forma prévia, contendo as funcionalidades e as informações que o aplicativo deverá possuir. Após essa concepção, deverá ser gerado uma especificação desse aplicativo para que a transmissão dos dados seja efetuada. Para esse passo foi definida uma linguagem de especificação de aplicações de curta duração, que consiste em um conjunto de códigos com palavras abreviadas com o objetivo de facilitar a transmissão, reduzindo a quantidade de dados trafegados. Um módulo externo é responsável pela tradução da especificação do aplicativo para essa linguagem de transmissão econômica e padronizada. Esses passos são executados externamente ao *smartphone*.

A especificação do aplicativo escrita na linguagem utilizada para transmissão deverá estar disponível para que o usuário possa acessá-la através do seu dispositivo. Quando o dispositivo do usuário acessa a especificação, o aplicativo é gerado com base na informação recebida. O controle do ciclo de vida do aplicativo gerado será de responsabilidade do gerador de aplicativos, programa previamente instalado no dispositivo do usuário. Assim, o aplicativo será arquivado ou excluído do dispositivo a partir de regras pré-determinadas. Essas regras podem ser estabelecidas com base em eventos como tempo transcorrido ou localização do dispositivo.

4.2 Arquitetura para aplicações de curta duração

A figura 3 apresenta a arquitetura de alto nível da solução proposta para implementar as aplicações de curta duração e o seu modelo de ciclo de vida, do ponto de vista do usuário final do aplicativo.

Essa solução é composta de um módulo externo de especificação, chamado *Especificador*, e três módulos internos, a saber, *Leitor e Tradutor de Especificações*, *Gerenciador* e o *Aplicativo* especificado (gerado). O Especificador é responsável pela modelagem de aplicativos e pela tradução das especificações na linguagem proposta neste trabalho. O módulo Leitor e Tradutor faz a leitura da especificação e gera o aplicativo; o Gerenciador configura o aplicativo de acordo com a informação recebida e gerencia o ciclo de vida do aplicativo. Dessa maneira, o módulo Gerenciador é responsável pela disponibilização, arquivamento e exclusão do aplicativo. O último módulo interno é o próprio aplicativo.

Figura 3 – Arquitetura de alto nível

Fonte: André Ramos Moreira, 2019

A solução proposta gera um aplicativo de ciclo de vida curto, a partir de uma especificação definida previamente. Após ser definida, a especificação deve ser convertida para um formato compreensível pelo Gerenciador, para que seja lida pelo dispositivo móvel do usuário.

O módulo Especificador, externo ao dispositivo, está fora do escopo desse trabalho que se concentra nos módulos internos do dispositivo móvel.

O módulo Leitor e Tradutor, interno ao aplicativo do dispositivo móvel, é responsável pela leitura e interpretação da especificação gerada. Uma vez que essa especificação for validada, o aplicativo transcreve toda a informação necessária para a montagem do aplica-

tivo especificado em um arquivo de configuração. Esse arquivo é gerado e armazenado no dispositivo móvel em um diretório criado especificamente para armazenar essas informações.

Para a construção do aplicativo, é realizada a seleção de um *layout* a partir de um conjunto pré-existente no aplicativo gerenciador, em seguida esse *layout* é configurado com base nas informações fornecidas pela especificação do aplicativo. Com essas informações, é executada a configuração desse *layout*, conforme suas características.

Os *layouts* são construídos para executar os comandos de ações obtendo as informações em tempo de execução por meio do pacote Java *Reflection*¹, o que exige que as informações das chamadas dessas ações estejam contidas na especificação e consequentemente no arquivo de configuração do aplicativo.

Após a instalação, na primeira abertura do aplicativo gerado, é criado um arquivo de dados para armazenar as informações do aplicativo gerado. Esses dados são serializados e armazenados nesse arquivo. O aplicativo pode ser utilizado pelo usuário conforme sua necessidade e, após a utilização, ele pode ser arquivado, recuperado ou excluído conforme seu ciclo de vida.

Cada aplicativo gerado pode ser arquivado, quando sua especificação é movida para uma pasta de arquivados; também pode ter os dados arquivados, quando o arquivo de dados é movido para a pasta de arquivados; ou excluídos completamente, quando os arquivos de especificação e de dados são removidos do dispositivo.

4.3 Aspectos do desenvolvimento

Nesta seção, serão abordados os aspectos técnicos do desenvolvimento do modelo de ciclo de vida e da arquitetura propostos. São apresentadas as diferentes tecnologias utilizadas, assim como os desafios tecnológicos encontrados.

4.3.1 Especificação do aplicativo

O primeiro passo do ciclo de vida é a especificação do aplicativo. Nela, as funcionalidades são listadas e em seguida traduzidas para a linguagem de especificação para que

¹ Java Reflection - Disponível em <https://www.oracle.com/technetwork/articles/java/javareflection-1536171.html> acessado em 11.01.2019

seja transmitida de forma compreensível ao gerenciador de ciclo de vida no dispositivo móvel.

Para uma eficiente organização e interpretação da linguagem de especificação foi escolhido o formato JSON², um acrônimo de *JavaScript Object Notation*, um formato compacto, de padrão aberto independente, para troca de dados simples e rápida entre sistemas. Esse formato foi escolhido por ter como característica principal a representação organizada das informações com o uso de poucos caracteres. Assim, a compreensão das funcionalidades do aplicativo gerado é feita de forma simples e sem o consumo de muito recursos.

Na tabela 1, são apresentados os principais atributos e estruturas utilizados para descrever aplicativos utilizando JSON. A coluna *Tag* contém o rótulo (*tag*) identificador do atributo ou item de estrutura ao qual esse item se refere. A coluna *Contido* indica se este rótulo está contido em algum outro nível de rótulo. A coluna *Nome* apresenta uma breve descrição de cada rótulo. A coluna *Tipo* apresenta que tipo de item esse rótulo se refere, normalmente um atributo ou uma estrutura. Estruturas são itens de alto nível que contêm um ou mais atributos. Atributos são itens únicos de último nível, sem itens inseridos. A coluna *Obrig* apresenta a obrigatoriedade do rótulo. E, por fim, a coluna *Multiplic* apresenta a multiplicidade do rótulo no JSON para este projeto. Nessa coluna o símbolo + significa ‘ou mais’, ou seja, é utilizado quando a multiplicidade o item é zero ou mais, ou um ou mais.

Na figura 4, é apresentado um trecho de uma especificação utilizando JSON para o exemplo de utilização do aplicativo para um *shopping center*. Essa especificação utiliza diversos campos da tabela 1. Ela é iniciada com o campo *nm* que traz a descrição do nome do aplicativo (Barra Azul Shopping) que faz referência ao nome do estabelecimento ao qual o aplicativo se aplica. Em seguida, o campo *tm* descreve o tempo para arquivamento do aplicativo (250 minutos). O próximo atributo, *lc*, descreve a área para funcionamento do aplicativo, composto pelo raio, *ra*, de 300 metros, e posição inicial, *pt*, contendo a longitude e latitude inicial. O próximo componente do JSON, *l*, traz o *layout* do aplicativo, composto pelo identificador do *layout*, *id*, e nome do *layout*, *n*. Ainda dentro da estrutura de *layout*, em seguida, *lst*, apresenta os campos de pesquisa, nesse caso, as lojas, com seus identificadores, *id*, nomes, *nm*, e valores, *vl*. Em seguida, é feita a chamada de uma imagem, também na estrutura do *layout*, *im*, com seu identificador, *id*, e sua fonte, *URL*.

² <https://www.ecma-international.org/publications/standards/Ecma-404.htm>

Tabela 1 – Atributos e dados para descrição de um aplicativo

Tag	Contido	Nome	Tipo	Obrig	Multipli
app	-	Tag inicial	TAG	Sim	1
nm	app	Nome do programa	Atributo	Sim	1
tm	app	Tempo para arquivamento (min)	Atributo	Não	1
lc	app	Área para funcionamento (mt)	Estrutura	Não	1
ra	lc	Raio para funcionamento (mt)	Atributo	Sim	1
pt	lc	Latitude, longitude inicial	Atributo	Sim	1
l	app	Tag inicial <i>layout</i>	Estrutura	Sim	1+
id	l	Identificador do <i>layout</i>	Atributo	Sim	1
n	l	Nome do layout	Atributo	Não	1
a	l	Área de texto	Estrutura	Não	0+
id	a	Identificador da área de texto	Atributo	Sim	1
v	a	Texto da área	Atributo	Não	1
p	l	Pesquisa	Estrutura	Não	0+
id	p	Identificador da pesquisa	Atributo	Sim	1
v	p	Texto da pesquisa	Atributo	Sim	1
im	l	Imagem	Estrutura	Não	0+
id	im	Identificador da imagem	Atributo	Sim	1
URL	im	Fonte da imagem	Atributo	Sim	1
b	l	Botão	Estrutura	Não	0+
id	b	Identificador do botão	Atributo	Sim	1
nm	b	Nome de exibição	Atributo	Sim	1
s	b	Estilo do botão	Atributo	Não	1
ac	l/b	Ação do botão	Estrutura	Sim	1
id	ac	Identificador da ação	Atributo	Sim	1
nm	ac	Nome da ação	Atributo	Sim	1
pk	ac	Pacote	Atributo	Sim	0+
cs	ac	Classe	Atributo	Sim	0+
mt	ac	Método	Atributo	Sim	0+
pe	ac	Parâmetro de entrada	Estrutura	Não	0+
id	pe	Identificador do parâmetro de entrada	Atributo	Sim	1
nm	pe	Nome do parâmetro	Atributo	Não	1
tp	pe	Tipo do parâmetro	Atributo	Sim	1
ps	ac	Parâmetro de saída	Estrutura	Não	0+
id	ps	Identificador do parâmetro de saída	Atributo	Sim	1
nm	ps	Nome do parâmetro	Atributo	Não	1
tp	ps	Tipo do parâmetro	Atributo	Sim	1
lst	l	Lista de itens	Estrutura	Não	0+
id	lst	Identificador	Atributo	Sim	1
nm	lst	Nome do atributo	Atributo	Sim	1
vl	lst	Valor do atributo	Atributo	Sim	1

Fonte: André Ramos Moreira, 2019

No caso do exemplo do *shopping center*, a imagem contém um mapa com a localização das lojas. É importante ressaltar que as imagens são carregadas de sua fonte para

Figura 4 – JSON *Barra Azul Shopping*

```

1  {
2    "app": [
3      {
4        "nm": "Barra Azul Shopping",
5        "tm": "250",
6        "lc": [
7          {
8            "ra": "300",
9            "pt": "-22.541189,-45.301691"
10         }
11       ],
12       "l": [
13         {
14           "id": "2",
15           "n": "shop",
16           "ac": [
17             {
18               "id": "1",
19               "nm": "Cadastro Promoção",
20               "pk": "com.mestrado.api",
21               "cs": "Cadastro",
22               "mt": "efetuarCadastroNvl",
23               "pe": [
24                 {
25                   "id": "1",
26                   "nm": "serverResposta",
27                   "tp": "String"
28                 }
29               ],
30               "ps": [
31                 {
32                   "id": "1",
33                   "nm": "sucesso",
34                   "tp": "boolean"
35                 }
36               ]
37             }
38           ],
39           "im": [
40             {
41               "id": "1",
42               "url": "https://bit.ly/2AiuczH"
43             }
44           ],
45           "a": [
46             {
47               "id": "1",
48               "v": "Oferta: desconto de 30% na Loja Baiana"
49             },
50             {
51               "id": "2",
52               "v": "Especial: Natal Barra Azul Shopping - Cadastre-se na promoção!"
53             }
54           ],
55           "lst": [
56             {
57               "id": "1",
58               "nm": "nome",
59               "vl": "Saveur Restaurant"
60             },
61             {
62               "id": "1",
63               "nm": "local",

```

Fonte: André Ramos Moreira, 2019

que a descrição do aplicativo se mantenha reduzida, em conformidade com os objetivos desse projeto. Por fim, ainda na estrutura do *layout*, diferentes áreas de textos, *a*, são especificadas, com seus identificadores, *id*, e textos, *v*. No apêndice A, é possível visualizar as diferentes especificações em JSON utilizadas neste trabalho.

4.4 Leitor e tradutor de especificações

No dispositivo móvel há um aplicativo responsável pela geração dos aplicativos especificados por meio da leitura e interpretação das especificações. Esse mesmo aplicativo também é responsável pela execução e pelo gerenciamento do ciclo de vida dos aplicativos.

O aplicativo gerenciador contém o módulo responsável pela leitura e interpretação da especificação criada anteriormente. Existem diferentes formas para transmissão de especificações para a aplicação, porém, elas estão fora do escopo desse trabalho.

Após a transmissão das especificações, é gerado um arquivo de configuração do aplicativo especificado. Nesse arquivo estão registradas todas as informações necessárias para a execução das funcionalidades especificadas, além de informações necessárias para o gerenciamento do ciclo de vida do aplicativo.

A especificação deve ser transmitida através de um código em JSON. Uma vez que o código da especificação transmitido é lido, ele é interpretado para preenchimento do arquivo de configuração do aplicativo a ser gerado. É feita a validação da especificação do JSON, inicialmente com a análise do código na qual são verificadas a sintaxe e a semântica da especificação. Por exemplo, em relação à sintaxe, observa-se se a sintaxe está correta e, em relação à semântica, se a informação de ciclo de vida do aplicativo está documentada.

Se o código JSON recebido for validado, é criado um arquivo de configuração para cada aplicativo. Esse arquivo é salvo com um nome específico e em um diretório já existente do dispositivo móvel, onde estarão todos os arquivos de configuração dos diferentes aplicativos a serem gerados. Essa criação e gerenciamento das pastas é feita pelo aplicativo gerador. Algumas informações adicionais podem ser incluídas no arquivo de configuração do aplicativo, conforme as diferentes necessidades dos aplicativos. Por exemplo, a data de geração do aplicativo pode ser incluída para ser utilizada como parâmetro para o gerenciamento do ciclo de vida.

Para a geração do aplicativo, é necessário descrever, na especificação, além das funcionalidades do aplicativo, qual *layout* será utilizado; essa informação também estará contida no arquivo de configuração. Há um grupo de *layouts* pré-definidos, por exemplo, cardápio, evento, mapa, entre outros, de forma que, na geração do aplicativo, é feita a seleção do tipo de *layout* a ser utilizado.

As chamadas das ações do *layout* também são realizadas com base na especificação do aplicativo. Essas chamadas são realizadas via Java *Reflection*, uma funcionalidade para determinar métodos e atributos desconhecidos de determinada classe ou método em tempo de execução. As informações das configurações do Java *Reflection* utilizadas nas ações são passadas na especificação do aplicativos por meios das *tags*. Podemos verificar como essas informações são disponibilizadas através da especificação apresentada no exemplo da figura 4 nas linhas de 16 a 38.

4.4.1 Gerenciador de aplicativos

O gerenciador de aplicativos é responsável pelo controle dos aplicativos. Ele pode conter zero ou mais aplicativos gerados ativos à disposição do usuário do dispositivo. Ao acessar o gerenciador de aplicativos, uma listagem dos aplicativos ativos é produzida por meio da leitura dos arquivos disponíveis no diretório. Na figura 5, é possível visualizar um exemplo de lista de aplicativos gerados pelo gerenciador e disponíveis para utilização. No exemplo, são listados os aplicativos descritos na seção 1.1; para acessá-los, basta clicar no botão do aplicativo de interesse.

Figura 5 – Lista de aplicativos

Fonte: André Ramos Moreira, 2019

4.4.2 Gerenciamento de ciclo de vida

Há dois modelos de ciclo de vida em que o gerenciador atua: voluntário e automático. No ciclo de vida voluntário, os aplicativos ativos podem ser excluídos ou arquivados a qualquer momento, de acordo com a demanda do usuário do dispositivo móvel, por meio de funcionalidade disponível no gerenciador.

Porém, o gerenciador pode também controlar o ciclo de vida do aplicativo de maneira automática. Há vários tipos de gatilhos para que o gerenciador execute a exclusão ou arquivamento dos aplicativos; nesse trabalho implementamos os controles por prazo e por localização. Quando o gatilho for por prazo, na especificação será transmitido esse prazo ou período em que o aplicativo deve ficar ativo. Quando for por localização, o raio de disponibilidade em que o aplicativo deve estar ativo também será transmitido na especificação. Essas informações são recebidas na especificação e registradas no arquivo de configuração do aplicativo gerado.

A alteração do estado do ciclo de vida do aplicativo se dá a partir da movimentação do arquivo de configuração. Para a alteração de *status* de **ativo** para **arquivado**, o arquivo de configuração é transferido da pasta de aplicativos ativos para pasta de arquivados. Para a alteração de **ativo** para **excluído**, o arquivo de configuração é excluído. Além disso, é possível arquivar ou remover apenas o arquivo de dados, ou seja, o aplicativo continua ativo, porém, os dados são arquivados (movidos para outra pasta) ou excluídos.

Uma vez arquivados, o aplicativo ou dados podem ser recuperados. A recuperação é feita por meio da movimentação dos arquivos para a pasta de ativos. Caso o aplicativo seja excluído, não é possível realizar sua recuperação; se o aplicativo for novamente necessário, ele deverá ser gerado outra vez.

4.4.3 Construção do aplicativo gerenciador

O aplicativo gerenciador é composto de algumas etapas: (i) leitura e interpretação de especificação, (ii) apresentação de aplicativo gerador e (iii) gerenciador do ciclo de vida. Na construção do aplicativo gerenciador foi utilizada a interface do tipo *Fragment* do Android para estruturação e fluxo de telas.

Para a leitura e interpretação da especificação, foram produzidas uma classe Java, responsável por listar os objetos do diretório de especificação, e uma tela codificada de forma a receber essa lista de arquivos do diretório de programas e verificar as regras de arquivamento e exclusão automáticos. Caso o aplicativo gerado esteja infringindo alguma das regras, o arquivamento ou exclusão é efetuado. Os demais aplicativos são apresentados em forma de lista na tela para que usuário possa escolher qual deseja acessar.

Quando um programa é selecionado ao clicar na tela, o *layout* a ser utilizado é verificado no arquivo de configuração do programa, e então esse *layout* é carregado. Ao abrir o *layout* há uma rotina que lê o arquivo de configuração do programa para obter as informações e preencher as variáveis, por exemplo, nome do aplicativo, nome de menus ou listas de produtos.

As ações que o aplicativo possui também são obtidas do arquivo de configuração. Quando uma ação é invocada em tempo de execução, são acessadas as informações de configuração utilizando a chave respectiva, uma vez que o Java Reflection não exige que nenhuma configuração prévia seja realizada.

Na abertura do aplicativo também é consultado o arquivo de dados. O armazenamento desses dados é realizado através de serialização de objeto, e para obter as informações necessárias é realizada a desserialização do objeto e a verificação se há dados que foram armazenados previamente. O armazenamento dos dados é realizado após quaisquer ações que alteram o conteúdo das estruturas ou dos dados, ou seja, sempre que for identificada alguma alteração nos dados, eles são serializados.

Para o gerenciamento do ciclo de vida, foram criadas telas de navegação para decidir quais programas serão manipulados, servindo tanto programas ativos quanto para programas arquivados. A mesma classe de listagem de arquivos de programas é utilizada para os programas ativos e arquivados, acessando os respectivos diretórios.

Uma vez que um programa ativo é selecionado, é aberta uma nova tela onde são habilitadas as opções para excluir ou arquivar o programa. Quando o usuário seleciona a opção de exclusão, o arquivo de configuração é removido do aparelho. Quando seleciona o arquivamento, o arquivo de configuração é movido para o diretório de arquivados.

Ao selecionar um aplicativo da lista dos aplicativos arquivados, são habilitadas as opções para ativação do aplicativo com ou sem os dados e também para exclusão. Da mesma forma que ao arquivar um aplicativo ativo, seu arquivo de configuração é movido,

nesse caso, a classe faz a movimentação inversa, trazendo o arquivo de configuração e, se for o caso, o arquivo de dados de volta ao diretório de aplicativos ativos.

4.5 *Aplicativo Barra Azul Shopping*

Para validação dos conceitos e ferramentas apresentados nesse projeto, foi desenvolvido o aplicativo do *Barra Azul Shopping*, conforme exemplo descrito na seção 1.1. O aplicativo utiliza *layout* específico para localização de lojas e exibição do mapa do estabelecimento, bem como para divulgação de promoções e busca por lojas específicas.

O primeiro passo desse processo foi elencar quais funcionalidades seriam validadas com esse exemplo. Após avaliações internas, concluiu-se que as funcionalidades nativas do Android permitem a avaliação da proposta.

Seguindo os passos da arquitetura proposta, depois da definição do aplicativo, o próximo passo foi gerar uma especificação com todas as funcionalidades do aplicativo, no formato compreensível pelo interpretador do aplicativo gerenciador. Em seguida, essa especificação foi lida pelo *smartphone* que contém o aplicativo gerenciador instalado.

O Gerenciador responsável pela leitura, tradução e interpretação da informação obteve a especificação e utilizou-a para gerar o aplicativo especificado. O aplicativo passou a estar disponível para o usuário do *smartphone*. Após o período de utilização – que pode variar de acordo com o aplicativo especificado – o Gerenciador pode arquivar o aplicativo ou removê-lo do *smartphone*. Aplicativos excluídos desalocam memória e liberam processamento, porém, para utilizá-los uma outra vez é necessário acessar a especificação e gerá-los novamente. Aplicativos arquivados liberam apenas processamento, porém, para utilizá-los novamente basta fazer a recuperação no próprio Gerenciador.

As próximas figuras apresentam as principais telas do aplicativo gerado neste projeto. Elas visam demonstrar o objetivo do trabalho que é a geração de aplicativos de vida curta de forma automatizada baseada em uma especificação simplificada.

A figura 6a apresenta a tela inicial do aplicativo gerador, quando é instalado e aberto pela primeira vez no dispositivo móvel do usuário final. Em seguida, a figura 6b, apresenta o menu onde é possível acessar o módulo para obtenção de uma especificação e geração de um novo aplicativo (botão “Novo Aplicativo”). O menu apresenta também o módulo de gerenciamento e controle do ciclo de vida (botão “Ciclo de Vida”), as configurações do

aplicativo gerador (botão “Configurações”), além das opções para compartilhar em redes sociais (botão “Compartilhe”) e um botão para que o usuário entre em contato com a equipe do aplicativo gerador (botão “Fale Conosco”).

Figura 6 – Telas iniciais do aplicativos.

Fonte: André Ramos Moreira, 2019

Em seguida, ao clicar no botão “Novo Aplicativo” da figura 6b, o usuário é direcionado para a tela apresentada na figura 7a. Nessa tela, é possível gerar um aplicativo exemplo, utilizando especificações previamente cadastradas no aplicativo gerador, ao clicar no botão “Gerar Exemplo”. Na tela da figura 7a, ele também permite que seja feita a leitura da informação de especificação de um aplicativo, clicando no botão “Leitura de Especificação”. Além disso, o usuário tem a sua disposição o botão “Tela Inicial” para voltar à tela inicial do aplicativo conforme apresentada na figura 6a.

Ao clicar no botão “Leitura de Especificação”, da figura 7a, o dispositivo faz a leitura da especificação. Após a recepção da especificação do aplicativo *Barra Azul Shopping*, essa informação é passada para o módulo Leitura e Tradução de Especificações para a análise. Caso a leitura e validação da especificação sejam efetuadas com sucesso, o usuário final recebe uma mensagem de criação do aplicativo, conforme tela apresentada na figura 7b.

Na figura 8, é apresentada novamente a tela inicial do aplicativo gerador, que, diferente da figura 6a, apresenta agora o aplicativo gerado, com o botão “BARRA AZUL SHOPPING” disponível para que o usuário acesse.

Figura 7 – Telas de geração do aplicativo.

Fonte: André Ramos Moreira, 2019

Após clicar no botão “BARRA AZUL SHOPPING” da tela apresentada na figura 8, é aberto o aplicativo descrito pela especificação apresentada na figura 4. A tela inicial desse aplicativo gerado em tempo de execução é apresentada na figura 9a. A pesquisa de lojas do *shopping* é apresentada no menu superior para que o usuário localize um estabelecimento específico que deseja visitar. É apresentado ainda o mapa do *shopping*. O aplicativo também possui a informação de promoções gerais do *shopping* e promoções específicas de lojas, por exemplo, desconto de 30% na Loja Baiana. Na tela apresentada na figura 9a, é possível visualizar ainda um botão de ação “Cadastro Promoção” em que o usuário pode cadastrar-se na promoção de Natal do *Barra Azul Shopping* para usuários cadastrados e utilizadores do aplicativo do *shopping*.

Em seguida, na tela da figura 9b, é apresentada a utilização da ferramenta de busca do aplicativo. Nesse exemplo, foi feita a busca pela “loja baiana”; como resultado, são apresentadas as principais informações da loja, nesse caso, sua localização e tipo. No entanto, outras informações podem ser ainda inseridas como, por exemplo, *website*, telefone, etc., de acordo com a necessidade de cada estabelecimento. Ao utilizar a busca,

Figura 8 – Tela inicial com aplicativo gerado

Fonte: André Ramos Moreira, 2019

as funcionalidades da tela inicial do aplicativo apresentadas na figura 9a também são mantidas disponíveis para o usuário final.

Ao clicar no botão “Cadastro Promoção”, disponível nas telas do aplicativo do *Barra Azul Shopping*, conforme figuras 9a e 9b, é aberta a tela de cadastro apresentada na figura 9c. Nessa tela, o usuário do aplicativo gerado cadastra seu nome, e-mail e telefone para receber promoções específicas do *Barra Azul Shopping*, se assim o desejar. Nessa tela estão disponíveis também o botão “Voltar”, caso o usuário deseje voltar para a tela anterior sem efetivar o cadastro, e o botão “Enviar”, para o usuário enviar as informações de seu cadastro para participar da promoção.

Ao clicar no botão “Ciclo de Vida” da tela apresentada na figura 6b, o usuário é direcionado para a tela apresentada na figura 10a. Nessa tela são disponibilizados os botões “Aplicativos Ativos”, no qual o usuário pode consultar a lista de aplicativos ativos, e “Aplicativos Arquivados”, em que ele pode consultar a lista de aplicativos arquivados. Para cada aplicativo, estão disponíveis as funcionalidades para arquivar ou excluir, no caso de aplicativo ativo, e para ativar ou excluir, no caso de um aplicativo arquivado. Além disso, o botão “Tela Inicial” para voltar à tela inicial do aplicativo, está disponível na tela apresentada na figura 10a.

Figura 9 – Telas do aplicativo *shopping*.

Fonte: André Ramos Moreira, 2019

Ao clicar no botão “Aplicativos Ativos” da tela apresentada na figura 10a, o usuário recebe a lista de aplicativos ativos e pode gerenciar o ciclo de vida de cada um deles. Como o aplicativo está ativo, é possível excluí-lo ou arquivá-lo. Na tela 10b são apresentadas as opções para gerenciamento do ciclo de vida de um aplicativo ativo, conforme a necessidade do usuário. O botão “Excluir Programa” permite que um aplicativo ativo seja permanentemente excluído do dispositivo móvel, sem a possibilidade de recuperação para uso futuro, ou seja, um aplicativo excluído teria que passar pelo processo de geração novamente para ser utilizado. O botão “Arquivar Programa” permite que um aplicativo seja arquivado, ou seja, é liberado espaço por meio da desalocação de memória e pelo uso de dados. Um aplicativo arquivado pode ser reativado para utilização pelo usuário final ou pode ser excluído permanentemente. É importante destacar que o ciclo de vida também pode ser gerenciado de forma automatizada, utilizando regras pré-estabelecidas. Por exemplo, um aplicativo relacionado a palpites para a Copa do Mundo pode ser automaticamente excluído ou arquivado após o término desse torneio. Na tela apresentada na figura 10b, o botão “Tela Inicial” também está disponível para utilização pelo usuário.

Ao clicar no botão “Aplicativos Arquivados” da tela apresentada na figura 10a, o usuário recebe a lista de aplicativos arquivados e pode gerenciar o ciclo de vida de cada um deles. Como o aplicativo está arquivado, é possível ativá-lo novamente, com ou sem os dados

anteriormente salvos, ou excluí-lo. Na figura apresentada na tela 10c, são apresentadas as opções para gerenciamento do ciclo de vida de um aplicativo arquivado, conforme a necessidade do usuário. O botão “Ativar Aplicativo” faz com que o aplicativo seja ativado para utilização pelo usuário, porém, sem os dados anteriormente salvos. Retomando o exemplo de um aplicativo relacionado a palpites para a Copa do Mundo, nesse caso o aplicativo seria carregado sem nenhum palpite anteriormente preenchido. O botão “Ativar Aplicativo e Dados” carrega o aplicativo gerado com todos os dados anteriormente salvos. Para aplicativo de palpites para a Copa do Mundo, nesse caso o aplicativo seria carregado com os palpites anteriormente preenchidos salvos. O botão “Excluir Aplicativo do Arquivo” exclui permanente o aplicativo gerado assim como qualquer dado registrado anteriormente. Na tela apresentada na figura 10c, também está disponível o botão “Tela Inicial”.

(a) Tela ciclo de vida.

(b) Tela opções do aplicativo ativo.

(c) Tela opções do aplicativo arquivado.

Figura 10 – Telas do ciclo de vida do aplicativo

Fonte: André Ramos Moreira, 2019

Caso de uso do aplicativo do *Barra Azul Shopping*

A utilização do aplicativo exemplo dar-se-ia da seguinte forma. O usuário João da Silva chega ao *Barra Azul Shopping* e, com seu dispositivo móvel, lê especificação do aplicativo. Após a leitura, o aplicativo gerador faz a interpretação da especificação e gera o aplicativo do *Barra Azul Shopping*. João da Silva faz o uso do aplicativo durante sua

estadia no estabelecimento, consultando diversos itens como localização de lojas específicas, promoções de produtos de seu interesse, eventos e outras informações relevantes. Após sair do estabelecimento, o aplicativo é arquivado pelo gerenciador, usando o parâmetro de distância para arquivamento automático. Após um período estabelecido, se não houver outros usos, o aplicativo é excluído automaticamente pelo gerenciador.

4.6 Considerações finais

Neste capítulo, foram apresentados o modelo de ciclo de vida e a arquitetura para gerenciamento de aplicativos de curta duração em dispositivos móveis. Foram abordados aspectos funcionais e técnicos do modelo e da arquitetura por meio do aplicativo do *shopping* Barra Azul. No próximo capítulo, será apresentada a validação da abordagem com a sua utilização em outros exemplos e a discussão de suas vantagens e limitações.

5 Estudo de caso da aplicação do modelo

O objetivo desse capítulo é apresentar um estudo de caso da aplicação do modelo e a arquitetura propostos neste trabalho, ou seja, ele visa fornecer evidências da efetividade da abordagem para a criação de aplicações de curta duração. Para tanto, a arquitetura de aplicativos de curta duração e o gerenciador de ciclo de vida foram aplicados a três exemplos típicos.

A seguir, são descritos os exemplos implementados, os desafios e as dificuldades encontradas, bem como as decisões de implementação, as funcionalidades e recursos nativos explorados.

5.1 Implementação de três exemplos

Foram definidos três exemplos com as características necessárias para a aplicação da abordagem proposta: *layout* e funcionalidades simples; baixa complexidade; e ciclo de vida limitado, seja por tempo, seja por localização. Os três exemplos, além dessas características, foram escolhidos por terem aplicabilidade real no dia-a-dia de usuários de dispositivos móveis. O primeiro exemplo, apresentado na seção 5.1.1, é uma aplicação para um estabelecimento comercial tipo *shopping center*, nomeado ficticiamente como “Barra Azul Shopping”. O segundo exemplo, apresentação na seção 5.1.2, é um aplicativo para um evento, chamado “Rackthon” nesse trabalho. E o terceiro exemplo, apresentado na seção 5.1.3, é um aplicativo para um restaurante fictício com o nome “Saveur Restaurant”.

5.1.1 Exemplo 1: *Barra Azul Shopping*

Nesse exemplo foi desenvolvido um aplicativo para *shopping centers* que contém a descrição das lojas, mapas, promoções, entre outras informações. Esse exemplo permitiu validar o funcionamento do modelo do aplicativo gerador, com um *layout* simples e leve e com funcionalidades que atendem às necessidades dos usuários do *Barra Azul Shopping*. Esse aplicativo foi o primeiro a ser utilizado nesse modelo e foi descrito detalhadamente, assim como sua utilização passo-a-passo, na seção 4.5.

É importante salientar que o *layout* e as funcionalidades desse aplicativo podem ser modificados pelo estabelecimento que vai utilizá-lo. Podem ser adicionados outros recursos, por exemplo, fotos das lojas ou produtos em promoção; também podem ser excluídas ou incluídas funcionalidades, por exemplo, o cadastro em promoções. O objetivo desse exemplo foi desenvolver uma aplicação para a abordagem proposta, com funcionalidades simples. O aplicativo é útil ao usuário final basicamente pelo período em que está no estabelecimento. Depois que ele deixa o *shopping center*, o aplicativo pode ser arquivado e excluído de forma automática, sem prejuízo para o usuário final e com a liberação dos recursos no dispositivo móvel.

5.1.2 Exemplo 2: *Rackthon - 2019*

Nesse exemplo, apresentado brevemente a seguir, e visualizável na figura 11, foi desenvolvido um aplicativo para um evento fictício *Rackthon - 2019*. Esse aplicativo descreve as atividades e datas dos eventos e permite consulta e adição de compromissos à agenda do celular, entre outras funcionalidades.

A tela inicial do aplicativo, contida na figura 11a, apresenta o nome do evento e sua descrição, detalhes do evento e o acesso à agenda em dias (dias 01, 02, 03, 04 e 05, nesse aplicativo exemplo). É importante destacar que essas informações podem ser alteradas de acordo com os diferentes eventos, assim como o *layout* do aplicativo.

Ao clicar em um dia de evento (nesse exemplo, dia 01), o usuário é direcionado para a tela apresentada na figura 11b. Nessa tela, é possível visualizar os dados relativos aos diversos eventos desse dia como local, horário, descrição, palestrantes, entre outras informações que podem variar de acordo com o tipo de evento. Na tela da figura 11b, o usuário também tem acesso à funcionalidade que permite a adição da palestra à agenda do dispositivo móvel. O aplicativo desenvolvido utiliza, além do gerador e do modelo de ciclo de vida, recursos nativos de dispositivos móveis que utilizam a plataforma Android.

Nesse exemplo foi possível explorar uma aplicação diferente para a abordagem proposta. É importante destacar que um aplicativo desse tipo é utilizado basicamente no período de duração de evento, por isso, tem um ciclo de vida curto, sendo arquivado e excluído automaticamente encerrado o evento, desonerando recursos (espaço e processamento) do dispositivo.

(a) Tela inicial do aplicativo *Rackthon - 2019*. (b) Agenda do aplicativo *Rackthon - 2019*.

Figura 11 – Aplicativo *Rackthon - 2019*.

Fonte: André Ramos Moreira, 2019

Na figura 12 é apresentado um trecho do arquivo JSON de especificação utilizado para a geração do aplicativo do evento *Rackthon - 2019*. É possível compará-lo ao arquivo JSON do aplicativo do *shopping* Barra Azul, disponível na figura 4, e verificar que há alguns pontos em comum e algumas diferenças fundamentais entre eles, elencados a seguir.

As *tags* utilizadas nos dois exemplos são, em essência, as mesmas, o que demonstra a facilidade no reúso dessa solução. A diferença fundamental entre eles está no conteúdo das *tags* que varia consideravelmente de uma aplicação para a outra, afinal estão em domínios distintos. Ou seja, para a criar a especificação do aplicativo do *Rackthon - 2019* a partir da especificação do aplicativo do “Shopping Barra Azul”, bastou definir o conteúdo que o aplicativo terá e inseri-lo nas respectivas *tags*. É possível visualizar os arquivos de especificação JSON completos no Apêndice A.

Figura 12 – JSON *Rackthon - 2019*

```

1  {
2    "app": [
3      {
4        "nm": "Rackthon - 2019",
5        "tm": "2500",
6        "lc": [
7          {
8            "ra": "1000",
9            "pt": "-23.541189, -46.301691"
10         }
11       ],
12       "l": [
13         {
14           "id": "4",
15           "n": "evento",
16           "ac": [
17             {
18               "id": "1",
19               "nm": "Adicionar calendário",
20               "pk": "com.mestrado.api",
21               "cs": "Calendario",
22               "mt": "inclirEventoCalendario",
23               "pe": [
24                 {
25                   "id": "1",
26                   "nm": "latitude",
27                   "tp": "double"
28                 },
29                 {
30                   "id": "2",
31                   "nm": "longitude",
32                   "tp": "double"
33                 }
34               ],
35               "ps": [
36                 {
37                   "id": "1",
38                   "nm": "eventID",
39                   "tp": "long"
40                 }
41               ]
42             }
43           ],
44           "a": [
45             {
46               "id": "1",
47               "t": "Conexões diferentes. Idéias de impacto."
48             }
49           ],
50           "im": [
51             {
52               "id": "1",
53               "url": "https://bit.ly/2C2ikS0"
54             }
55           ],
56           "lst": [
57             {
58               "id": "1",
59               "nm": "dia",
60               "vl": "1"
61             },
62             {
63               "id": "1",

```

Fonte: André Ramos Moreira, 2019

5.1.3 Exemplo 3: *Saveur Restaurant*

Nesse exemplo, descrito nessa seção e apresentado na figura 13, foi desenvolvido um aplicativo para menu digital de um restaurante que permite, entre outras funcionalidades, consultar os pratos do cardápio e fazer pedidos.

Na figura 13a, é apresentada a tela inicial do aplicativo para menu digital de um restaurante. Esse aplicativo apresenta o nome do estabelecimento e os diferentes tipos de prato (prato principal, sobremesas, bebidas, etc.) com seus nomes, fotos, descrições e preços; o aplicativo ainda permite realizar pedidos.

A figura 13b apresenta o “carrinho de compras” do aplicativo do restaurante. Nela são descritos os pedidos selecionados com sua respectiva quantidade. O “carrinho de compras” permite adicionar, remover ou excluir itens, além de voltar para o menu da tela da figura 13a para adicionar itens diferentes, utilizando o botão “Adicionar mais itens”, ou finalizar o pedido e efetuar o pagamento.

Figura 13 – Aplicativo *Saveur Restaurant*.

Fonte: André Ramos Moreira, 2019

Na figura 14 é apresentado um trecho do arquivo JSON de especificação utilizado para a geração do aplicativo do *Saveur Restaurant*. É possível compará-lo ao arquivo JSON do aplicativo do *shopping* Barra Azul, disponível na figura 4 e do arquivo JSON do aplicativo do evento *Rackathon - 2019*, disponível na figura 12, e verificar que há alguns pontos em comum e algumas diferenças fundamentais entre eles, elencadas a seguir.

As *tags* utilizadas nos três exemplos são, em essência, as mesmas, o que reafirma a facilidade no reúso desta solução. Assim como no exemplo anterior, a diferença fundamental entre eles está no conteúdo das *tags* que varia de uma aplicação para a outra. A criação da especificação do aplicativo do *Saveur Restaurant* a partir da especificação do aplicativo do *shopping* Barra Azul ou do *Rackathon - 2019* consiste na inserção das *tags* com as informações do restaurante. É possível visualizar os arquivos de especificação JSON completos no Apêndice A.

Com esse exemplo, foi possível explorar mais uma aplicação do modelo proposto. Cabe ressaltar que o aplicativo pode ser modificado tanto em aspectos de dados, quanto de *layout* e ainda de funcionalidades. Por exemplo, o estabelecimento pode alterar dados ao determinar quais itens devem constar de seu menu. Pode alterar o *layout* inserindo seu logotipo, por exemplo. E pode alterar funcionalidades, por exemplo, ao utilizar outros recursos nativos do dispositivo, como NFC (*Near-Field Communication* ou Comunicação por campo de proximidade) para realizar o pagamento. Da mesma forma que os aplicativos anteriores, o uso do aplicativo do restaurante é limitado, primordialmente, ao período em que usuário está no estabelecimento. Portanto, pode ser arquivado e excluído depois, seja por tempo, seja por localização, liberando recursos do dispositivo móvel sem ônus para o usuário final.

5.1.4 Resultados da construção dos exemplos

A construção dos três exemplos descritos anteriormente indica que o modelo e arquitetura propostos são viáveis e aplicáveis a situações cotidianas de usuários de dispositivos móveis. O ciclo de vida desses três aplicativos – desde a elaboração de sua especificação, transmissão, geração e utilização do aplicativo pelo usuário final, e, finalmente, o gerenciamento de ciclo de vida – pode ser descrito e implementado utilizando o modelo e a arquitetura desenvolvidos para aplicações de curta duração. As funcionalidades e recursos

Figura 14 – JSON *Saveur Restaurant*

```

1  {
2    "app": [
3      {
4        "nm": "Saveur Restaurant",
5        "lc": [
6          {
7            "ra": "100",
8            "pt": "-23.541189,-46.301691"
9          }
10       ],
11       "l": [
12         {
13           "id": "3",
14           "n": "rest",
15           "ac": [
16             {
17               "id": "1",
18               "nm": "Efetuar pagamento",
19               "pk": "com.mestrado.api",
20               "cs": "Pagamento",
21               "mt": "efetuarPagamentov1",
22               "pe": [
23                 {
24                   "id": "1",
25                   "nm": "valor",
26                   "tp": "double"
27                 }
28               ],
29               "ps": [
30                 {
31                   "id": "1",
32                   "nm": "sucesso",
33                   "tp": "boolean"
34                 }
35               ]
36             }
37           ],
38           "lst": [
39             {
40               "id": "1",
41               "nm": "tipo",
42               "vl": "Prato Principal"
43             },
44             {
45               "id": "1",
46               "nm": "nome",
47               "vl": "Bife com fritas"
48             },
49             {
50               "id": "1",
51               "nm": "descricao",
52               "vl": "Contra filet com batatas fritas"
53             },
54             {
55               "id": "1",
56               "nm": "preco",
57               "vl": "33.77"
58             },
59             {
60               "id": "1",
61               "nm": "imagem",
62               "vl": "https://bit.ly/2CAGswW"
63             },

```

Fonte: André Ramos Moreira, 2019

explorados, assim como desafios, decisões de implementação e dificuldades encontradas nessa tarefa, serão descritas nas seções a seguir.

5.2 *Desafios e decisões de implementação*

A seguir são listadas os principais desafios e dificuldades encontrados durante o desenvolvimento, sendo eles: (i) Desenvolvimento do sistema gerador de especificação; (ii) Gerenciamento das permissões dos aplicativos gerados; e (iii) Gerenciamento do ciclo de vida por tempo e por localização.

Desenvolvimento do sistema gerador de especificação: Para esse modelo não foi desenvolvido um sistema que gera automaticamente as especificações dos aplicativos. As especificações são feitas em JSON, de acordo com a necessidade de cada aplicativo.

Gerenciamento das permissões dos aplicativos gerados: Para cada aplicativo, o sistema operacional Android exige uma lista de permissão com consentimento do usuário para liberação de uso dos recursos do *smartphone*. O aplicativo gerador deve gerenciar essa lista de permissões para que, após a instalação do aplicativo gerado, o usuário possa executar as funcionalidades sem que o sistema operacional bloqueie alguma funcionalidade em função das permissões de acesso. A alternativa encontrada para esse desafio foi utilizar uma solução semelhante à usada nas lojas de aplicativos: a solicitação de todas as permissões antes da instalação do aplicativo gerador.

Gerenciamento do ciclo de vida por tempo e por localização: O ciclo de vida do aplicativo pode ser gerenciado por tempo, por localização ou pelo próprio usuário do aplicativo gerador. Nas duas primeiras situações, o arquivamento ou exclusão do aplicativo é feito pelo aplicativo gerador sem a necessidade de ação do usuário, de acordo com a especificação. Por exemplo, se o usuário do aplicativo do restaurante está a uma distância do estabelecimento, o aplicativo não é mais necessário. Em outra situação, passados os dias do evento, o aplicativo não será mais usado. A forma de controle do ciclo de vida é transmitida na especificação e é feita pelo aplicativo gerador. A cada abertura do aplicativo gerador, a lista de aplicativos ativos é consultada e nesse momento é feita a verificação de quais devem permanecer ativos e quais devem ser arquivados ou excluídos, de acordo com as diferentes especificações. Caso algum aplicativo deva ser arquivado ou excluído, essa tarefa é feita de forma automática pelo aplicativo gerador, sem a necessidade de intervenção pelo usuário final.

5.2.1 Funcionalidades exploradas

Com o desenvolvimento dos exemplos apresentados na seção 5.1, foi possível explorar e validar uma série de funcionalidades, sendo as principais:

- Elaboração da especificação do aplicativo a ser gerado utilizando a linguagem de marcação JSON.
- Transmissão da especificação do aplicativo para o dispositivo móvel.
- Geração do aplicativo utilizando o aplicativo gerador.
- Gerenciamento do ciclo de vida do aplicativo, que contempla arquivamento do aplicativo, arquivamento dos dados, reativação do aplicativo, reativação dos dados e exclusão do aplicativo.

5.3 Por que foi escolhida a plataforma Android

O aplicativo gerador foi desenvolvido para a plataforma Android devido à sua flexibilidade, liberdade para desenvolvimento, por ser baseada em Linux de código aberto, e à sua disponibilidade em diversos dispositivos de diferentes formatos (DEVELOPERS, 2011). Além disso, a plataforma Android é líder absoluta de mercado atualmente, com mais de 85% do *marketshare* mundial de *smartphones*¹, e com maior base de conhecimento acessível para desenvolvimento de novas aplicações.

5.3.1 Recursos nativos explorados

Para ilustrar a real aplicabilidade do modelo e arquitetura propostos foram desenvolvidos os exemplos listados na seção 5.1, utilizando uma série de recursos nativos do Android, como câmera, mapa, agenda, entre outros. A câmera é utilizada para obter a especificação dos aplicativos. A agenda e o mapa são utilizados no aplicativo exemplo do *Rackthon*, descrito na seção 5.1.2, para adicionar as atividades que o participante deseja participar e o local onde essas atividades ocorrerão.

Com isso, foi possível validar não apenas a aplicabilidade do modelo e arquitetura propostos para diferentes tipos de aplicativos, mas também a interface do modelo com

¹ Gartner, 2018. Disponível em <https://www.gartner.com/newsroom/id/3859963>. Acessado em 11.12.2018

diversos recursos nativos do Android. Essa comunicação permite que uma gama grande recursos sejam utilizados nos aplicativos gerados apenas com o acionamento de recursos já existentes no *smartphone*, sem a necessidade de desenvolvimento de novas funcionalidades.

5.4 Discussão

Comparando o modelo proposto com as atuais principais formas de distribuição de software para dispositivos móveis, *App Stores*, verifica-se que ele possui vantagens: economia de dados, acesso independente de disponibilidade de rede, pois é possível trabalhar *off-line*, e o gerenciamento do ciclo de vida que permite economia de recursos como espaço de memória e processamento do dispositivo móvel.

A economia de dados é obtida por meio da construção do aplicativo em tempo de execução, baseado na leitura da especificação, sem a necessidade de *downloads* de aplicativos e suas atualizações, ao contrário do que acontece nas *App Stores*. O acesso ao aplicativo independente de disponibilidade de rede se dá devido aos mesmos fatores: enquanto para acessar um aplicativo de uma *App Store*, o celular precisa estar conectado em algum tipo de rede; com a abordagem proposta, essa restrição não se faz necessária.

Além disso, quando o usuário instala um aplicativo oriundo de *App Stores*, ele é o responsável por excluir os aplicativos e os dados quando não há mais uso. O usuário pode ter receio de excluir algo importante, ou pode simplesmente se esquecer de fazer a exclusão. Por isso, o aplicativo pode ficar por muito tempo ocupando a memória sem uso. Por outro lado, com gerenciamento de ciclo de vida proposto, o aplicativo pode ser automaticamente excluído ou arquivado quando deixa de ser necessário ao usuário do dispositivo.

É importante destacar que o modelo de ciclo de vida baseia-se nos princípios de linha de produto de software, fábrica de software e arcabouços (*frameworks*). Ele é voltado a aplicativos para dispositivos móveis com características específicas como trivialidade de funções, ciclo de vida limitado, conjunto de dados pré-existentes, botões relacionados a ações e *layout* corriqueiro. Dessa maneira, o modelo de ciclo de vida consiste em uma linha de produto de software com reaproveitamento de artefatos; por exemplo, *layouts* e código para invocar ações (*APIs – Application Programming Interfaces*).

O modelo também pode ser visto como uma fábrica de software que utiliza um *framework*. Mais especificamente, o conceito de *framework* é utilizado na etapa de concepção

da especificação do aplicativo utilizando a linguagem de descrição de aplicativos e a arquitetura de gerenciamento de ciclo de vida. Porém, o desenvolvimento de uma ferramenta para construir essa especificação de forma automatizada e simplificada está fora do escopo deste trabalho.

Não há restrições a tipos de produto ou serviços para os aplicativos gerados pela abordagem. Diferentes domínios, além daqueles dos exemplos aqui apresentados, podem ser atendidos pela solução proposta. O modelo e arquitetura atendem especificações de aplicativos que podem variar de acordo com as diferentes demandas, além de permitir o gerenciamento do ciclo de vida variável por tempo, por localização ou por eventuais outros critérios estabelecidos.

É interessante destacar que o tempo consumido para a geração dos aplicativos do *Rackthon - 2019* e do *Saveur Restaurant* foi consideravelmente menor do que o de desenvolvimento do primeiro aplicativo exemplo, *Barra Azul Shopping*. O primeiro aplicativo demandou mais de 48 horas de desenvolvimento enquanto o *Rackathon - 2019* consumiu apenas 16, em sua maioria para utilização da funcionalidade nativa do Android – Agenda – e poucas para adaptação do *layout* e inserção de conteúdo. Para o desenvolvimento do aplicativo do *Saveur Restaurant*, o tempo gasto foi ainda menor, apenas 8 horas, voltadas basicamente para elaboração do carrinho de compras. O consumo de horas foi bastante reduzido devido, principalmente, ao alto reúso que a solução desenvolvida neste trabalho permite. Conforme visualizável nas figuras 4, 12 e 14 e também no Apêndice A, os arquivos de especificação JSON são extremamente similares entre si, gerando alto reaproveitamento de código e economia de recursos. Apesar de o número de linhas ser diferente entre as três especificações dos aplicativos, boa parte delas são variações apenas em conteúdo, sem afetar o desenvolvimento de sistemas.

Outro ponto que deve ser destacado é a possível aplicação do modelo desenvolvido em outras plataformas, principalmente no iOS, segunda maior plataforma em *marketshare*². A aplicação nessa plataforma depende de alguns requisitos diferentes do Android. Por exemplo, a requisição de permissões para uso de recursos do celular é bem mais restrita no iOS do que no Android. Porém, a aplicabilidade do modelo é independente de plataforma, sendo necessárias apenas alterações de questões técnicas de acordo com cada plataforma.

² Gartner, 2018. Disponível em <https://www.gartner.com/newsroom/id/3859963>. Acessado em 11.12.2018

Portanto, independente de plataforma, o conceito desenvolvido neste trabalho é válido para todos os aplicativos de dispositivos móveis que possuam ciclo de vida limitado.

5.5 Limitações

Essa pesquisa desenvolveu uma arquitetura e um modelo de ciclo de vida para aplicações de curta duração para dispositivos móveis. A arquitetura e o modelo são restritos a aplicativos cujo ciclo de vida seja curto, aplicados a um período limitado de tempo, com funcionalidades e *layout* triviais. A arquitetura e o modelo também estão desenvolvidos especificamente para dispositivos móveis que utilizam o sistema operacional Android, sem validação para outras plataformas de dispositivos móveis.

Apesar dessa proposta ter sido implementada utilizando uma linguagem de especificação genérica, a complexidade dos aplicativos especificados está vinculada ao limite de representação dessa linguagem. Neste trabalho, também não foi desenvolvido um módulo gerador de especificações automatizadas.

O modelo desenvolvido não define a forma de transmissão da especificação para o aplicativo gerenciador, pois ela está fora do escopo do trabalho. Entretanto, a transmissão da especificação irá requerer implementação específica para os diferentes meios.

Estão prontos para uso, sem necessidade de nenhuma ação, os aplicativos que possuem os *layouts* já implementados (*shopping*, restaurante e evento). Esses *layouts* podem ser alterados sem necessidade de implementação. Por exemplo, o *layout* utilizado no aplicativo do *shopping* pode ser utilizado em um museu sem alterações além da aparência, porém, para aplicativos que demandem *layouts* totalmente distintos, é necessária a implementação de um *layout* específico para tal fim.

As atribuições das ações dos aplicativos devem ser simples com chamadas diretas a serviços ou APIs. Não podem ser utilizadas estruturas ou formas complexas; por exemplo, o programa desenvolvido não tem a capacidade de trabalhar com requisições do tipo REST³.

A validação da aplicabilidade do modelo proposto nesse trabalho é voltada apenas para os aplicativos de exemplo implementados (*shopping*, restaurante e evento). Não foram feitas outras validações, que são sugeridas como trabalhos futuros no capítulo 6.

³ What is REST? - <https://www.codecademy.com/articles/what-is-rest> - Acessado em 05.01.2019

Esse trabalho é totalmente voltado para atender aos problemas de geração, distribuição e gerenciamento do ciclo de vida de aplicativos de vida curta.

5.6 *Considerações finais*

Neste capítulo foi apresentada a validação do modelo e arquitetura desenvolvidos neste trabalho. O capítulo iniciou com a apresentação dos três exemplos implementados (*shopping*, evento e restaurante) e os resultados obtidos com a implementação. Também foram descritos os principais desafios e decisões de implementação, assim como as razões para escolha da plataforma utilizada. Foi apresentada ainda uma discussão a respeito das implementações, resultados obtidos e as limitações deste trabalho. A partir dos resultados e da validação realizada, as conclusões desse trabalho são apresentadas no próximo capítulo.

6 Conclusão

Nesse capítulo são apresentadas as conclusões obtidas com a aplicação do modelo e arquitetura propostos e desenvolvidos neste trabalho. É apresentada uma síntese do trabalho, as contribuições realizadas e sugestões de trabalhos futuros.

6.1 Síntese do trabalho

Os aplicativos de curta duração que possuem funcionalidades triviais são utilizados por um período limitado de tempo e para fins específicos, perdendo a utilidade uma vez que esse fim específico é atingido. No entanto, geralmente eles são mantidos pelos usuários nos dispositivos móveis, causando alguns problemas como consumo de rede, uso desnecessário de espaço de armazenamento e até mesmo possível comprometimento da segurança do dispositivo.

Esta dissertação apresentou um modelo de ciclo de vida e uma arquitetura para geração de aplicativos de curta duração. Os aplicativos são descritos utilizando uma linguagem de especificação que permite expressar aplicativos com ciclo de vida curto, de uso limitado e *layout* trivial para diferentes domínios de aplicação. A especificação é lida por um aplicativo gerador e gerenciador de ciclo de vida de aplicativos que gera, controla, arquiva e exclui de forma automática os aplicativos, reduzindo e solucionando os problemas citados anteriormente.

O ciclo de vida desenvolvido inicia com a geração do aplicativo a partir da especificação que descreve suas funcionalidades e o tipo de *layout* necessário. O aplicativo é utilizado e reutilizado tantas vezes quantas necessárias, de acordo com as demandas do usuário e do domínio em que está inserido. Em seguida, há duas possibilidades: arquivamento ou exclusão do aplicativo. Quando um aplicativo é arquivado, ele pode ser recuperado para usos posteriores ou pode ser excluído no futuro. O modelo proposto contempla todas as fases desde a concepção do aplicativo, sua geração, uso, arquivamento e recuperação, concluindo com a exclusão, ou seja, do início ao fim da vida útil do aplicativo.

A arquitetura é composta de um módulo de especificação externo, chamado *Especificador*, e três módulos internos, a saber, *Leitor e Tradutor de Especificações*, *Gerenciador* e o *Aplicativo* especificado (gerado). O Especificador não foi desenvolvido de forma auto-

matizada neste trabalho. O módulo Leitor e Tradutor faz a leitura da especificação e gera o aplicativo especificado; o Gerenciador configura o aplicativo de acordo com a informação recebida e gerencia o ciclo de vida do aplicativo. Dessa maneira, ele é responsável pela disponibilização, arquivamento e exclusão do aplicativo, conforme a necessidade do usuário. O último módulo interno é o próprio aplicativo gerado pronto para utilização.

O modelo e arquitetura desenvolvidos foram utilizados em três aplicativos exemplos, seguindo suas especificações, com características leves, simples, *layout* trivial e funcionalidades básicas. Essa aplicação demonstrou que o modelo e arquitetura propostos e desenvolvidos nesse trabalho atendem às necessidades de aplicativos em diferentes domínios (restaurante, *shopping* e evento) e pode ser aplicado em situações com diferentes restrições (tempo, local, etc.).

6.2 Contribuições

Este trabalho contribui para a área de engenharia de software, especificamente, para o desenvolvimento de aplicativos de curta duração para dispositivos móveis. Ele apresenta um modelo de ciclo de vida e uma arquitetura que promovem reúso de recursos, economizam espaço e processamento do dispositivo e reduzem os riscos aos quais os usuários de dispositivos móveis estão expostos.

Especificamente, as principais contribuições deste trabalho são:

- um modelo de ciclo de vida para geração e controle de aplicativos de curta duração inovador, sem similares na academia e na indústria, conforme pesquisa bibliográfica realizada;
- uma arquitetura única, própria para o modelo de ciclo de vida desenvolvido;
- um aplicativo gerador e gerenciador de aplicativos de curta duração com aplicação do modelo de ciclo de vida e arquitetura desenvolvidos;
- três aplicativos de curta duração desenvolvidos utilizando o modelo e a arquitetura propostos, com funcionalidades e domínios distintos (*shopping center*, *evento* e *restaurante*); e
- notação para especificação de aplicativos de curta duração utilizando JSON.

A abordagem proposta nesta dissertação apresenta vantagens em relação aos modelos de distribuição de aplicativos existentes, sendo as principais: economia de dados, acesso

independente de disponibilidade de rede e economia de recursos (espaço de memória e processamento) no dispositivo móvel.

6.3 *Trabalhos futuros*

Como trabalhos futuros, sugere-se conduzir experimentos para avaliar a adesão dos usuários e desenvolvedores à abordagem proposta. Os desenvolvedores utilizariam o modelo e a arquitetura para desenvolver um novo aplicativo. Essa abordagem seria comparada com outras técnicas para desenvolvimento de aplicativos para dispositivos móveis em termos de tamanho da especificação e número de horas para criar uma aplicação, por exemplo. O modelo de aceitação de tecnologia (TAM – *Technology Acceptance Model*), proposto por Davis (1989), poderia ser utilizado para avaliar a aceitação pelos usuários da tecnologia dos aplicativos desenvolvidos utilizando o modelo e a arquitetura propostos. Também se sugere a aplicação da abordagem a aplicativos de domínios distintos dos de restaurantes, *shoppings* e eventos.

Trabalhos futuros tecnológicos incluem a criação do módulo de especificação dos aplicativos e a aplicação do modelo e da arquitetura à plataforma iOS. O primeiro trabalho ajudaria o desenvolvedor a criar aplicações mais rapidamente e menos sujeitas a erros, pois atualmente ele precisa criar um arquivo JSON manualmente. A implementação da abordagem para plataforma iOS atenderia aproximadamente 14% do mercado de dispositivos móveis¹.

Por fim, sugere-se aplicar métodos de segurança na transmissão do arquivo de especificação.

¹ Gartner, 2018. Disponível em <https://www.gartner.com/newsroom/id/3859963>. Acessado em 11.12.2018

Referências²

- ABRAHAMSSON, P.; HANHINEVA, A.; HULKKO, H.; IHME, T.; JÄÄLINOJA, J.; KORKALA, M.; KOSKELA, J.; KYLLÖNEN, P.; SALO, O. Mobile-D: an agile approach for mobile application development. In: ACM. *Companion to the 19th annual ACM SIGPLAN conference on Object-oriented programming systems, languages, and applications*. New York, NY, USA: ACM, 2004. (OOPSLA '04), p. 174–175. ISBN 1-58113-833-4. Disponível em: <http://doi.acm.org/10.1145/1028664.1028736>. Citado 2 vezes nas páginas 33 e 39.
- AHMET, Z.; HOLMQUIST, L.-E. Sharing mobile services: beyond the app store model. In: ACM. *Proceedings of the 12th international conference on Human computer interaction with mobile devices and services*. New York, NY, USA: ACM, 2010. (MobileHCI '10), p. 383–384. ISBN 978-1-60558-835-3. Disponível em: <http://doi.acm.org/10.1145/1851600.1851676>. Citado 2 vezes nas páginas 35 e 39.
- AL-AKKAD, A.; RAFFELSBERGER, C. How do I get this app? A discourse on distributing mobile applications despite disrupted infrastructure. In: ISCRAM. *ISCRAM*. Toulouse, France, 2014. Citado na página 34.
- BASS, L.; CLEMENTS, P.; COHEN, S.; NORTHROP, L.; WITHEY, J. *Product Line Practice Workshop Report*. Pittsburg, CA, USA, 1997. Citado na página 27.
- BASTANI, O.; ANAND, S.; AIKEN, A. Interactively verifying absence of explicit information flows in android apps. *ACM SIGPLAN Notices*, ACM, v. 50, n. 10, p. 299–315, 2015. Citado 2 vezes nas páginas 37 e 39.
- BEIMBORN, D.; PALITZA, M. Enterprise app stores for mobile applications-development of a benefits framework. *Proceedings of the 19th Americas Conference on Information Systems*, AMCIS, Chicago, IL, 2013. Citado 2 vezes nas páginas 34 e 39.
- BEZERRA, E. *Princípios de Análise e Projeto de Sistemas com UML*. Elsevier Editora, 2007. ISBN 9788535226263. Disponível em: <https://books.google.com.br/books?id=O45DvgAACAAJ>. Citado 3 vezes nas páginas 19, 20 e 21.
- BIEGEL, G.; CAHILL, V. A framework for developing mobile, context-aware applications. In: IEEE. *Pervasive Computing and Communications, 2004. PerCom 2004. Proceedings of the Second IEEE Annual Conference on*. Orlando, FL, USA, 2004. p. 361–365. Citado na página 33.
- BÖHMER, M.; HECHT, B.; SCHÖNING, J.; KRÜGER, A.; BAUER, G. Falling asleep with Angry Birds, Facebook and Kindle: a large scale study on mobile application usage. In: ACM. *Proceedings of the 13th international conference on Human computer interaction with mobile devices and services*. New York, NY, USA: ACM, 2011. (MobileHCI '11), p. 47–56. ISBN 978-1-4503-0541-9. Disponível em: <http://doi.acm.org/10.1145/2037373.2037383>. Citado na página 14.
- BURES, T.; HNETYNKA, P.; PLASIL, F. Strengthening architectures of smart cps by modeling them as runtime product-lines. In: ACM. *Proceedings of the 17th international ACM Sigsoft symposium on Component-based software engineering*. New York, NY,

² De acordo com a Associação Brasileira de Normas Técnicas. NBR 6023.

USA: ACM, 2014. (CBSE '14), p. 91–96. ISBN 978-1-4503-2577-6. Disponível em: <http://doi.acm.org/10.1145/2602458.2602478>. Citado na página 38.

BUZAN, T.; BUZAN, B. *Mind Map Book*, 1/e. New York: Rajpal & Sons, 2010. Citado na página 29.

CLEMENTS, P.; NORTHROP, L. *Software product lines*. Boston, MA, USA: Addison-Wesley, 2002. 563 p. Citado 2 vezes nas páginas 22 e 27.

DAVIS, F. D. Perceived usefulness, perceived ease of use, and user acceptance of information technology. *MIS quarterly*, JSTOR, p. 319–340, 1989. Citado na página 73.

DEB, K.; PRATAP, A.; AGARWAL, S.; MEYARIVAN, T. A fast and elitist multiobjective genetic algorithm: Nsga-ii. *IEEE transactions on evolutionary computation*, IEEE, v. 6, n. 2, p. 182–197, 2002. Citado na página 32.

DEBASTIANI, C. A. *Definindo escopo em projetos de software*. São Paulo, SP, Brasil: Novatec Editora, 2016. Citado na página 29.

DEVELOPERS, A. *What is android*. USA: Android Developers, [http://developer.android.com/guide/basics/what-is ...](http://developer.android.com/guide/basics/what-is...), 2011. Citado na página 66.

FANG, M.; LEYH, G.; DOERR, J.; ELSNER, C.; ZHAO, J. Towards model-based derivation of systems in the industrial automation domain. In: ACM. *Proceedings of the 19th International Conference on Software Product Line*. Nashville, TN, USA, 2015. p. 283–292. Citado na página 32.

FOX, A.; PATTERSON, D.; PATTERSON; KON, F.; CORDEIRO, D. *Construindo Software como Serviço: Uma Abordagem Ágil Usando Computação em Nuvem*. Strawberry Canyon LLC, 2015. ISBN 9780984881277. Disponível em: <https://books.google.com.br/books?id=CIn1rQEACAAJ>. Citado 2 vezes nas páginas 20 e 23.

FRANCESE, R.; RISI, M.; TORTORA, G. Management, sharing and reuse of service-based mobile applications. In: IEEE. *Mobile Software Engineering and Systems (MOBILESoft), 2015 2nd ACM International Conference on*. Florence, Italy, 2015. p. 105–108. Citado 2 vezes nas páginas 34 e 39.

GAMMA, E.; HELM, R.; JOHNSON, R.; VLISSIDES, J. *Design Patterns: Elements of Reusable Object-oriented Software*. Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc., 1995. ISBN 0-201-63361-2. Citado 2 vezes nas páginas 22 e 23.

GOMEZ, M.; MARTINEZA, M.; MONPERRUS, M.; ROUVOY, R. When app stores listen to the crowd to fight bugs in the wild. In: IEEE. *Software Engineering (ICSE), 2015 IEEE/ACM 37th IEEE International Conference on*. Florence, Italy, 2015. v. 2, p. 567–570. Citado 2 vezes nas páginas 36 e 39.

GREENFIELD, J.; SHORT, K. Software factories: assembling applications with patterns, models, frameworks and tools. In: ACM. *Companion of the 18th annual ACM SIGPLAN conference on Object-oriented programming, systems, languages, and applications*. New York, NY, USA: ACM, 2003. (OOPSLA '03), p. 16–27. ISBN 1-58113-751-6. Disponível em: <http://doi.acm.org/10.1145/949344.949348>. Citado na página 32.

GUZMAN, E.; BHUVANAGIRI, P.; BRUEGGE, B. Fave: Visualizing user feedback for software evolution. In: IEEE. *Software Visualization (VISSOFT), 2014 Second IEEE Working Conference on*. Victoria, BC, Canada, 2014. p. 167–171. Citado 2 vezes nas páginas 38 e 39.

HARRISON, R. *Symbian OS C++ for mobile phones (With CD)*. West Sussex, England: John Wiley & Sons, 2003. v. 1. Citado na página 33.

JAAKSI, A. Developing mobile browsers in a product line. *IEEE software*, IEEE, v. 19, n. 4, p. 73–80, 2002. Citado 3 vezes nas páginas 27, 31 e 39.

KEOGH, J.; TOPLEY, B. K. *J2ME*. Hyderabad, India: Osborne/McGraw-Hill, 2003. Citado na página 33.

KNOWLES, J. D.; CORNE, D. W. Approximating the nondominated front using the pareto archived evolution strategy. *Evolutionary computation*, MIT Press, v. 8, n. 2, p. 149–172, 2000. Citado na página 32.

KUDLA, T.; ESCH, M.; OTA, D.; SCHWARZ, G. An open process for software development in classified environments as prerequisite for military app stores. In: IEEE. *Military Communications and Information Systems (ICMCIS), 2015 International Conference on*. Cracow, Poland, 2015. p. 1–9. Citado 2 vezes nas páginas 34 e 39.

LARSEN, P.; HOMESCU, A.; BRUNTHALER, S.; FRANZ, M. Sok: Automated software diversity. In: IEEE. *Security and Privacy (SP), 2014 IEEE Symposium on*. San Jose, CA, USA, 2014. p. 276–291. Citado 2 vezes nas páginas 38 e 39.

MARTIN, W.; SARRO, F.; JIA, Y.; ZHANG, Y.; HARMAN, M. A survey of app store analysis for software engineering. *IEEE Transactions on Software Engineering*, IEEE, 2016. Citado na página 14.

MEDVIDOVIC, N.; TAYLOR, R. N. Software architecture: foundations, theory, and practice. In: ACM. *Proceedings of the 32nd ACM/IEEE International Conference on Software Engineering-Volume 2*. New York, NY, USA: ACM, 2010. (ICSE '10), p. 471–472. ISBN 978-1-60558-719-6. Disponível em: <http://doi.acm.org/10.1145/1810295.1810435>. Citado na página 24.

MIZOUNI, R.; MATAR, M. A.; MAHMOUD, Z. A.; ALZAHMI, S.; SALAH, A. A framework for context-aware self-adaptive mobile applications SPL. *Expert Systems with applications*, Elsevier, v. 41, n. 16, p. 7549–7564, 2014. Citado 2 vezes nas páginas 32 e 39.

MYLLÄRNIEMI, V.; KUJALA, S.; RAATIKAINEN, M.; SEVOÑN, P. Development as a journey: factors supporting the adoption and use of software frameworks. *Journal of Software Engineering Research and Development*, v. 6, n. 1, p. 6, Jun 2018. ISSN 2195-1721. Disponível em: <https://doi.org/10.1186/s40411-018-0050-8>. Citado 4 vezes nas páginas 22, 24, 25 e 26.

PAHL, M.-O.; CARLE, G. Taking smart space users into the development loop: An architecture for community based software development for smart spaces. In: ACM. *Proceedings of the 2013 ACM conference on Pervasive and ubiquitous computing adjunct publication*. New York, NY, USA: ACM, 2013. (UbiComp '13 Adjunct), p. 793–800. ISBN 978-1-4503-2215-7. Disponível em: <http://doi.acm.org/10.1145/2494091.2497320>. Citado 2 vezes nas páginas 35 e 39.

PARNAS, D. L. On the criteria to be used in decomposing systems into modules. *Communications of the ACM*, ACm, v. 15, n. 12, p. 1053–1058, 1972. Citado na página 20.

PASCUAL, G. G.; LOPEZ-HERREJON, R. E.; PINTO, M.; FUENTES, L.; EGYED, A. Applying multiobjective evolutionary algorithms to dynamic software product lines for reconfiguring mobile applications. *Journal of Systems and Software*, Elsevier, v. 103, p. 392–411, 2015. Citado 2 vezes nas páginas 32 e 39.

PEREZ, S. *Consumers Spend 85Apps See Heavy Use*. 2015. Online; Acessado em 02-setembro-2017. Disponível em: <https://techcrunch.com/2015/06/22/consumers-spend-85-of-time-on-smartphones-in-apps-but-only-5-apps-see-heavy-use/>. Citado na página 14.

QUINTON, C.; DUCHIEN, L.; HEYMANS, P.; MOUTON, S.; CHARLIER, E. Using feature modelling and automations to select among cloud solutions. In: IEEE PRESS. *Proceedings of the Third International Workshop on Product Line Approaches in Software Engineering*. Zurich, Switzerland, 2012. p. 17–20. Citado na página 33.

RAHMAN, A. L. A.; ARIF, Z.; RAZALI, T. R. A. T.; ISMAIL, A. B.; SAWAL, M. Z. H. M.; OMAR, R. A. R. Review the factors influence the successful design of mobile digital library interface: Malaysian perspectives. *Advanced Science Letters*, American Scientific Publishers, v. 24, n. 7, p. 5018–5021, 2018. Citado na página 14.

RUIZ, I. M.; NAGAPPAN, M.; ADAMS, B.; BERGER, T.; DIENST, S.; HASSAN, A. An examination of the current rating system used in mobile app stores. *IEEE Software*, IEEE, 2017. Citado 2 vezes nas páginas 37 e 39.

SHAW, M.; GARLAN, D. *Software Architecture: Perspectives on an Emerging Discipline*. Upper Saddle River, NJ, USA: Prentice-Hall, Inc., 1996. ISBN 0-13-182957-2. Citado 3 vezes nas páginas 22, 23 e 24.

SHEN, L.; PENG, X.; ZHAO, W. Software product line engineering for developing self-adaptive systems: Towards the domain requirements. In: IEEE. *Computer Software and Applications Conference (COMPSAC), 2012 IEEE 36th Annual*. Izmir, Turkey, 2012. p. 289–296. Citado na página 27.

VISHWANATH, A.; HARRISON, B.; NG, Y. J. Understanding how social-behavioural science theory can explain the design of software websites. In: IEEE. *System Sciences (HICSS), 2016 49th Hawaii International Conference on*. Koloa, HI, USA, 2016. p. 5823–5830. Citado 2 vezes nas páginas 36 e 39.

WANG, Y.; NG, Y. Y.; ZHOU, H.; DONG, Y.; JI, Z. The improvement of the trustworthiness of android app stores in china. In: IEEE. *Software Engineering Conference (APSEC), 2016 23rd Asia-Pacific*. Hamilton, New Zealand, 2016. p. 357–360. Citado 2 vezes nas páginas 36 e 39.

WENZEL, S. App store models for enterprise software: A comparative case study of public versus internal enterprise app stores. In: SPRINGER. *International Conference of Software Business*. Cham, Switzerland, 2014. p. 227–242. Citado 2 vezes nas páginas 34 e 39.

Apêndice A – Especificações dos exemplos utilizados nesse trabalho no formato JSON.

1 Especificação do aplicativo evento

```
{
  "app": [
    {
      "nm": "Rackthon - 2019",
      "tm": "2500",
      "lc": [
        {
          "ra": "1000",
          "pt": "-23.541189, -46.301691"
        }
      ],
      "l": [
        {
          "id": "4",
          "n": "evento",
          "ac": [
            {
              "id": "1",
              "nm": "Adicionar calendário",
              "pk": "com.mestrado.api",
              "cs": "Calendario",
              "mt": "inclirEventoCalendario",
              "pe": [
                {
                  "id": "1",
                  "nm": "latitude",
                  "tp": "double"
                }
              ],
            }
          ],
        }
      ],
    }
  ],
}
```

```

        "id": "2",
        "nm": "longitude",
        "tp": "double"
    }
],
"ps": [
    {
        "id": "1",
        "nm": "eventID",
        "tp": "long"
    }
]
},
"a": [
    {
        "id": "1",
        "t": "Conexões diferentes. Idéias de impacto."
    }
],
"im": [
    {
        "id": "1",
        "url": "https://bit.ly/2C2ikS0"
    }
],
"lst": [
    {
        "id": "1",
        "nm": "dia",
        "vl": "1"
    },
    {

```



```

    "id": "1",
    "nm": "nome",
    "vl": "Escalabilidade de Startups"
  },
  {
    "id": "1",
    "nm": "palestrante",
    "vl": "João Wang"
  },
  {
    "id": "1",
    "nm": "data",
    "vl": "20/02/2019"
  },
  {
    "id": "1",
    "nm": "hora",
    "vl": "10:00"
  },
  {
    "id": "1",
    "nm": "duracao",
    "vl": "1:45"
  },
  {
    "id": "1",
    "nm": "local_descricao",
    "vl": "Prédio do Café - Sala 1152"
  },
  {
    "id": "1",
    "nm": "local",
    "vl": "-23.541189, -46.301691"
  }

```

```

},
{
  "id": "1",
  "nm": "descricao",
  "vl": "Toda startup quer atingir taxas de crescimentos absurdas,
        mas poucas conseguem. As poucas que conseguem, precisam
        rapidamente escalar o negocio!"
},
{
  "id": "2",
  "nm": "dia",
  "vl": "1"
},
{
  "id": "2",
  "nm": "nome",
  "vl": "Conexões Cognitivas"
},
{
  "id": "2",
  "nm": "palestrante",
  "vl": "Hu Jan Park"
},
{
  "id": "2",
  "nm": "data",
  "vl": "20/02/2019"
},
{
  "id": "2",
  "nm": "hora",
  "vl": "14:00"
},

```

```

    {
      "id": "2",
      "nm": "duracao",
      "vl": "2:00"
    },
    {
      "id": "2",
      "nm": "local_descricao",
      "vl": "Theatro Vitória"
    },
    {
      "id": "2",
      "nm": "local",
      "vl": "-23.541170, -46.301695"
    },
    {
      "id": "2",
      "nm": "descricao",
      "vl": "Conexões Cognitivas: relações estéticas, intelectuais e
            espirituais na experiência humana."
    }
  ]
}
]
}
]
}

```

2 Especificação do aplicativo restaurante

```

{
  "app": [
    {

```

```

"nm": "Saveur Restaurant",
"lc": [
  {
    "ra": "100",
    "pt": "-23.541189,-46.301691"
  }
],
"l": [
  {
    "id": "3",
    "n": "rest",
    "ac": [
      {
        "id": "1",
        "nm": "Efetuar pagamento",
        "pk": "com.mestrado.api",
        "cs": "Pagamento",
        "mt": "efetuarPagamentov1",
        "pe": [
          {
            "id": "1",
            "nm": "valor",
            "tp": "double"
          }
        ],
        "ps": [
          {
            "id": "1",
            "nm": "sucesso",
            "tp": "boolean"
          }
        ]
      }
    ]
  }
]

```

```
],  
"lst": [  
  {  
    "id": "1",  
    "nm": "tipo",  
    "vl": "Prato Principal"  
  },  
  {  
    "id": "1",  
    "nm": "nome",  
    "vl": "Bife com fritas"  
  },  
  {  
    "id": "1",  
    "nm": "descricao",  
    "vl": "Contra filet com batatas fritas"  
  },  
  {  
    "id": "1",  
    "nm": "preco",  
    "vl": "33.77"  
  },  
  {  
    "id": "1",  
    "nm": "imagem",  
    "vl": "https://bit.ly/2CAGswW"  
  },  
  {  
    "id": "2",  
    "nm": "tipo",  
    "vl": "Prato Principal"  
  },  
  {
```

```
    "id": "2",
    "nm": "nome",
    "vl": "Feijoadada"
  },
  {
    "id": "2",
    "nm": "descricao",
    "vl": "Feijoadada com acompanhamentos"
  },
  {
    "id": "2",
    "nm": "preco",
    "vl": "27.35"
  },
  {
    "id": "2",
    "nm": "imagem",
    "vl": "https://bit.ly/2ENUMne"
  },
  {
    "id": "3",
    "nm": "tipo",
    "vl": "Sobremesa"
  },
  {
    "id": "3",
    "nm": "nome",
    "vl": "Surpresa de morango"
  },
  {
    "id": "3",
    "nm": "descricao",
    "vl": "Bolo com morangos e merengue"
```

```
},
{
  "id": "3",
  "nm": "preco",
  "vl": "12.22"
},
{
  "id": "3",
  "nm": "imagem",
  "vl": "https://bit.ly/2Rg06V5"
},
{
  "id": "4",
  "nm": "tipo",
  "vl": "Sobremesa"
},
{
  "id": "4",
  "nm": "nome",
  "vl": "Mil folhas morango"
},
{
  "id": "4",
  "nm": "descricao",
  "vl": "Massa folheada recheada"
},
{
  "id": "4",
  "nm": "preco",
  "vl": "15.50"
},
{
  "id": "4",
```

```
    "nm": "imagem",
    "vl": "https://bit.ly/2FcDvnL"
  },
  {
    "id": "5",
    "nm": "tipo",
    "vl": "Sobremesa"
  },
  {
    "id": "5",
    "nm": "nome",
    "vl": "Hot pot de chocolate"
  },
  {
    "id": "5",
    "nm": "descricao",
    "vl": "Sorvete coberto com chocolate"
  },
  {
    "id": "5",
    "nm": "preco",
    "vl": "8.73"
  },
  {
    "id": "5",
    "nm": "imagem",
    "vl": "https://bit.ly/2Q1mNcJ"
  },
  {
    "id": "6",
    "nm": "tipo",
    "vl": "Bebidas"
  },
}
```



```
{
  "id": "6",
  "nm": "nome",
  "vl": "Refrigerante"
},
{
  "id": "6",
  "nm": "descricao",
  "vl": "Lata refrigerante - 350 ml"
},
{
  "id": "6",
  "nm": "preco",
  "vl": "5.50"
},
{
  "id": "6",
  "nm": "imagem",
  "vl": "https://bit.ly/2Ta3eB2"
},
{
  "id": "7",
  "nm": "tipo",
  "vl": "Bebidas"
},
{
  "id": "7",
  "nm": "nome",
  "vl": "Suco Natural"
},
{
  "id": "7",
  "nm": "descricao",
```

```

        "v1": "Copo de suco natural - 400 ml"
    },
    {
        "id": "7",
        "nm": "preco",
        "v1": "9.70"
    },
    {
        "id": "7",
        "nm": "imagem",
        "v1": "https://bit.ly/2VasVDh"
    }
]
}
]
}
]
}

```

3 Especificação do aplicativo *shopping*

```

{
  "app": [
    {
      "nm": "Barra Azul Shopping",
      "tm": "250",
      "lc": [
        {
          "ra": "300",
          "pt": "-22.541189,-45.301691"
        }
      ],
      "l": [

```

```

{
  "id": "2",
  "n": "shop",
  "ac": [
    {
      "id": "1",
      "nm": "Cadastro Promoção",
      "pk": "com.mestrado.api",
      "cs": "Cadastro",
      "mt": "efetuarCadastroNv1",
      "pe": [
        {
          "id": "1",
          "nm": "serverResposta",
          "tp": "String"
        }
      ],
      "ps": [
        {
          "id": "1",
          "nm": "sucesso",
          "tp": "boolean"
        }
      ]
    }
  ],
  "im": [
    {
      "id": "1",
      "url": "https://bit.ly/2Aiuczh"
    }
  ],
  "a": [

```

```

{
  "id": "1",
  "v": "Oferta: desconto de 30% na Loja Baiana"
},
{
  "id": "2",
  "v": "Especial: Natal Barra Azul Shopping -
  Cadastre-se na promoção!"
}
],
"lst": [
  {
    "id": "1",
    "nm": "nome",
    "vl": "Saveur Restaurant"
  },
  {
    "id": "1",
    "nm": "local",
    "vl": "Praça de alimentação - Loja 55"
  },
  {
    "id": "1",
    "nm": "tipo",
    "vl": "Alimentício"
  },
  {
    "id": "2",
    "nm": "nome",
    "vl": "Loja Baiana"
  },
  {
    "id": "2",

```

```

        "nm": "local",
        "vl": "1º andar - Loja 10"
    },
    {
        "id": "2",
        "nm": "tipo",
        "vl": "Vestuário"
    },
    {
        "id": "3",
        "nm": "nome",
        "vl": "Ferragens do Miguel"
    },
    {
        "id": "3",
        "nm": "local",
        "vl": "1º andar - Loja 209"
    },
    {
        "id": "3",
        "nm": "tipo",
        "vl": "Utensílios domésticos"
    }
]
}
]
}
]
}

```