

INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA DO CEARÁ
DEPARTAMENTO DE TELEMÁTICA
PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

Abner Augusto Lima de Oliveira

**Models On Paper - Utilizando Prototipação em Papel em um
Processo Orientado a Modelos para o Suporte à Simulação de
Aplicações Android**

Fortaleza – CE

31 de outubro de 2017

Abner Augusto Lima de Oliveira

**Models On Paper - Utilizando Prototipação em Papel em
um Processo Orientado a Modelos para o Suporte à
Simulação de Aplicações Android**

Dissertação submetida ao Programa de Pós-Graduação em Ciência da Computação do Instituto Federal de Educação, Ciência e Tecnologia do Ceará, como requisito parcial para obtenção do grau de Mestre em Ciência da Computação.

Área de Concentração: Engenharia de Software

Orientador: Prof. Dr. Cidcley Teixeira de Souza

Instituto Federal de Educação, Ciência e Tecnologia do Ceará (IFCE)
Programa de Pós-Graduação em Ciência da Computação (PPGCC)

Fortaleza – CE

31 de outubro de 2017

Dados Internacionais de Catalogação na Publicação
Instituto Federal do Ceará - IFCE
Sistema de Bibliotecas - SIBI
Ficha catalográfica elaborada pelo SIBI/IFCE, com os dados fornecidos pelo(a) autor(a)

- O48m Oliveira, Abner Augusto Lima de.
 Models On Paper : Utilizando Prototipação em Papel em um Processo Orientado a Modelos para o
 Suporte à Simulação de Aplicações Android / Abner Augusto Lima de Oliveira. - 2017.
 113 f. : il. color.
- Dissertação (Mestrado) - Instituto Federal do Ceará, Mestrado em Ciência da Computação, Campus
Fortaleza, 2017.
 Orientação: Prof. Dr. Cidcley Teixeira de Souza.
1. Prototipação. 2. Android. 3. Desenvolvimento orientado a modelos. 4. Reconhecimento de objetos. I.
 Titulo.

CDD 004

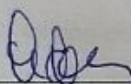
ABNER AUGUSTO LIMA DE OLIVEIRA

Models On Paper: Utilizando Prototipação em Papel em um Processo Orientado a Modelos
para o Suporte à Simulação de Aplicações Android

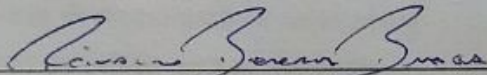
Dissertação submetida à Coordenação do Curso de Pós-graduação em Ciência da
Computação do Instituto Federal de Educação, Ciência e Tecnologia do Ceará, como
requisito parcial para a obtenção do título de Mestre em Ciência da Computação, área de
concentração Ciência da Computação.

Aprovada em 28 / 09 / 2017.

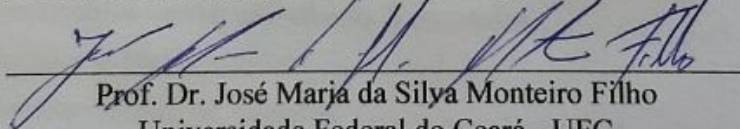
BANCA EXAMINADORA



Prof. Dr. Cidcley Teixeira de Souza (Orientador)
Instituto Federal de Educação, Ciência e Tecnologia do Ceará - PPGCC/IFCE



Prof. Dr. Reinaldo Bezerra Braga
Instituto Federal de Educação, Ciência e Tecnologia do Ceará - PPGCC/IFCE



Prof. Dr. José Maria da Silva Monteiro Filho
Universidade Federal do Ceará - UFC

*Dedico este trabalho a minha companheira Pedrita Bessa e ao meu filho Leônidas Augusto,
pelo apoio incondicional durante essa empreitada.*

Agradecimentos

Ao meu orientador, Dr. Cidcley Teixeira de Souza, por sempre estar disponível, pela paciência, pelo incentivo, e grande colaboração ao longo do desenvolvimento desta dissertação.

Ao Programa de Pós-Graduação em Ciência da Computação (PPGCC) do Instituto Federal de Educação, Ciência e Tecnologia do Ceará (IFCE), por prover os meios para a minha participação no SBES'17.

A todos os professores do programa que foram tão importantes nessa fase da minha vida, mostrando que novos caminhos sempre podem ser trilhados.

Agradeço também a minha esposa, Pedrita Bessa, que de forma especial e carinhosa me apoiou e encorajou nos momentos de dificuldades.

Meus agradecimentos aos amigos que fiz durante esse mestrado, que fizeram parte da minha formação e me concederam bons momentos de descontração.

A todos que direta ou indiretamente fizeram parte da minha formação, o meu muito obrigado.

“O homem não é nada além daquilo que a educação faz dele.”
(IMMANUEL KANT)

Resumo

A demanda atual por apps tem movimentado sobremaneira o mercado de desenvolvimento de software. Por serem difundidos massivamente e utilizados por milhares de usuários diferentes é necessário que seja dada uma atenção especial ao projeto das interfaces e às formas de interação dos usuários com os apps. Uma forma eficiente de avaliar esses aspectos junto aos clientes antes de se iniciar o desenvolvimento de qualquer código é a prototipação. Um protótipo é um exemplo precoce ou modelo construído para testar conceitos. Contudo, a prototipação muitas vezes é ignorada devido ao custo adicional introduzido no processo de desenvolvimento dos apps e ao tempo que ele pode requerer. Nesse trabalho é apresentado um processo de prototipação para apps baseado na construção em papel de rascunhos de interfaces e de suas interações. Posteriormente esses rascunhos são capturados pela câmera de um smartphone, gerando uma versão do app que poderá ser utilizada pelo cliente e onde as interações poderão ser avaliadas. Esse processo também gera uma versão digital do projeto baseado em uma DSL (Domain Specific Language), pela qual esse protótipo poderá ser refinado, gerando, se necessário, uma nova versão executável para avaliação ou um novo modelo em papel para ser apresentado e modificado pelo cliente. Esse processo é validado nesse trabalho através de um estudo de caso, no qual uma aplicação é modelada em papel através de um estêncil para interface Android, gerando a versão digital do projeto assim como os códigos que implementam as telas e as estruturas de navegação entre elas.

Palavras-chave : Prototipação. Android. Desenvolvimento orientado a modelos. Reconhecimento de objetos.

Abstract

The current demand for apps has greatly moved the software development market. Due to their massive distribution and use by thousands of different users, special attention needs to be paid to the interfaces design and the ways of interaction with these apps. An efficient way to evaluate these aspects in advance is prototyping. However, prototyping is often overlooked because of the additional cost it has put into the app development process and the time that may require it. The aim of this paper is to present a prototyping process for apps based on the paper construction of interface drafts and their interactions. Later these drafts are captured by the camera of a smartphone, generating a version of the app that can be used by the client to evaluate the interactions. This process also generates a digital version of the project based on a Domain Specific Language (DSL), whereby this prototype can be refined, generating, if necessary, a new executable version for evaluation or a new paper model to be presented and modified by the client. The validation of the proposal is done through a case study, in which an application is modeled on paper through a stencil for Android interface. Thus, generating the digital version of the project as well as the codes that implement the screens and the navigation structures between them.

Keywords: Prototyping. Model-Driven Development. Android.

Lista de ilustrações

Figura 1 – Ciclo de Refinamento	27
Figura 2 – Processo de Design Lean UX	32
Figura 3 – Fases do Design Thinking UX	33
Figura 4 – Processo de Genérico de Design (PRESSMAN; MAXIM, 2016)	34
Figura 5 – Evolução de Protótipos	35
Figura 6 – Representação de Conceitos de Iniciativas Orientadas a Modelos (AMEL- LER, 2009)	44
Figura 7 – Arquitetura de Camadas da MOF	45
Figura 8 – Modelo de Geração de Código com Acceleo	48
Figura 9 – Sistema de visão computacional extraído de (GONZALEZ; WOODS, 2010)	52
Figura 10 – Processo de Casamento de Modelos (ARAÚJO, 2009)...	53
Figura 11 – Processo de Revisão Sistemática (BIOLCHINI et al., 2005).. . . .	55
Figura 12 – Processo MOP.	67
Figura 13 – Componentes essenciais.	69
Figura 14 – Estrutura básica de navegação.. . . .	70
Figura 15 – Barra superior (AppBar).	70
Figura 16 – Barra inferior de navegação (BottomBar).	71
Figura 17 – Tipos de botões do Material.	71
Figura 18 – Exemplo Cartão do Material.	72
Figura 19 – Exemplo de Caixa de Diálogo do Material.. . . .	72
Figura 20 – Exemplo de lista com controles de ações.	73
Figura 21 – Componentes de Entrada e Seleção.. . . .	74
Figura 22 – Estêncil Físico.	75
Figura 23 – Folha do caderno de layout.	75
Figura 24 – Recorte da lista simples.	76
Figura 25 – Recorte do Header.. . . .	76
Figura 26 – Corte do Rodapé	77
Figura 27 – Corte do CARD	77
Figura 28 – Corte do <i>DIALOG</i>	77
Figura 29 – Recorte da Lista	78
Figura 30 – Recorte dos Botões	78
Figura 31 – Recorte dos Campos de Entrada de Texto...	78
Figura 32 – Recorte dos Componentes de Seleção.	78
Figura 33 – Metamodelo Proposto.	81
Figura 34 – Geração do Código Fonte do Metamodelo.. . . .	82

Figura 35 – Criação de novo arquivo de modelo ..	82
Figura 36 – Ilustração do Sirius Specification Editor	83
Figura 37 – Estrutura de Projeto Cordova Padrão.	85
Figura 38 – Estrutura da Pasta www.	86
Figura 39 – Protótipo em papel.	89
Figura 40 – Refinamento.	91
Figura 41 – Resultado.	93

Lista de tabelas

Tabela 1 – Fontes de buscas e strings.	58
Tabela 2 – Seleção de artigos por repositório	59
Tabela 3 – Classificação dos artigos quanto a plataforma.	60
Tabela 4 – Classificação dos artigos quanto a arquitetura.	60
Tabela 5 – Principais ferramentas de prototipação. (CUELLO; VITTONI, 2013) .	64
Tabela 6 – Especificação do Modelo de Domínio.	103
Tabela 7 – Classe App	103
Tabela 8 – Classe Screen	104
Tabela 9 – Classe UIComponent	104
Tabela 10 – Classe SimpleComponent.	104
Tabela 11 – Classe GroupComponent	104
Tabela 12 – Classe Button	104
Tabela 13 – Classe FAB	104
Tabela 14 – Classe Flat	105
Tabela 15 – Classe Raised	105
Tabela 16 – Classe Content	105
Tabela 17 – Classe Label	105
Tabela 18 – Classe Radio	105
Tabela 19 – Classe Switch	105
Tabela 20 – Classe ProgressBar.	106
Tabela 21 – Classe Image	106
Tabela 22 – Classe GroupComponent	106
Tabela 23 – Classe Card	106
Tabela 24 – Classe Dialog	106
Tabela 25 – Classe List	106
Tabela 26 – Classe ListItem	107
Tabela 27 – Classe Bar	107
Tabela 28 – Classe AppBar	107
Tabela 29 – Classe ToolBar	107

Lista de abreviaturas e siglas

ACM DL	Association Computation Machinery Digital Library
APK	Android Package
CBIR	Content-Based Image Retrieval
CSS	Cascading Style Sheets
DSL	Domain-Specific Language
EMF	Eclipse Modeling Framework
Epsilon	Extensible Plataform of Integrated Languages for mOdel maNagement
ETL	Epsilon Transformation Language
GIMP	GNU Image Manipulation Program
GMF	Graphical Modeling Framework
HTML	HyperText Markup Language
ICS	Ice Cream Sandwich
IEEE	Instituto de Engenheiros Elétricistas e Eletrônicos
M2T	MODEL TO TEXT
MATLAB	MATrix LABoratory
MDD	MObel-Driven Development
MOF	Meta Object Facility
OCR	Optical Character Recognition
OMG	Object Management Group
POP	Prototyping on Paper
SIFT	Scale-Invariant Feature Transform
SURF	Speeded-UP Robust Features
UI	User Interface

UML	Unified Modeling Language
UX	User eXperience
XMI	XML Metadata Interchange
XML	eXtensible Markup Language

Sumário

1	INTRODUÇÃO	25
1.1	Motivação	25
1.2	Objetivos Geral e Específicos	28
1.3	Produção Científica	28
1.4	Estrutura da Dissertação	28
2	FUNDAMENTAÇÃO TEÓRICA	31
2.1	Prototipação	31
2.1.1	Classificação de Protótipos	34
2.1.2	Comparativo entre Técnicas Tradicionais e Ferramentas Digitais . . .	36
2.1.2.1	<i>Sketching</i>	37
2.1.2.2	<i>Whiteboard</i>	38
2.1.2.3	Protótipo em papel	38
2.1.2.4	<i>Stenciling</i>	39
2.1.2.5	Ferramentas de edição de texto.	39
2.1.2.6	Ferramentas de edição de imagens.	40
2.1.2.7	Software de apresentação.	41
2.1.2.8	Softwares de prototipação e <i>wireframing</i>	41
2.1.2.9	Software projetado para prototipação	42
2.1.3	Conclusão	42
2.2	Desenvolvimento orientado a modelos	43
2.2.1	Modelos e Metamodelos	44
2.2.2	Transformações de modelos	46
2.2.3	Geração de Código	46
2.2.4	Ferramentas	46
2.2.4.1	Acceleo	47
2.2.5	Conclusão	48
2.3	Visão Computacional e Casamento de Modelos	48
2.3.1	Visão computacional	49
2.3.1.1	Reconhecimento.	50
2.3.1.2	Análise de movimento.	51
2.3.1.3	Restauração de imagens	51
2.3.2	Sistema de visão computacional.	51
2.3.3	Casamento de Modelos (<i>Template Matching</i>)	52
2.4	Conclusão	53

3	TRABALHOS RELACIONADOS	55
3.1	Planejamento e condução da revisão	55
3.1.1	Objetivos da pesquisa.	55
3.1.2	Questão da pesquisa	55
3.2	Estratégia de busca e seleção de fontes	56
3.2.1	Critérios de seleção	56
3.2.2	Processo de seleção preliminar.	56
3.2.3	Processo de seleção final.	57
3.2.4	Extração de informações	57
3.2.5	Condução da pesquisa	57
3.3	Resultados	59
3.3.1	Plataformas.	59
3.3.2	Arquitetura	59
3.3.3	Ferramentas e técnicas.	60
3.4	Conclusão	65
4	O PROCESSO MOP	67
4.1	Etapas	67
4.2	O Estêncil	68
4.2.1	Principais Componentes do Material.	68
4.2.2	Especificação do Estêncil.	74
4.3	App de Reconhecimento e Servidor Scilab	79
4.4	Editor Gráfico de Modelos	80
4.4.1	Projeto de Modelagem	80
4.4.2	Projeto de Especificação de Ponto de Vista.	82
4.4.3	Refinamento	84
4.5	Simulação no Apache Cordova	84
4.5.1	Estrutura de pastas do projeto	85
4.5.2	Explicando a interatividade	86
4.5.3	Simulação ou Compilação	87
4.6	Conclusão	87
5	ESTUDO DE CASO	89
5.1	Desenhe	89
5.2	Digitalize	89
5.3	Refine	90
5.4	Publique e teste	92
5.5	Conclusão	92
6	CONCLUSÃO	95

REFERÊNCIAS	97
--------------------	-----------

APÊNDICES	101
------------------	------------

APÊNDICE A – ESPECIFICAÇÃO DO METAMODELO MOP V1.0103	
---	--

APÊNDICE B – CÓDIGO RESULTANTE DO PROTÓTIPO .	. . . 109
--	------------------

1 Introdução

Aplicativos móveis, ou simplesmente apps, vêm se tornando cada vez mais sofisticados e mais presentes no cotidiano das pessoas. Aliado à rápida evolução de dispositivos como smartphones e tablets, esses apps têm oferecido funcionalidades cada vez mais complexas.

Por serem difundidos massivamente e utilizados por milhares de usuários diferentes é necessário que seja dada uma atenção especial ao projeto das interfaces e às formas de interação dos usuários com os apps. Uma forma eficiente de avaliar esses aspectos junto aos clientes antes de se iniciar o desenvolvimento de qualquer código é a prototipação.

Um protótipo é um exemplo precoce ou modelo construído para testar um conceito. A etapa de prototipação por muitas vezes é ignorada devido ao custo adicional que acarreta no processo de desenvolvimento do software (PRESSMAN; MAXIM, 2016). No entanto é, amplamente aceito no mundo do desenvolvimento. A retirada dessa etapa pode levar a falta de comunicação entre os envolvidos, problemas com o orçamento e falta de qualidade no produto. A prototipação oferece a oportunidade de realizar ensaios para diversos tipos de usuários, antes mesmo de levar o projeto para a fase de codificação.

Devido à simplicidade e ao baixo custo, a prototipação em papel vem sendo utilizada já a bastante tempo em fases iniciais de projetos de software (SNYDER, 2003). Nessa abordagem, a equipe esboça em papel as telas e as interações realizadas pelos usuários com o objetivo de discutir alguns aspectos mais gerais das aplicações. Também utilizados em testes de usabilidade, os protótipos em papel servem para simular a utilização da aplicação, na qual os usuários podem tocar ou clicar em elementos da interface em papel de modo que essas interações possam ser avaliadas.

A técnica de prototipação em papel já é adotada atualmente para o desenvolvimento de apps, como apresentado em (Marvel, 2017). Contudo, os protótipos são utilizados apenas para avaliar as interfaces, sendo descartados posteriormente. Dessa forma, os resultados da prototipação não produzem artefatos que sejam diretamente incorporados ao processo de desenvolvimento do app.

1.1 Motivação

Pode se observar ao longo dos últimos anos a grande popularização das aplicações móveis, normalmente denominadas apenas de aplicativos ou *apps*. No mesmo sentido dessa popularização, cresce a demanda pelo desenvolvimento dos mais diversos tipos de aplicativos, dos mais variados escopos e nas mais variadas áreas.

Nesse cenário, os desenvolvedores tem encontrado usuários com grande habilidade em utilizar esses aplicativos e, por conta disso, com maiores exigências com relação ao que os aplicativos devem fornecer. Assim, esses desenvolvedores muitas vezes tem o desafio de se comunicar com um cliente que já tem familiaridade com a tecnologia, mas não tem conhecimento técnico para desenvolver seus próprios produtos.

Muitas vezes as primeiras ideias desses aplicativo são iniciadas em rascunhos das telas e das estruturas de navegação relacionadas às funcionalidades requisitadas. Esses rascunhos tanto podem ser feitos pelos desenvolvedores como também pelos próprios usuários e, muitas vezes acabam sendo realizadas em rabiscos em papel, pelo fato de tornar a prototipação mais rápida e pela mínima exigência de recursos para a realização dessa prototipação.

Protótipos de alta fidelidade não são geralmente utilizados no início do ciclo de vida de um produto, pois pode ser difícil e lento a produzir. Os designers que vem tentando construir esses protótipos podem precisar da ajuda de engenheiros de software e desenvolvedores na equipe. Além de levar tempo e esforço considerável para criar, implementar qualquer alteração sobre este tipo de protótipos também pode ser demorado.

Para produzir protótipos que são de baixo custo, fácil e rápido para recriar, os designers podem recorrer a técnicas de prototipação de baixa fidelidade como o protótipo de papel (UXPIN, 2015). Esta técnica envolve a criação de desenhos de interfaces que podem ser utilizadas como modelos de um design em testes de usabilidade. Prototipação em papel (sketch ou stencil) é prático e útil, pois permite a entrada de usuários finais nas fases iniciais do ciclo de design. (UXPIN, 2015).

Ao contrário de protótipos de alta fidelidade, protótipos em papel pecam nos testes de usabilidade e na precisão do design. Por exemplo, um protótipo em papel de um site só pode ser experimentado no pedaço de papel em que ele foi criado, ao passo que um protótipo baseado em HTML e CSS pode ser experimentado em um navegador. Protótipos em papel também dependem de um facilitador para agir "como o computador", mudando diferentes estados da interface do usuário em resposta a ações de um participante (Bolchini, 2009).

A falta de contexto físico de protótipos de papel é particularmente problemático para projetar experiências móveis. Como os *apps* estão dentro dos dispositivos móveis, podem ser experimentado em muitas posições possíveis e em diversos contextos. Mostrando protótipos em um ambiente incomum (como um pedaço de papel sobre uma mesa) não dá aos usuários uma experiência completa e precisa do design. Protótipos em papel simples podem testar interfaces de usuário e conteúdo de um produto, mas deixam muito a desejar.

Se fosse possível replicar mais da experiência do usuário móvel a partir de protótipos de baixa fidelidade, seria mais proveitoso do que esperar produzir protótipos de alta

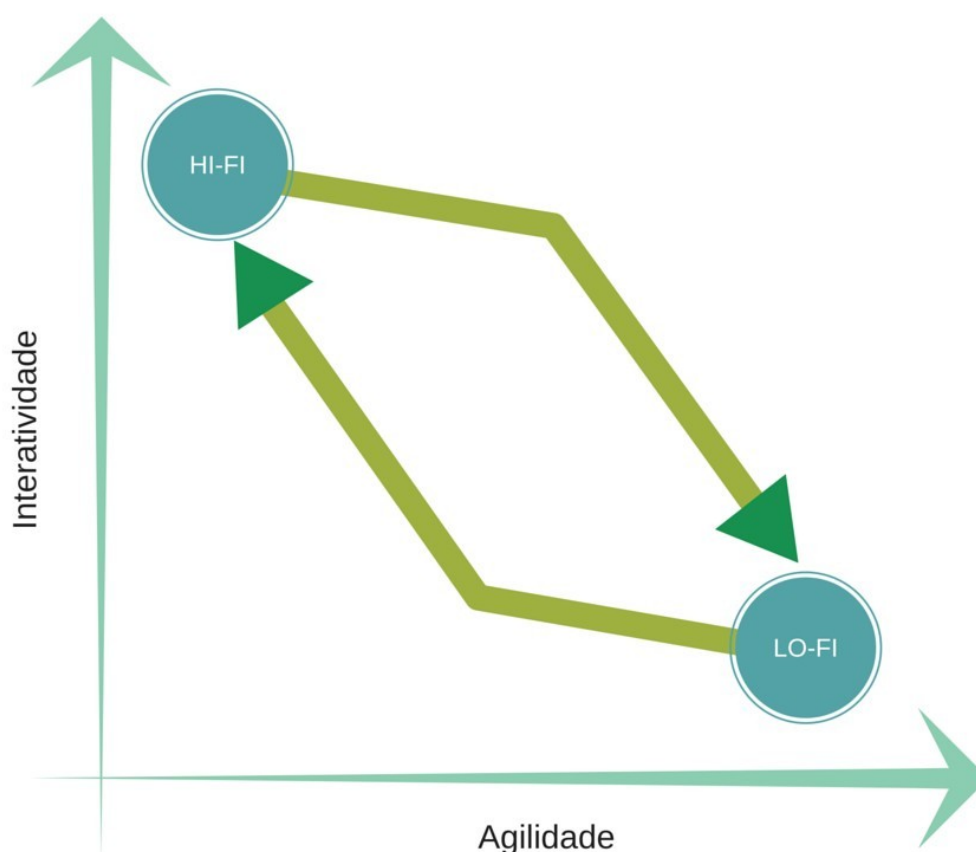


Figura 1 – Ciclo de Refinamento

fidelidade no decorrer do processo de design de interface. Esta pesquisa procura desenvolver um processo de prototipação, que visa minimizar as desvantagens da prototipação de baixa e alta fidelidade. O processo tem como objetivo aterrar o fosso entre a baixa e a alta fidelidade diminuindo o tempo de fabricação de um protótipo de alta fidelidade, bem como fornecendo interatividade ao protótipo de baixa fidelidade. A metodologia ainda promove um ciclo de refinamento e alternâncias entre o baixo e alto nível, facilitando ainda mais a percepção do conceito da interface para vários participantes do seu desenvolvimento. A Figura 1 apresenta um gráfico baseado no conceito de refinamento de protótipo.

Nesse sentido esse trabalho apresenta um processo de prototipação de *apps* no qual os protótipos em papel geram artefatos diretamente utilizados no desenvolvimento da versão final do produto. Denominado MOP (*Models On Paper*), esse processo utiliza uma abordagem orientada a modelos no qual os protótipos em papel são mapeados em modelos que podem ser manipulados pelos desenvolvedores em um projeto, em formato digital.

A partir desses modelos, é gerada uma versão executável da aplicação para ser utilizada diretamente no *smartphone*, possibilitando uma simulação digital do protótipo em papel. Além disso, é possível realizar ajustes nos elementos descritos no papel, gerando uma versão atualizada para impressão que pode ser novamente manipulada pela equipe,

em um processo cíclico de refinamento do produto.

1.2 Objetivos Gerais e Específicos

O objetivo principal dessa pesquisa é a definição de um processo baseado em uma Linguagem de Domínio Específico - LDE para a prototipação de interfaces de aplicações móveis. Essa LDE se caracteriza por fornecer mecanismos para que os projetos possam ser desenhados em papel e capturados posteriormente através da câmera digital do *smartphone* e reconhecidos por uma ferramenta que irá gerar tanto a versão digital do projeto como os códigos que implementam as telas e as estruturas de navegação entre elas.

Em seu sentido mais particular, os objetivos específicos são:

- Realizar uma revisão sistemática da literatura como foco nos processos de prototipação para dispositivos móveis;
- Propor uma técnica para geração de protótipos de interface de usuários que envolve prototipação de alta e baixa fidelidade;
- Criar um estêncil em acetato ou acrílico com os elementos de layout para o esboço dos protótipos para interfaces Android;
- Desenvolver um modelo conceitual que permita a criação de protótipos para aplicações móveis;
- Implementar as ferramentas necessárias para utilização do modelo conceitual;
- Um Estudo de Caso visando a prova de conceito do Processo de Prototipação MOP.

1.3 Produção Científica

Durante o desenvolvimento desta dissertação, o seguinte artigo científico foi publicado:

Abner Augusto Lima de Oliveira and Cidcley Teixeira de Souza. 2017. Paper Prototyping in a Model-Driven Process for Android Application Simulation Support. In Proceedings of SBES'17, Fortaleza, CE, Brazil, September 20–22, 2017. DOI: <https://doi.org/10.1145/3131151.3131151>

1.4 Estrutura da Dissertação

Além desse capítulo, a dissertação está organizada da seguinte forma:

- **Capítulo 2 - Fundamentação Teórica:** demonstra fundamentos e técnicas de prototipação de interfaces, descreve os principais conceitos de Desenvolvimento Orientado a Modelos e as Linguagens de Domínio Específico que também é um

dos aspectos mais relevantes para a dissertação, e, expõe os conceitos de visão computacional utilizados no desenvolvimento do MOP;

- **Capítulo 3 - Trabalhos Relacionados:** nesse capítulo é apresentada uma revisão sistemática de literatura que contém as principais técnicas de prototipação de interface de usuário para dispositivos móveis, bem como, as ferramentas digitais para prototipação em destaque atualmente;
- **Capítulo 4 - O Processo MOP:** nesse capítulo é apresentada a definição do processo, cenários de utilização e as especificação das ferramentas que compõem o processo.
- **Capítulo 5 - Estudos de Caso:** aborda o capítulo de avaliação, expondo um estudo de caso, demonstrando os passos da utilização do processo para a criação de protótipo de aplicativo de lembretes;
- **Capítulo 6 - Conclusão:** no último capítulo são tiradas as conclusões, lições aprendidas e a descrição de possíveis trabalhos futuros.

2 Fundamentação Teórica

Este capítulo apresenta os principais conceitos envolvidos neste trabalho e está dividido em três seções. A primeira apresenta conceitos de prototipação, vantagens e desvantagens na sua utilização, classificação de protótipos, comparativos entre técnicas de prototipação de interfaces móveis. A segunda seção traz conceitos sobre Desenvolvimento Orientados a Modelos, Linguagens de Domínio Específico e as principais ferramentas disponíveis atualmente. A última parte elenca conceitos de Reconhecimento de Objetos, qual a organização de um sistema de Visão Computacional, Descritores e técnicas de Casamento de Modelos.

2.1 Prototipação

Ao se iniciar a ideia para um site ou aplicativo é comum se iniciar o esboço das telas e interações entre elas, mas a parte dura é compreender como essa ideia tomará forma e que tenha uma boa aceitação pelo usuário. E é aqui que o esboço é útil. Não importa o quão inteligente seja o projetista, é impossível ir de uma ideia simples para a entrega do projeto sem passar pelas fases de levantamento de requisitos, definição de aspectos visíveis, construção e avaliação do protótipo.

Projetar a interface do usuário é um processo. O processo de ideação inicia-se com a ideia, mas essa ideia precisa ser traduzida em uma interface para o usuário. Não é suficiente dizer, "Eu quero um aplicativo que faz X, Y e Z". É necessário saber o que o usuário verá em cada tela de seu aplicativo para fazer X, Y e Z. Esboçar eleva o nível de criatividade e força o *designer* a pensar sobre a experiência do usuário. Esboçar permite que o *designer* visualize a interação tela a tela e que sua ideia é algo que é visível e claro na interface do usuário (LIN et al., 2000).

Na Engenharia de Software, esse processo é parte integrante de um processo de desenvolvimento de software, conhecido como Projeto ou *Design* (PRESSMAN; MAXIM, 2016). O processo de Design é responsável por todo o planejamento antes de se iniciar o desenvolvimento, nele é definido a arquitetura do software, design da interface de usuário, validação de requisitos, estrutura de dados dentre vários outros (SOMMERVILLE et al., 2003). Toda a especificação feita durante o processo de design tem por objetivo a construção de um software com qualidade. Existem diversos modelos de projeto de software, sejam eles tradicionais como o proposto por PRESSMAN (PRESSMAN; MAXIM, 2016) e SOMMERVILLE (SOMMERVILLE et al., 2003) ou processo mais inovadores como o Design de Interação, *Lean UX* (GOTHELF, 2013) e *Design Thinking* (BROWN et al., 2010).

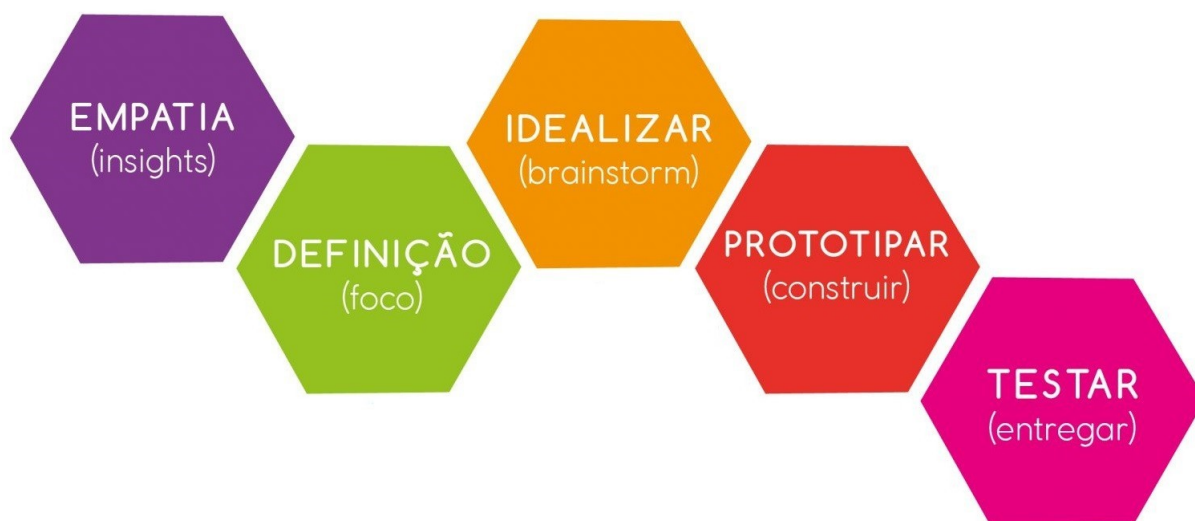


Figura 3 – Fases do Design Thinking UX

são estabelecidos todos os elementos que permitirão a solução do problema de uma forma centrada no usuário.

Durante a terceira fase do Design Thinking inicia-se a construção de ideias. São realizados *brainstorms* para levantar o máximo de soluções para o problema, depois são escolhidos as melhores formas de contornar o problema. Na etapa de prototipação são produzidos uma série de versões de baixo custo do software, com intuito de investigar a eficiência das soluções propostas durante a terceira fase. Protótipos podem ser compartilhados e testados dentro da própria equipe, em outros departamentos, ou em um pequeno grupo de pessoas de fora da equipe de design. Nessa fase o objetivo principal é identificar a melhor solução possível para cada um dos problemas apontados nas fases anteriores. A fase final é a de testes, onde os projetistas avaliam completamente o protótipo final. Mesmo durante esta fase, alterações e aperfeiçoamentos são realizados a fim de descartar soluções impróprias.

De acordo com os parágrafos anteriores pôde-se observar que a prototipação é parte integrante do processo de Projeto, seja em metodologias atuais ou mais tradicionais. Mesmo que a prototipação possa ser utilizada como um Processo de Desenvolvimento de Software é mais comum vê-la na fase de Projeto, onde auxilia os envolvidos a compreender melhor o software a ser construído.

O protótipo é um modelo temporário do software a ser desenvolvido, que representa uma solução de baixo custo, onde os usuários podem interagir e tecer críticas sobre a real experiência de seu uso (PRESSMAN; MAXIM, 2016). A prototipação é uma parte crucial de processos iterativos de design, no *Design Thinking* e em processos de design centrado no usuário (BROWN et al., 2010). O desenvolvimento de protótipos permitem refinar os

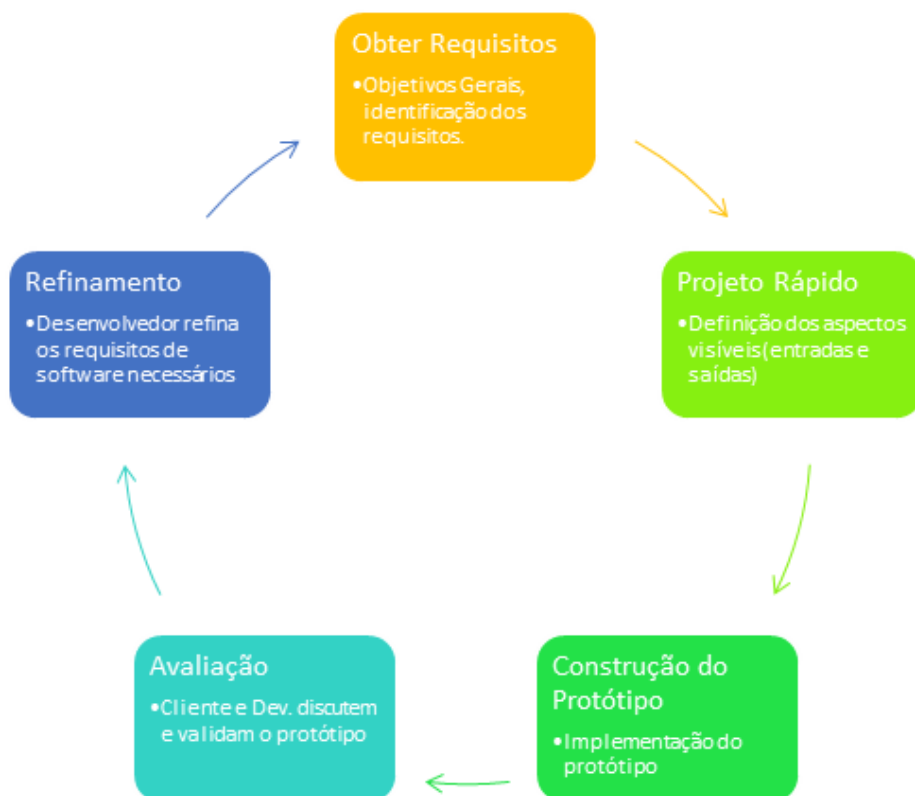


Figura 4 – Processo de Genérico de Design (PRESSMAN; MAXIM, 2016)

requisitos de forma assertiva, avaliar interativamente as ideias, validar as soluções junto ao usuário final e antecipar empecilhos e problemas, reduzindo riscos e otimizando custos (SOMMERVILLE et al., 2003).

Na engenharia de software o paradigma da prototipação é definido em quatro estágios: Comunicação; Projeto Rápido; Construção e Teste. A Figura 4 ilustra o processo, que se inicia com a comunicação, onde é feita uma reunião para definição dos requisitos do software e quais áreas precisam de mais atenção, no projeto rápido é feita uma modelagem básica, uma interface representando os principais aspectos do software a serem exibidos na tela. O projeto rápido direciona a criação de um protótipo, que será analisado e de acordo com o resultado servirá para o aprimoramento dos requisitos (PRESSMAN; MAXIM, 2016; SOMMERVILLE et al., 2003).

2.1.1 Classificação de Protótipos

Os protótipos podem ser utilizados em diferentes fases do projeto de um software e, podendo assumir funções diferentes de acordo com a etapa do projeto e o objetivo da avaliação. Pode-se classificar os protótipos em duas principais categorias, de acordo com as características a seguir.

Um protótipo pode ser classificado quanto ao nível de fidelidade, ou seja, o quanto se assemelha ao software final, levando em consideração suas funcionalidades, interatividade

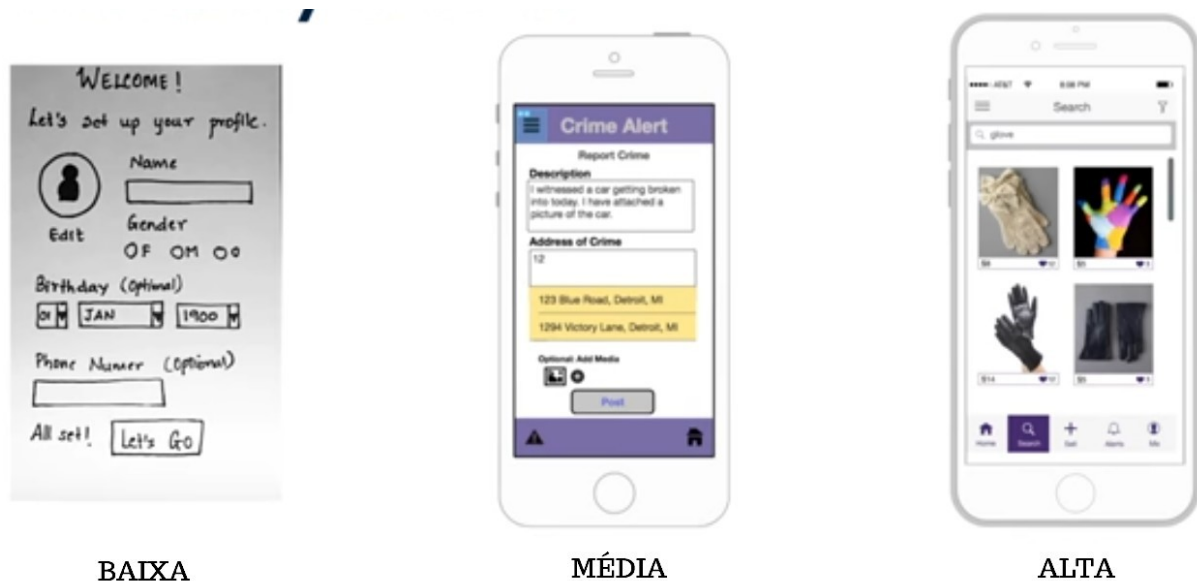


Figura 5 – Evolução de Protótipos

e a interface. São classificados em baixa, média e alta fidelidade (GREMILLION; ZIEBA, 2016).

Protótipos de baixa fidelidade são aqueles gerados rapidamente, de forma barata que servem basicamente para ilustrar um conceito da interface. Permite o teste de diversas ideias, ajudam a validar requisitos e regras de negócio de maneira eficiente (WARFEL, 2009).

Os protótipos de média fidelidade são gerados rapidamente, e diferentemente dos protótipos de baixa fidelidade utilizam ferramentas de prototipação digital. Protótipos de média fidelidade oferecem interfaces com poucos recursos, mas ainda sim são funcionais. A partir de um protótipo de média fidelidade é possível passar mais rapidamente de uma forma conceitual para a implementação do software final (WARFEL, 2009).

Os protótipos de alta fidelidade oferecem um interface bem mais próxima do resultado final. São ideais para testar interações complexas e soluções para usabilidade, pois fornecem ao usuário uma experiência realista. Esses protótipos são produzidos com os mesmos materiais da versão final do software e demandam mais conhecimento técnico e mais tempo para serem desenvolvidos (GREMILLION; ZIEBA, 2016).

De acordo com nível de fidelidade do protótipo sua nomenclatura pode ser diferente. Os protótipos de baixa fidelidade são conhecidos como *sketches* (esboços), geralmente feitos a mão livre ou com uso de técnicas de *Sketching*, *Stenciling* ou *Whiteboard*. Os *wireframes* apresentam uma estrutura mais refinada, mas geralmente sem cores, os *mockups* exibem conteúdo com cores, os protótipos tem seu *layout* bem próximo da versão final e possuem interatividade. A figura 5 apresenta exemplos de protótipos de baixa, média e alta fidelidade.

Existem vários modelos de prototipação (PRESSMAN; MAXIM, 2016), os quais podem ser classificados quanto a sua reutilização. Destacam-se os modelos de Prototipação Descartável e de Prototipação Evolucionária. A Prototipação Descartável apresenta um protótipo que é usado apenas para a avaliação, sendo descartado após a compreensão dos requisitos. O sistema final é desenvolvido utilizando-se um outro processo de desenvolvimento de software. A Prototipação Evolucionária é uma técnica de desenvolvimento que realiza entregas de protótipos, que são refinados a cada ciclo iterativo.

2.1.2 Comparativo entre Técnicas Tradicionais e Ferramentas Digitais

Essa seção apresenta um comparativo entre as principais técnicas de prototipação de interface para aplicações móveis, vantagens e desvantagens de cada técnica, bem como as visões da prototipação de acordo com os profissionais.

O propósito de qualquer *wireframe* varia de acordo com as pessoas que criam ou utilizam. Seja feito em *sketch*, *grey boxes*, *wireframes*, *mockups* ou em ferramentas digitais, a mesma informação pode ser repassada. Portanto é muito menos sobre a ferramenta e mais sobre a rapidez que se pode transmitir a informação a quem necessita. A seguir, discute-se a visão de cada ator no cenário do processo de Design de Interface.

Designer de *User Experience* - UX e Arquitetos de informação: utilizam *wireframes* para mostrar ao usuário como flui a visualização das páginas, bem como detalhes sobre a estrutura, funcionalidades, comportamento e conteúdos e hierarquia das informações. Fluxogramas, *wireframes* e *storyboards* são utilizados para alcançar esse objetivo.

Designers gráfico:criam *wireframes*, *mockups*, protótipos interativos que serão entregues ao Desenvolvedor. Geralmente combinam *sketching*, *storyboarding*, *wireframes lo-fi* e *hi-fi* para alcançar o resultado desejado.

Desenvolvedores: usam o *wireframe* para ter uma noção maior dos elementos e das funcionalidades do *site*, *app* ou outro software.

Analista de negócios:utilizam *wireframes* para apoiar visualmente as regras de negócio e requisitos de interação para uma tela. Por exemplo, um analista publicidade ou conteúdo provavelmente se preocupa mais com o conteúdo, funcionalidade e comportamento enquanto que uma regulamentação analista pode se preocupam mais com hierarquia ou estrutura de informação.

Gerentes de Projetos e de Produtos: utilizam os *wireframes* para garantir que os requisitos e objetivos do projeto estejam sendo atingidos.

Sócios e Clientes: também utilizam os *wireframes* para garantir que os requisitos estejam sendo atingidos, porém sua principal preocupação é seu próprio negócio. Portanto eles desejam algo limpo e legível que resolva seu problema rapidamente.

A seguir segue uma lista com as principais técnicas de prototipação não-digital e suas vantagens e desvantagens.

2.1.2.1 Sketching

Há muitas maneiras de se criar um conceito de interface. O *sketch*, é um esboço e serve para ilustrar um conceito de interface de modo mais básico. O que mais importa é entender o que o usuário vai ver, basicamente, em cada tela que vai levá-los para o resultado desejado.

Vantagens:

- **Rapidez:** é realmente rápido pegar um pedaço de papel e começar a desenhar as ideias do produto. Não é necessário pesquisar elementos, memorizar atalhos de teclados, alterar cursor de ferramenta ou qualquer outro empecilho que venha a atrasar a sua ideia;
- **Flexibilidade:** o conceito pode ser mudado para alto ou baixo nível com facilidade, o refinamento é feito na ponta do lápis. Aplicativos de prototipagem são mais engessados, definidos para uma estrutura pré-definida;
- **Foco:** nessa técnica pode-se focar na estrutura, conteúdo, funcionalidade e comportamento do produto. Não é necessário se preocupar com tipos de letras, cores e estilos em geral. À medida que o *wireframe* é refinado são incluídas essas particularidades.

Desvantagens:

- **Retrabalho:** não existem modelos ou elementos de reuso, por mais rápido que seja editado o trabalho ele se tornará tedioso. Com o esboço finalizado não há opção de exportar esse trabalho para outra aplicação e reiniciar de onde foi parado. Todo o árduo trabalho tem que ser transcrito para uma ferramenta de edição gráfica ou de prototipação digital, como Balsamiq ou Axure;
- **Baixa-Fidelidade:** a falta de elementos de estilos pode tornar o trabalho mais focado, porém baixa a fidelidade do protótipo, é muito mais difícil mostrar as iterações, animações e as cores;
- **Sem interações:** há limitações para demonstrar interações com *sketch* não é possível vincular desenhos juntos. Ferramentas digitais fornecem um auxílio maior nesse quesito;
- **Sem controle de versão:** esta é uma questão inerente às ferramentas não-digitais, não tem como ter controle das alterações feitas;
- **Colaboração limitada:** não há como todos fazerem alterações no mesmo esboço ao mesmo tempo. As validações das alterações são feitas mediante reuniões. Antes ou depois disso as poucas *stakeholders* tem acesso ao *sketch*;
- **Sem padrão:** Elementos podem ser inconsistentes, se não houver cuidado, as

proporções e o dimensionamento podem ficar completamente fora da realidade, o que poderia impactar diversas informações transmitida em esboços. *Stencils*, *wireframes*, *mockups*, e software de apresentação fornecer uma melhor solução para este problema.

2.1.2.2 Whiteboard

Tem as mesmas características do *sketching*, só que é feito em uma quadro branco. Usado geral como meio de troca e organização de ideias em ambiente corporativo. Algumas ferramentas digitais de prototipagem já fornecem o conceito de *whiteboard*, visando estimular a comunicação da equipe de design. Possui e as mesmas vantagens e desvantagens do *Sketching* e com algumas características adicionais.

Vantagens:

- **Escala:** o tamanho permite trabalhar o esboço de forma mais detalhada;
- **Destaque:** em escritórios geralmente as paredes são pintadas de branco ou adornadas com quadros brancos ou vidros, com notas e diagramas espalhado por toda parte. Isso favorece a um ambiente colaborativo, pois qualquer pessoa pode entender e opinar sobre o *layout* em questão. Devido a essa natureza colaborativa essa ferramenta é bastante utilizada como ferramenta de prototipação.

Desvantagens:

- **Dificuldade no desenho:** como é fixo na parede se torna difícil esboçar;
- **Detalhamento limitado:** o nível de detalhamento se torna limitado pois a ferramentas utilizadas para desenhar são de difícil manuseio. Mesmo assim existem pessoas capacitadas a fazer um bom esboço no *Whiteboard*. O detalhamento de uma ferramenta digital é infinitamente maior.

2.1.2.3 Protótipo em papel

É bem semelhante às técnicas discutidas anteriormente, o diferencial é que os elementos de *design* são cortados e montados para formar as telas. Existem pacotes padronizados que aceleram o processo de prototipação em papel. Protótipos em papel economizam tempo e dinheiro, uma vez que permitem aos desenvolvedores testar as interfaces do produto.

Vantagens:

- **Fidelidade:** o nível de fidelidade com o produto final fica a critério do *designer* ;
- **Padronização:** são diversos elementos recortados com a mesma forma, criando uma estrutura padronizada;
- **Interatividade:** é possível fixá-los na parede, na tela do celular, mover e reorganizar os elementos adicionando interatividade aos recortes facilitando a

compreensão dos usuários.

Desvantagens:

Essa técnica tem as mesmas desvantagens do *sketching* e do *whiteboard*. Aida apresenta a demora na produção do protótipo, pois os recortes e sua organização acabam levando mais tempo para o seu desenvolvimento.

2.1.2.4 Stenciling

É semelhante ao *sketching* com o diferencial de existir a padronização da estrutura. Existem diversos kits de estêncil para interface de usuários que permitem a geração de ideias de aplicativos com facilidade, que auxiliam no esboço dos conceitos de UI, UX, interações e *wireframes*. Cada estêncil utiliza o padrão para qual foi projetado, por exemplo, interface do Android Lollipop.

Vantagens:

- **Rapidez:** com o estêncil é possível criar interfaces detalhadas rapidamente;
- **Padronização:** com a padronização se torna mais fácil o entendimento da interface;
- **Estrutura:** esse tipo de ferramenta fornece um melhor alinhamento dos elementos e do conteúdo facilitando a compreensão das características do produto;
- **Design limpo:** como o designer utiliza uma ferramenta e não a mão livre a aparência da interface fica mais limpa.

Desvantagens:

- **Flexibilidade:** quando se começa a utilizar o estêncil perde-se um pouco da autonomia entre baixar ou elevar o nível do seu esboço, devido a padronização dos elementos;
- **Elementos desatualizados:** os estilos são atualizados constantemente o que leva ao envelhecimento da ferramenta. Como exemplo o Android mudou completamente suas regras de design da versão para 4 (ICS) para a 5 (Lollipop);
- **Foco na ferramenta:** com a preocupação com o uso da ferramenta perde-se um pouco do foco na estrutura, conteúdo e comportamento.

Foram demonstradas as vantagens e desvantagens das principais técnicas de prototipagem em meios não digitais. A seguir, são demonstradas as características das principais técnicas de prototipação digital.

2.1.2.5 Ferramentas de edição de texto

Todos já tiveram contato com as ferramentas de edição de texto como: Google Docs, Writer ou Microsoft Word dentre outras. São ferramentas comuns e de fácil acesso

online. Atualmente estão disponíveis versões de editores online onde se pode trabalhar em grupo, ou seja, os documentos podem ser compartilhados, várias pessoas podem editar o mesmo documento ou apenas acompanhar o desenvolvimento do projeto.

O principal ponto positivo é que todos tem familiaridade com esse modelo de ferramenta, todos sabem que os processadores de texto contem elementos básicos de *wireframes* como: formas, cores e texto.

2.1.2.6 Ferramentas de edição de imagens

Existem muitos aplicativos, digitais, gratuitos e pagos, que podem ser utilizados, porém a grande maioria dos designers utilizam o Adobe Illustrator ou Photoshop para criar *wireframes*. Ainda existem ferramentas populares como o GIMP, Pixlr dentre outros editores de imagens bastante populares no mercado.

Vantagens:

- **Familiaridade:** como a maioria dos designers utilizam essa ferramentas, eles se adaptaram a esse tipo de fluxo de trabalho. Estão acostumados com os modelos de interface pre fabricados,atalhos de teclado.Contudo existem ferramentas mais específicas para a criação de interfaces que agilizam o trabalho e podem trazer recursos mais avançados como interações e animações;
- **Biblioteca de elementos:** Os kits da web e design de aplicativos móveis que existem em si são todos muito fáceis de usar e vão se provar realmente útil quando o designer tiver que montar um layout rapidamente, para o cliente ou sua equipe de desenvolvimento,ou mesmo por apenas capturando uma ideia *layout*. Tudo que é necessário é manobrar os elementos pré-construídos de cada biblioteca no lugar, editar levemente conforme necessário, e será criado em um tempo muito curto um *wireframe* de baixa fidelidade do seu projeto;
- **Fidelidade:** com a capacidade de usar qualquer forma,texto , cor, itens da biblioteca de modelos é possível literalmente qualquer *wireframe*. É possível criar em qualquer tipo de fidelidade, exceção de interações e animações.

Desvantagens:

- **Difícil aprendizado:** essas ferramentas são tão flexíveis, isto é, não é focada em *wireframes*, que pode tornar o aprendizado mais demorado. É necessário um usuário avançado para que um *wireframe* ou protótipo seja feito com rapidez.
- **Bibliotecas:** o avanço é rápido e apesar da vasta gama de modelos na internet pode ser complicado achar os elementos certo e produzir seus próprios elementos demandam tempo.
- **Colaboração limitada:** poucas ferramentas de edição de imagens tem um recursos de trabalho simultâneo. Esse recurso de interatividade dos colaboradores

é bastante comum em aplicativos como o Balsamiq, Proto.io, Axure dentre vários outros.

- **Sem interatividade:** não existe interatividade entre as telas, não existe uma opção de visualizá-los em série ou ligar *wireframes* e *mockups*. Ferramentas como o Flinto tem a proposta de resolver esse problema.

2.1.2.7 Software de apresentação

Os softwares de apresentação são tão conhecidos quanto os de edição de texto, a maioria das pessoas já utilizaram em alguma fase da sua vida. É uma opção básica para a criação e apresentação de *wireframe*. Existem diversas mas as predominantes são o PowerPoint e o Keynote, porém existem boas ferramentas como o Prezi.

Vantagens:

- **Familiaridade:** Tal como acontece com software de processamento de texto, a maioria sabe o básico. E os seus recursos realmente são fáceis de aprender. Contém recursos avançados como animações, transições de slides e assim por diante;
- **Bibliotecas de elementos:** o Keynotopia pode rapidamente se tornar uma ferramenta de básica de *wireframe* utilizando dos elementos das bibliotecas além das formas básicas, cores, textos, animações, hiperlinks;
- **Experiência do usuário:** pela natureza da ferramenta o designer é forçado a pensar na experiência real do usuário e não somente de somente telas separadas. Para usuários mais avançados podem criar links entre os slides de apresentação de formas mais complexas que não acompanham necessariamente a progressão linear da apresentação;
- **Interatividade:** a interatividade apresentada por esse tipo de ferramenta foi demonstrada no item anterior.

Desvantagens:

- **Bibliotecas de elementos:** ocorre a mesma problemática do editores de imagem;
- **Colaboração limitada:** poucas ferramentas disponibilizam o recurso de vários designers trabalharem na mesma tela ao mesmo tempo. Nos aplicativos de prototipação já ocorre com mais frequência.

2.1.2.8 Softwares de prototipação e *wireframing*

A beleza de software de prototipação de aplicativos é que ele é projetado especificamente para esta finalidade, para que eles proporcionar o equilíbrio perfeito entre a funcionalidade, a curva de aprendizado, e facilidade de uso. Ambos os iniciantes e

veteranos designers usam ferramentas especializadas. Para iniciantes a facilidade de uso, e para veteranos os controles criados para suas necessidades particulares. Essas ferramentas variam em suas capacidades, sendo alguns mais afinados a determinadas situações do que outros, por isso é melhor para encontrar o caminho mais adequado para suas necessidades. Existem vários aplicativos dentre as eles estão: UXPin, Invision, MockFlow, Justinmind, Axure, OmniGraffle, JustProto, Flinto, ou Marvel.

2.1.2.9 Software projetado para prototipação

Vantagens:

- **Rapidez:** como eles são projetados especificamente para *wireframing*, as ferramentas mais rápidas para a criação, colaboração e para apresentação de wireframes. Pode ser mais rápido do que estêncil ou *sketch* pois existem características de ambos, porém de forma digital. Contudo é necessário observar se não está sendo deixando o lado detalhes na estrutura, conteúdo, funcionalidade e comportamento do produto;
- **Bibliotecas de elementos:** ferramentas como Justinmind, JustProto, e UXPin vêm com construído e atualizando constantemente bibliotecas de elementos ainda permitem que o usuário crie seus próprios elementos;
- **Interações avançadas:** com a atualização dos sistemas operacionais móveis aparecem novas formas de interações. Essas ferramentas estão em constante atualização;
- **Colaboração:** a maioria dessas ferramentas possuem a capacidade de várias pessoas acessarem e editarem o mesmo protótipo simultaneamente, possui histórico de versão, gestão de tarefas e armazenamento em nuvem;
- **Apresentação:** permitem exportar em PDF, software de apresentação, simulador Web ou de interface em dispositivos móveis.

Desvantagens:

- **Tempo de aprendizado:** pode exigir alguns dias para assimilar o uso das ferramentas;
- **Limitação de funcionalidade:** com exceção de algumas a maioria das ferramentas, geram protótipos de baixa fidelidade e funcionalidade reduzida. Contudo já existem algumas ferramentas que já estão integradas com editores de imagens como Photoshop, Illustrator ou Sketch.

2.1.3 Conclusão

Para escolher a melhor técnica de prototipação de interface é preciso levar em consideração as especificidades tanto do produto quanto da equipe. Quais plataformas os

designers e desenvolvedores estão familiarizados. Quais os objetivos específicos, prazo e limitações do projeto de design de interface de usuário. A prototipação é importante pois é um meio rápido e eficiente de validar uma ideia ou um novo conceito. Existem várias formas de se iniciar um projeto de software, contudo a prototipação é primordial para o seu sucesso, pois reduz o tempo e o custo de desenvolvimento. De acordo com JAKOB NIELSEN (NIELSEN, 1994), realizar alterações em projeto finalizado chega a ter um custo cem vezes mais elevado do que efetuar essas alterações antes da codificação.

Na seção a seguir são apresentados conceitos sobre Desenvolvimento Orientado a Modelos, necessários para o entendimento da proposta dessa dissertação.

2.2 Desenvolvimento orientado a modelos

Os pesquisadores e desenvolvedores de software ao longo dos anos criaram abstrações que os ajudam a programar em termos de sua intenção de projeto, e, assim, escondendo ao mesmo tempo a complexidade desses ambientes. O uso de modelos para o desenvolvimento de software vem desde o modelo conceitual de Entidade-Relacionamento - ER, descrito por Peter Chen (CHEN, 1976). Durante a década de 1980 surgiram diversas abordagens derivadas de modelo ER. Já na década de 1990, mais precisamente em 1997, surgiu a primeira versão do *Unified Modeling Language* - UML (BOOCH et al., 1996). Hoje, a UML é amplamente utilizada como base para o desenvolvimento de software baseado em modelos. ER e UML são as raízes da Model-Driven Engineering - MDE (AMELLER, 2009). A Engenharia Orientada a Modelos surgiu por volta dos anos 2000 com o aumento do uso de modelos no desenvolvimento de software, juntamente com a abordagem Model-Driven Architecture - MDA, definida pelo Object Management Group - OMG (MILLER et al., 2003).

Os conceitos derivados de iniciativas de desenvolvimento de software orientado a modelos acabam se tornando confusos. A seguir, são discutidas as principais diferenças entre Engenharia Orientada a Modelos (MDE), Arquitetura Orientada a Modelos (MDA) e Desenvolvimento Orientado a Modelo (MDD).

O MDD é um paradigma de desenvolvimento que utiliza modelos como o principal artefato do processo de desenvolvimento de software. O MDA é uma visão particular da OMG em relação ao MDD. Sendo assim, o MDA pode ser considerado como um subconjunto de MDD. Já o MDE pode ser considerado um conjunto que contém o MDD, pois sua premissa de engenharia vai além do processo de desenvolvimento de software (AMELLER, 2009). A figura 6 ilustra, em forma de conjuntos a diferenciação, entre os conceitos supracitados.

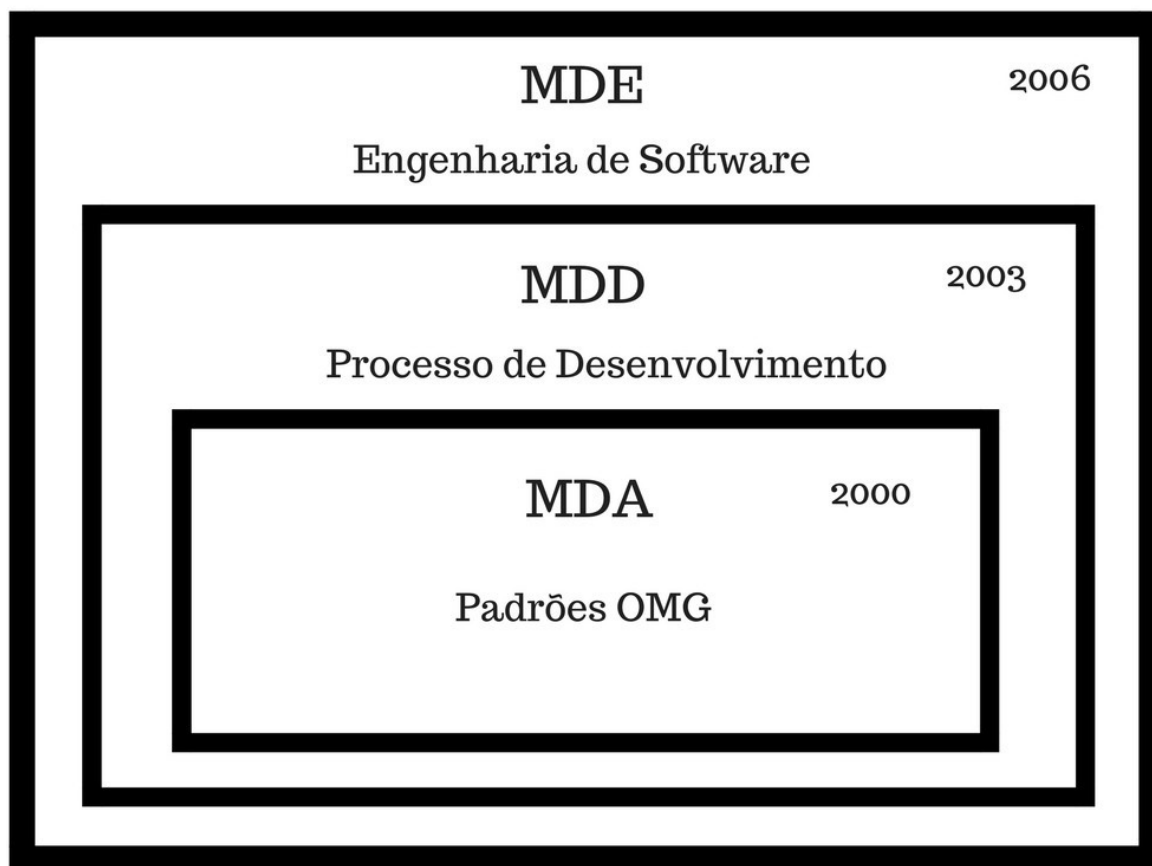


Figura 6 – Representação de Conceitos de Iniciativas Orientadas a Modelos (AMELLER, 2009)

2.2.1 Modelos e Metamodelos

Na visão da MDA, um modelo é uma representação abstrata da construção de uma software, contém todas as especificações do escopo, regras, conceitos e informações que se aplicam ao domínio desse sistema (MILLER et al., 2003). Na engenharia de software, os modelos são artefatos que fornecem uma visualização do sistema a ser construído, oferece uma nível de abstração da problemática que facilita a compreensão do escopo do sistema.

O modelos definidos pela MDA são (AMELLER, 2009):

- **Computational Independent Model - CIM:** em português, Modelo Independente de Computação, é o modelo que descreve o funcionamento do software, também chamado de modelo de negócios;
- **Platform Independent Model - PIM:** o Modelo Independente da Plataforma descreve o sistema sem ter a preocupação com a linguagem de implementação;
- **Platform Specific Model - PSM:** esse modelo é especificado com foco na tecnologia de implementação. O código é gerado a partir de transformações no PSM.

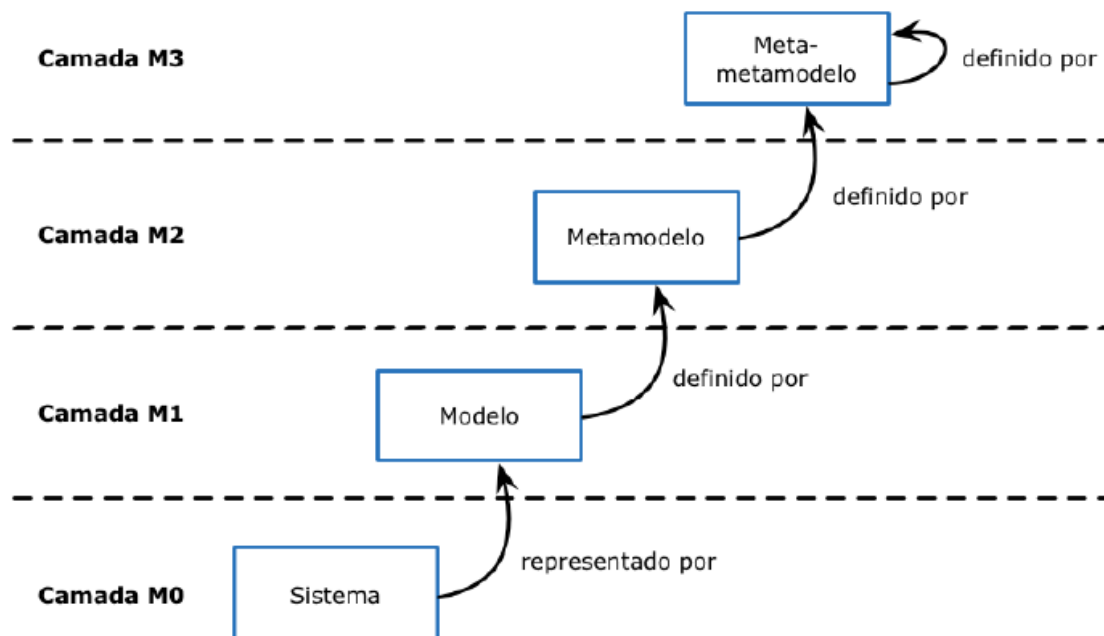


Figura 7 – Arquitetura de Camadas da MOF

A utilização da MDA no desenvolvimento de software oferece várias vantagens como o aumento na produtividade devido a redução de tempo de codificação por conta das transformações de PIM para PSM, oferece ainda o benefício da geração de código para diversas plataformas diferentes utilizando apenas um PIM. O esforço nesse processo de desenvolvimento é em gerar um modelo de qualidade, independente e com alto nível de abstração (MILLER et al., 2003).

Um metamodelo é um modelo no qual suas instâncias define outro modelo (FOWLER; PARSONS, 2013). Sendo mais específico, um metamodelo é uma representação gráfica para a formalização das informações do domínio que fornecem mecanismos independentes da plataforma para especificar a sintaxe e semântica das ferramentas a serem desenvolvidas para a solução do domínio. Os metamodelos utilizam notações em XML para realizar as transformações para a plataforma resultante. Um meta-metamodelo tem como propósito definir um metamodelo tal como o modelo é definido pelo metamodelo. (MILLER et al., 2003) A OMG define a arquitetura da *Meta Object Facility* - MOF com quatro camadas como ilustra a figura 7.

A camada M0 contém os da aplicação e representa o sistema em tempo de execução. A camada M1 contém os modelos da aplicação e está diretamente relacionada com a camada M0, cada elemento da camada M0 é representado na camada M1. A camada M2 contém o metamodelo que define o modelo da camada 1, e por fim, a camada M3 que contém o meta-metamodelo que define quais elementos podem ser exibidos pelo metamodelo, cada elemento da camada M2 são instanciados por elementos da camada 2.

2.2.2 Transformações de modelos

As transformações que tem a função de converter um modelo em outro, consiste em basicamente em mapear os elementos do modelo inicial no elementos modelo final através dessas regras de transformação. (MENS; GORP, 2006) Com a definição das regras de transformação estabelecidas, é possível definir um modelo definido pelo metamodelo alvo através de transformações automáticas. (KLEPPE; WARMER; BAST, 2003) As transformações podem ser efetuadas de texto para modelo, modelo para modelo ou de modelo para texto.

No caso em que a saída é texto, a transformação é chamada de transformação de modelo para texto ou M2T. É uma transformação comum devido ao texto ser um formato onipresente que é usado por muitas ferramentas como formato fonte, intermediário ou alvo, ou uma combinação de todos. (STEINBERG et al., 2008)

2.2.3 Geração de Código

Existem diferentes maneiras de gerar código. Algumas ferramentas podem gerar diferentes classes de arquivos de um modelo usando a transformação M2T para complementar ou substituir o modelo. Na MDE esta é a maneira mais comum de gerar código. Códigos genéricos também podem ser gerados, e muitos programas fazem isso. O importante é que o desenvolvedor use uma linguagem de programação que suporte um mecanismo eficiente e poderoso para geração de código (LARSEN, 2014). O processo MOP utiliza o Acceleo para a geração de código. Na seção de Ferramentas é apresentada um breve relato do funcionamento dessa ferramenta.

2.2.4 Ferramentas

Existem várias ferramentas que permitem desenvolver uma Linguagem de Domínio Específico. Na lista a seguir é apresentadas brevemente algumas ferramentas utilizadas atualmente:

- **DSLTOOLS:** essa é a suíte de MDD da Microsoft onde é possível criar o seu próprio editor de modelos, integrado ao Visual Studio¹. A ferramenta permite criação, edição, visualização automatização do processo de desenvolvimento de software corporativo (BÉZIVIN et al., 2005);
- **EMF/GMF:** o *Eclipse Modeling Framework* - EMF é uma ferramenta de modelagem e geração de código para a construção de ferramentas ou outro gênero de aplicações baseada em modelos de dados estruturados. A partir do metamodelo criado, o EMF proporciona, em tempo de execução, a geração e validação do código, produzindo um conjunto de classes Java agregado ao modelo,

¹ Ambiente de desenvolvimento da Microsoft

ainda oferece a visualização e edição do modelo. O EMF é constituído por uma interface gráfica para criar e editar metamodelos (Ecore) e geração de código. O editor gráfico é gerado através do *Graphical Modeling Framework* - GMF, usando o metamodelo (Ecore) para definir o sistema através de um componente gráfico (STEINBERG et al., 2008);

- **Epsilon:** é uma família de linguagens e ferramentas para geração de código, transformação do modelo a modelo, validação do modelo, comparação, migração e refatoração para trabalhar com modelos. Dentre as suas ferramentas evidencia-se o Eugenia, que é um *front-end* para GMF. O seu objetivo é acelerar o processo de desenvolvimento de um editor de GMF e diminuir a barreira de entrada para novos desenvolvedores. Para este fim, o Eugenia permite que os desenvolvedores para gerar um editor de GMF totalmente funcional apenas especificando algumas anotações de alto nível no metamodelo *Ecore* (ROSE; KOLOVOS; PAIGE, 2012).

Para a produção da processo de prototipação MOP foi utilizado a o *framework* Sirius, o qual será detalhado a seguir.

O *framework* Sirius é uma solução para o rápido desenvolvimento de editores de modelo. Sua estrutura foi construída a partir do *Graphical Modeling Framework*. Os modelos podem ser produzidos em forma de diagramas, tabelas e árvores e podem ser organizados logicamente de acordo com o ponto de vista. A ferramenta utiliza o padrão EMF, que já é bastante difundido indústria (STEINBERG et al., 2008). Os principais conceitos que norteiam desenvolvimento com o Sirius são: o ponto de vista, representação, mapeamento, estilo e ferramenta (Eclipse Foundation, 2017).

O ponto de vista é um dos principais elementos, onde é definido as especificações de representação. A representação é uma construção gráfica do domínio, descreve a estrutura, aparência e comportamento dos editores de modelo. Um mapeamento é um subconjunto do modelo, identifica como os elementos devem ser representados. Cada mapeamento pode conter diversos estilos associados a ele. A ferramenta fornece a capacidade de ação a representação. (Eclipse Foundation, 2017)

2.2.4.1 Acceleo

O Acceleo é uma ferramenta desenvolvida pela OMG, que oferece a geração de código a partir de uma linguagem de transformação conhecida como *MOF Model Text Transformation Language* - MOFM2T (SPECIFICATION, 2008). Essa linguagem permite que os desenvolvedores utilizem a abordagem orientada a modelos para a criação de aplicativos.

Com o Acceleo é possível gerar código a partir de modelos EMF. Sendo assim, é possível gerar código de qualquer linguagem textual (HTML, Java, Python e etc) a partir de modelos UML e outras linguagens de domínio específico. A ferramenta dispõe de

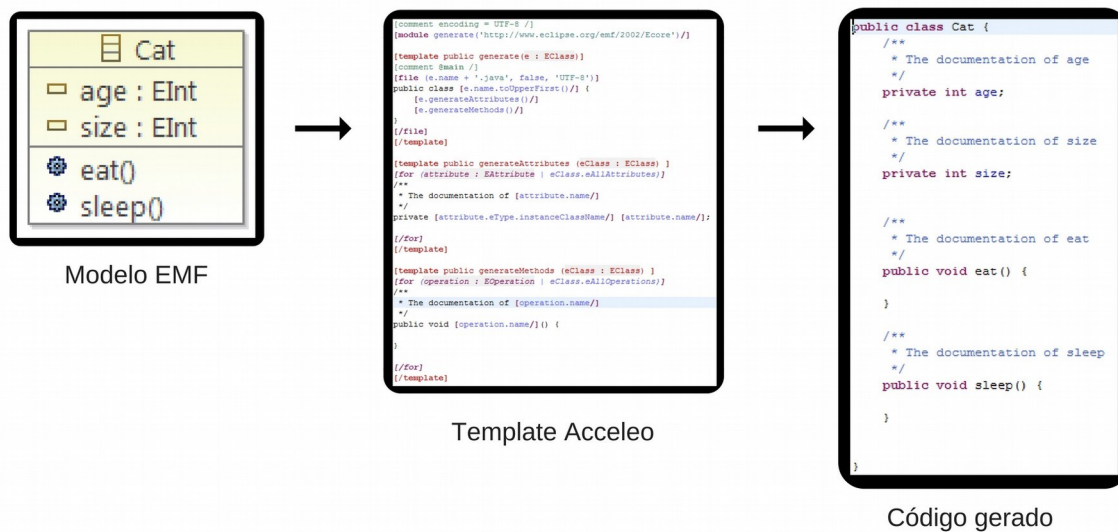


Figura 8 – Modelo de Geração de Código com Aceleo

um editor de módulos para geração de código, onde é inserida a sintaxe para realizar a transformação M2T. O código inserido no editor é baseado no modelo EMF. Após a criação do módulo é possível gerar o código na linguagem final. A figura 8, ilustra o processo de geração de código a partir de modelo EMF, o *template* Aceleo e o código final (MUSSET et al., 2006).

2.2.5 Conclusão

Nesta seção foram introduzidos conceitos sobre as MDE, MDD e MDA. Foram destacadas definições de modelo, metamodelos, arquitetura de camadas MOF, transformações de modelos e o funcionamento das principais ferramentas utilizadas para a criação do Editor de Modelos dentro da Processo de Prototipação MOP. A seguir serão apresentados conceitos sobre visão computacional e casamento de modelos.

2.3 Visão Computacional/Casamento de Modelos

A área de processamento digital de imagens, é uma subcategoria do Processamento de Sinais que utiliza diversos tipos de algoritmos para viabilizar aplicações nas áreas de Visão Computacional e Reconhecimento de Padrões. Um sistema de processamento de imagens, de uma forma sucinta, pode ser representado em quatro operações: a aquisição, onde há a entrada da imagem digital; processamento, onde há a aplicação dos algoritmos; armazenamento; e a saída (FILHO; NETO, 1999) .

Uma imagem digital é a representação numérica de uma imagem bidimensional que permite seu armazenamento, transferência, impressão ou reprodução, e seu processamento por meios eletrônicos. Matematicamente, uma imagem pode ser definida por uma **função $f(x,y)$ em R^2 , onde x e y são coordenadas espaciais e a amplitude de qualquer par ordenado (x,y) representa a intensidade de cinza da imagem naquele ponto**. Assim, uma imagem digital é uma matriz cujos índices de linhas e colunas identificam um ponto na imagem, o valor do elemento da matriz identifica o nível de cinza naquele ponto. Os elementos dessa matriz digital são chamados de elementos pictóricos, elementos da imagem, *pels* ou *pixels*. *Pixel* é o termo mais utilizado para identificar elementos em uma imagem digital (GONZALEZ; WOODS, 2010).

Em imagens binárias cada *pixel* pode assumir valores de 0 e 1. Em imagens em escala de cinza, geralmente, os valores podem variar entre 0 e 255. Já em imagens coloridas, cada componente de cor do *pixel* é representado por três ou quatro valores também variando de 0 a 255.

Para construir um sistema de visão computacional ou sistema de reconhecimento de objetos é necessários bem mais operações do que as descritas anteriormente.

2.3.1 Visão computacional

Visão computacional é um campo interdisciplinar que auxilia compreensão de imagens e vídeos por computadores. Definindo de forma mais genérica, a visão computacional procura para automatizar tarefas que o sistema visual humano pode fazer (GONZALEZ; WOODS, 2010).

As principais tarefas de um sistema de visão computacional incluem técnicas para a aquisição, processamento, análise e compreensão imagens digitais, e em geral, lidam com a extração de dados do mundo real, a fim de produzir informações numéricas ou simbólicas, por exemplo, nas formas de reconhecimento de objetos (FILHO; NETO, 1999).

Como disciplina científica, a visão computacional está preocupada com a teoria por trás de sistemas artificiais que extraem informações a partir de imagens. Os dados da imagem podem assumir muitas formas, tais como sequências de vídeo, vistas a partir de múltiplas câmaras ou de dados multidimensional de um *scanner* médico. Como uma disciplina tecnológica, procura aplicar suas teorias e modelos para a construção de sistemas de visão computacional.

Dentre algumas aplicações dos sistemas de visão computacional destacam-se: a detecção de eventos, acompanhamento de vídeo, reconhecimento de objetos e a restauração de imagem.

Cada uma das áreas de aplicação empregam uma variedade de tarefas de visão computacional, que podem ser resolvidos usando uma variedade de métodos. Alguns

exemplos de tarefas de visão computacional são apresentados a seguir:

2.3.1.1 Reconhecimento

O problema clássico em visão computacional, processamento de imagem e visão de máquina é a de determinar se ou não os dados da imagem contem algum objeto específico, recurso ou atividade. Diferentes variedades de reconhecimento do problema está descrita na literatura como:

- **Reconhecimento de objetos ou classificação de objeto:** Reconhecimento de objetos é um processo para identificar um objeto específico em uma imagem digital ou vídeo. Algoritmos de reconhecimento de objeto dependem de correspondência, aprendizagem, ou de reconhecimento de padrões algoritmos usando baseado em aparência. o reconhecimento de objetos é útil em aplicações, tais como a estabilização de vídeo, sistemas de estacionamento de veículos automáticos, e a reconhecimento de placas de sinalização de trânsito (BRADSKI, 1998);
- **Identificação:** os exemplos incluem a identificação de face ou de impressão digital de uma pessoa específica, escrita, ou identificação de um veículo específico (BRADSKI, 1998);
- **Deteção:** utilizados para detecção de possíveis células ou tecidos em imagens médicas ou a detecção de um veículo anormais em um sistema de polícia rodoviária;
- **Recuperação de imagens baseada em conteúdo:** consiste em uma busca feita por imagens semelhantes a um determinado padrão fornecido. O que torna isso possível é a comparação feita entre aspectos da imagem dada e as imagens armazenadas na base de dados. Os sistemas *Content-Based Image Retrieval* CBIR permitem o retorno de imagens utilizando outras imagens para a busca. Os critérios de similaridade são obtidos a partir da extração de características da imagem como cor, textura e forma (RUI et al., 1998);
- **Estimativa de pose:** usado para estimar a posição ou orientação de um objeto em relação a câmara;
- **Reconhecimento óptico de caracteres (OCR):** identificação de caracteres em imagens de texto impresso ou manuscrito (GOMES et al., 2014);
- **Leitura de códigos 2D:** usado em leitura de códigos 2D, tais como matriz de dados e *QR codes*;
- **Reconhecimento facial** (BRADSKI, 1998);

2.3.1.2 Análise de movimento

Existem várias tarefas relacionadas com a análise de movimento. Exemplos dessas tarefas são:

- **Vigilância automatizada:** o monitoramento de um local para detectar suspeitos atividades ou eventos improváveis;
- **Rastreamento:** seguindo os movimentos de um conjunto de pontos de interesse ou objetos na sequência de imagens (BRADSKI, 1998);
- **Monitoramento de tráfego:** encontro em tempo real das estatísticas de tráfego para direcionar o fluxo tráfego (COIFMAN et al., 1998);
- **Navegação do veículo:** planejamento de trajetória baseada em vídeo.

2.3.1.3 Restauração de imagens

O principal objetivo é a remoção ou redução de degradações das imagens geradas no processo de aquisição. As principais fontes de degradações incluem ruído no sensor, perda de foco, movimento relativo da câmera, turbulência atmosférica, entre outras. Em geral, técnicas de restauração consideram o modelo de degradação e usam um processo inverso para obter a imagem original (GONZALEZ; WOODS, 2010).

2.3.2 Sistema de visão computacional

A organização de um sistema de visão computacional é altamente dependente da aplicação. Os métodos de reconhecimento de padrões têm a função de detectar padrões em imagens digitais a partir de exemplos fornecidos, também conhecidos como modelos, máscaras ou *templates*. Um sistema de visão computacional fornece a uma máquina a capacidade de emular algumas funções de um sistema visual biológico, como a habilidade detalhar uma cena contida em uma imagem digital, a partir da análise dos padrões contidos nela, independente de cor, tamanho, posição, orientação ou quaisquer padrões visuais (GONZALEZ; WOODS, 2010; FILHO; NETO, 1999). A Figura 9 apresenta um sistema de visão computacional típico, que compreende etapas de aquisição de imagens, pré-processamento, segmentação, representação, descrição ou extração de características e reconhecimento, podendo ser classificado em três níveis de abstração.

A primeira etapa do sistema é a aquisição da imagem. Uma imagem digital é capturada por um ou vários sensores de imagem, que, além de vários tipos de câmeras sensíveis à luz, incluem sensores de gama, aparelhos de tomografia, radar, câmeras de ultra-som, dentre outros. Pode ter como resultado uma imagem 2D comum, um volume 3D, ou uma sequência de imagens.

A segunda etapa é o pré-processamento que geralmente envolve técnicas de melhoria de contraste, remoção de ruído e correção do foco. É nessa fase que são realçados pontos

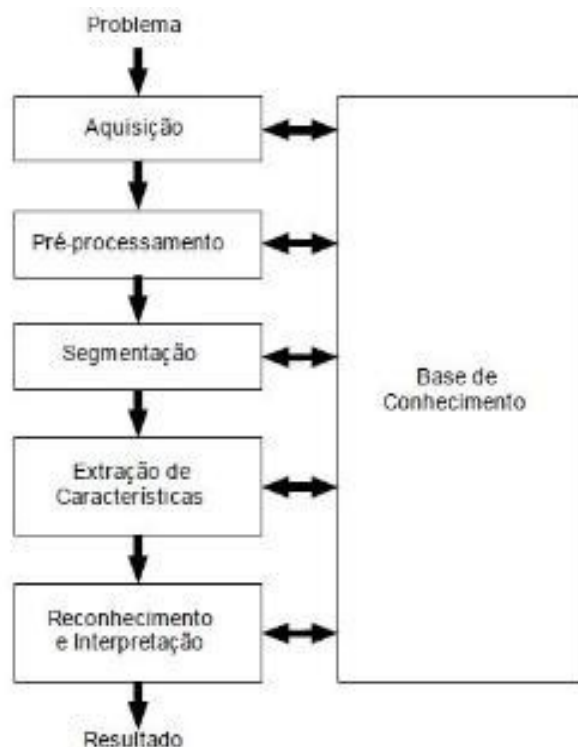


Figura 9 – Sistema de visão computacional extraído de (GONZALEZ; WOODS, 2010)

de interesse, tais como pontos, bolhas e cantos, para que facilite a visualização ou a segmentação. A etapa a seguir refere-se à segmentação que tem como objetivo separar um conjunto de objetos ou regiões de interesse. A segmentação é um dos processos mais importantes e interfere diretamente no resultado. Um exemplo de segmentação é o uso de detecção de bordas para a extração de caracteres individuais em placas. Outro exemplo comum de segmentação é a binarização onde os objetos da imagem resultante são regiões contíguas de *pixels* brancos e em fundo preto ou vice-versa.

A extração de atributos ocorre na etapa de representação e descrição. No caso de ser uma forma, a extração pode ser feita a partir da região ou contorno do objeto. Em algumas aplicações são combinados descritores de contorno e região. O último passo é o reconhecimento e interpretação, pode-se dizer que a análise das imagens nada mais é que um processo de descobrimento, de identificação e entendimento de padrões. O reconhecimento e interpretação de imagem se enquadram no alto nível de processamento de imagens e pode ser visto como um processo que rotula um padrão, com base no conjunto de atributos da base de conhecimento.

2.3.3 Casamento de Modelos (*Template Matching*)

O Casamento de Modelos ou *Template Matching* é uma técnica de reconhecimento de padrões que visa encontrar similaridades entre uma imagem e uma imagem modelo através da análise de características ou regiões de interesse. A Figura 10 apresenta o

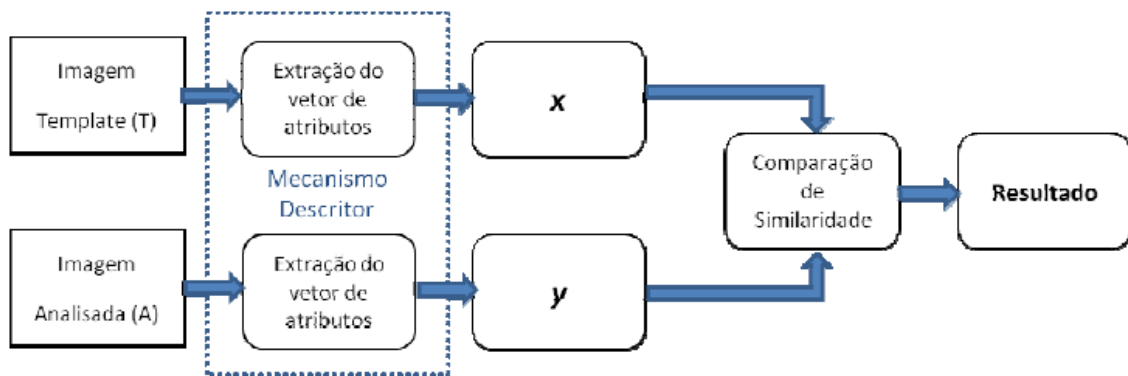


Figura 10 – Processo de Casamento de Modelos (ARAÚJO, 2009)

processo de Casamento de Modelos, que se inicia com a extração de características visuais de ambas as imagens que resultam em vetores, depois é feita a comparação, onde é verificado se há semelhanças entre a imagem e o modelo, se o grau de similaridade estiver de acordo com o limiar definido pelo usuário então diz-se que há um casamento entre a imagem e o modelo (*template*) (ARAÚJO, 2009).

No processo de prototipação MOP, os conceitos de processamento digital de imagens e *template matching* foram implementados utilizando *Computer Vision System Toolbox* (MATHWORKS, 2017a) e *Image Processing Toolbox* (MATHWORKS, 2017b) do software de programação matricial MATLAB. Esse software provê um conjunto de funções e aplicativos que ajudam a projetar e simular sistemas de visão computacional. Com ele é possível realizar detecção de características, extração e correspondência, detecção de objetos e processamento de vídeo, além de geração de código.

A *Computer Vision System Toolbox* contém os algoritmos que realizam o casamento de modelos. Eles utilizam conceitos matemáticos como a Soma das Diferenças Absolutas ou Distância de Manhattan, a Soma das Diferenças dos Quadrados ou Distância Euclidiana e a Máxima Diferença Absoluta, também conhecida com Distância Uniforme.

2.4 Conclusão

Esse capítulo abordou os principais conceitos de tecnologias e ferramentas utilizadas para a concepção do processo de prototipação MOP. No próximo capítulo é apresentado uma revisão sistemática de literatura com foco na identificação de trabalhos contendo técnicas e ferramentas de prototipação de interface para dispositivos móveis.

3 Trabalhos Relacionados

Como forma de elencar trabalhos relacionados foi realizada uma revisão sistemática baseada no processo de (BIOLCHINI et al., 2005). Uma revisão sistemática da literatura tem como objetivo identificar, avaliar e interpretar todas as pesquisas relevantes para uma determinada questão de pesquisa (BIOLCHINI et al., 2005). A questão de pesquisa que foi definida é: “Quais as principais técnicas e ferramentas para prototipação de interfaces para dispositivos móveis?”. A realização de uma revisão sistemática segue um protocolo de etapas bem definidas com ilustrado na Figura 11. A sua execução é bastante exaustiva, sendo assim, foi utilizada o StArt - *State of Art through System Review*, uma ferramenta de apoio ao pesquisador.

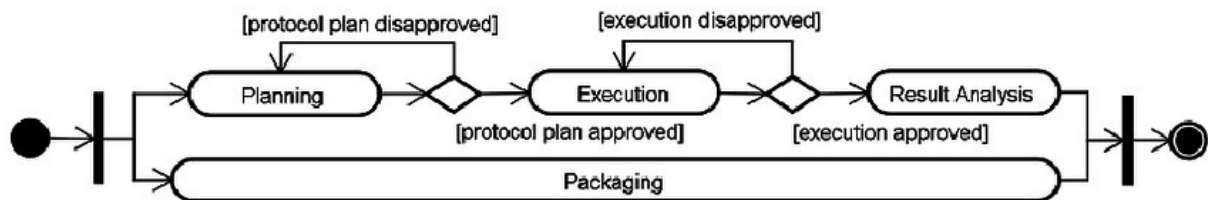


Figura 11 – Processo de Revisão Sistemática (BIOLCHINI et al., 2005).

3.1 Planejamento e condução da revisão

O planejamento é a fase inicial do processo de revisão sistemática, nessa fase são definidos os objetivos da pesquisa, as questões a serem investigadas, definição de repositórios e estratégias de buscas. Para conduzir a pesquisa é feita a identificação, seleção e análise dos artigos, sendo classificados de acordo com os critérios de inclusão e exclusão.

O capítulo está dividido da seguinte forma: a seção 2 descreve o processo de projetar e executar a revisão sistemática, a seção 3 detalha os resultados obtidos e, finalmente, a seção 4 apresenta as conclusões e trabalhos futuros.

3.1.1 Objetivos da pesquisa

Identificar e analisar as principais técnicas e ferramentas de prototipação de interface, que vêm sendo desenvolvidas para aplicações em dispositivos móveis.

3.1.2 Questão da pesquisa

Para alcançar o objetivo enunciado nesta revisão foi definida uma questão, que também servirá de apoio para os critérios de inclusão e exclusão dos artigos identificados.

Questão: quais as técnicas e ferramentas de prototipação de interfaces para dispositivos móveis? O que os estudos atuais mostram? Intervenção técnicas e ferramentas de prototipação de interface de usuário

- **Efeito:** análise crítica das técnicas e ferramentas de prototipação de interface de usuário para dispositivos móveis;
- **Aplicação:** academia e indústria de software;
- **Resultado:** técnicas e ferramentas de prototipação de interface de usuário aplicadas ao contexto de dispositivos móveis;
- **População:** projetos de desenvolvimento de aplicações para dispositivos móveis, que adotem a prototipação de interface de usuário como fase de desenvolvimento.

3.2 Estratégia de busca e seleção de fontes

Nessa fase é definido os repositórios de busca linguagem e as palavras-chave.

Fonte: dentre os principais repositórios de artigos relacionados ao tema foram escolhidos o ACM DL, IEEE Xplore, ScienceDirect e Scopus. Foram aceitos artigos da língua portuguesa ou inglesa e as palavras-chave foram geradas de acordo com a questão e sua especificidade.

- **Intervenção:** *wireframe, user experience, UX, user interface, UI prototyping, paper prototyping, mockup, sketch, rapid prototyping, low-fi, lo-fi, low-fidelity, hi-fi, high-fidelity, stencils, framework.*
- **População:** *mobile development application.*
- **Resultado:** *prototyping, prototype.*

3.2.1 Critérios de seleção

Nessa fase é definido o conjunto de critérios de seleção dos artigos primários. A inclusão dos estudos foi determinada pelos seguintes critérios:

- Estudos sobre prototipação de interface de usuários para aplicações em dispositivos móveis.
- Estudos que estão disponíveis na web.
- Estudos nas línguas inglesa e portuguesa.
- Estudos publicados entre 2010 e março de 2015.
- Estudos com conteúdo integral.
- Estudos duplicados não entram na seleção.

3.2.2 Processo de seleção preliminar

O processo de seleção se inicia com a construção de strings de buscas baseadas nas palavras-chave definidas anteriormente. As strings são inseridas nos motores de busca

dos repositórios que retorna o conjunto de artigos primário. É feita uma leitura das palavras-chave, resumo, introdução e conclusão de cada artigo, quando disponível, para se definir se é relevante a pesquisa ou não, com base nos critérios de seleção.

3.2.3 Processo de seleção final

Após realizada a seleção dos artigos relevantes é feita uma leitura completa dos estudos com o objetivo de ratificar sua relevância à pesquisa. Por ser uma leitura mais profunda essa fase pode acarretar mudanças no protocolo definido, pois a cada artigo lido é possível reavaliar os critérios de inclusão e exclusão e balancear a importância de diferentes estudos.

3.2.4 Extração de informações

Uma das premissas da extração é gerar um formulário para a coleta das informações relevantes a pesquisa. Os principais dados extraídos foram:

- **Projeto:** nome da técnica ou ferramenta;
- **Visão geral:** breve análise do trabalho;
- **Metodologia:** definição da técnica ou ferramenta desenvolvida;
- **Arquitetura:** tipo de arquitetura do software;
- **Plataforma:** plataforma a ser implantada (Android, iOS e etc);
- **Programação:** nível de conhecimento em programação necessário.

Após a extração é realizada a uma análise estatística dos resultados, gerando tabelas e gráficos, bem como um relatório com uma síntese dissertativa das informações.

3.2.5 Condução da pesquisa

A revisão foi conduzida no ano de 2015 entre os meses de março e maio. Foram encontrados um total 303 estudos nos 4 repositórios escolhidos. Todos os artigos passaram pelas fases de seleção inicial, seleção final e extração de informações. Durante a fase de condução são criadas as strings de busca de acordo com as peculiaridades de cada repositório. A Tabela 1 apresenta as strings geradas para a condução da revisão.

A etapa de seleção preliminar foi realizada com a leitura do título, resumo, palavras-chave, introdução e conclusão. De acordo com as informações obtidas foi determinada a inclusão ou exclusão do artigo da revisão sistemática. Os artigos que passaram para a seleção final foram lidos por completo e suas informações foram extraídas de acordo com formulário definido durante a fase do protocolo. A Tabela 2 apresenta a classificação dos artigos de acordo com o seu repositório.

Tabela 1 – Fontes de buscas e strings

Repositório	String	Observação
IEEE Xplore	(((((Stencils OR "software user interface"OR "paper prototyping"OR wireframe OR "user interface"OR "user experience"OR mockup OR "UI prototyping"OR framework OR UX OR "lowfidelity"OR "high-fidelity"OR "rapid prototyping") AND (mobileapplication prototyping))))	Busca em metadados 2010 a 2015
ACM DL	((("Mobile Application development") and (prototyping orprototype) and (mockup or mock-up or wireframe or "software userinterface"or stencils or framework or "paper prototyping"or "UI prototyping"or "user experience"or UI or UX or "low-fidelity"or"high-fidelity"or "rapid prototyping"))	2010 a 2015
ScienceDirect	("Mobile Application development") AND (prototyping or prototype) AND (mockup OR wireframe OR "software user interface"OR stencils OR "paper prototyping"OR "UI prototyping"OR "user experience"OR UI OR "UX"OR "low-fidelity"OR "high-fidelity"OR "rapidprototyping")	Busca em metadados 2010 a 2015
Scopus	(TITLE-ABS-KEY ((mockup OR wireframe OR stencils ORframework OR "user experience"OR uiOR ux OR prototypeOR "software user interface"OR ui prototyping OR "UIprototyping"OR "paper prototyping"OR "rapid prototyping"OR "low-fidelity"OR "high-fidelity")) AND TITLE-ABSKEY((mobile application development AND prototyping)))AND (LIMIT-TO (PUBYEAR , 2015) OR LIMIT-TO (PUBYEAR , 2014) OR LIMIT-TO (PUBYEAR , 2013) ORLIMIT-TO (PUBYEAR , 2012) OR LIMIT-TO (PUBYEAR ,2011) OR LIMIT-TO (PUBYEAR , 2010)))	Busca em metadados 2010 a 2015

Durante leitura de **PID-MAD 2014: Second International Workshop on Prototyping to Support the Interaction Designing in Mobile Application Development** foram identificados vários estudos significativos que foram inseridos manualmente na revisão sistemática; o PID-MAD é um workshop que acontece em conjunto com MobileHCI, uma das mais importantes conferências do mundo no tocante a interação humano-computador em dispositivos móveis. Ele oferece uma plataforma para as comunidades interessadas discutirem o tema, compartilharem experiências, a fim de enfrentar os desafios futuros. Os artigos inseridos são listados a seguir:

Managing Consistency in Wizard of Oz Studies: A Challenge of Prototyping Natural Language Interactions (HÖYSNIEMI; HÄMÄLÄINEN; TURKKI, 2004);

Rapid creation of sketch-based native Android prototypes with Blended Prototyping (BAHR, 2013);

A Blackboard-Like Architecture for the Development of Evolving High Fidelity

Tabela 2 – Seleção de artigos por repositório

Fonte	Encontrados	Duplicados	Seleção inicial	Seleção final
IEEE Xplore	52	0	13	7
ACM DL	148	6	55	35
ScienceDirect	14	3	6	3
Scopus	89	9	23	16
Manual	7	0	7	7
TOTAL	310	18	104	68

Mobile Application Prototypes (IGLER; BRAUMANN; BÖHM,);

Toward a more conscious use of prototyping in mobile user experience design (Thorsten Hochreuter, Kirstin Kohler, and Mareen Maurer);

Integrating Mobile Design Patterns within a User Interface Development Environment (GIOVANNI; SEBILLO; VITIELLO,);

Towards Using Context Personas to Support Prototyping of Mobile Business Apps (KIEFER; HESS, 2013);

A Tool-based Approach for Structuring Feedback for User Interface Evaluations of Mobile Applications (BÖHM; IGLER, 2013).

Como ilustrado na Tabela 2 um total de 68 artigos foram selecionados para a extração de informações. Desse montante, 57 foram publicados em 2013 e 2014, outros 40 nos anos de 2010, 2011, 2012 e apenas 2 artigos foram publicados até abril de 2015.

3.3 Resultados

Os resultados obtidos foram classificados com base no formulário de extração de conhecimento. Pôde-se gerar tabelas e informações estatísticas a partir da tabulação dos dados. As seções seguintes apresentarão resultados mais detalhados de cada classificação.

3.3.1 Plataformas

Esta seção apresenta a classificação dos estudos de acordo com a plataforma para qual foi desenvolvida a técnica ou ferramenta de prototipação. A maioria dos estudos selecionados aponta para um conceito de desenvolvimento multiplataforma, geralmente baseado em HTML5, CSS3 e JavaScript.

3.3.2 Arquitetura

Aplicativos para dispositivos móveis podem ser desenvolvidos baseados em páginas web, nativamente ou de forma híbrida (LEICHTENSTERN; ANDRÉ; REHM, 2010). Ainda

Tabela 3 – Classificação dos artigos quanto a plataforma

Plataforma	Estudos	Percentual
Android	E1, E4, E10, E17, E19, E26, E29, E35, E36, E37, E38, E63	38,7,64%
JavaME	E2	1,4%
BlackBerry	E15	1,4%
iOS	E41, E44	2,94%
Multiplataforma	E3, E5, E6, E7, E8, E9, E12, E13, E14, E16, E18, E20, E21, E22, E23, E24, E27, E28, E30, E32, E34, E39, E40, E43, E45, E46, E47, E48, E49, E51, E52, E53, E54, E55, E56, E58, E59, E60, E61, E62, E64, E65, E66, E67, E68	66,17%

Tabela 4 – Classificação dos artigos quanto a arquitetura

Plataforma	Estudos	Percentual
Web-based	E6, E9, E13, E16, E17, E27, E30, E32, E43, E48, E51, E52, E53, E54	42,3,52%
Cloud-based	E18, E23	2,94%
Nativa	E1, E2, E3, E4, E5, E7, E10, E12, E14, E15, E19, E20, E1, E22, E24, E26, E28, E29, E36, E37, E38, E39, E41, E45, E46, E49, E62, E63, E64, E65, E66, E67, E68	47,05%

existem os aplicativos cloud-based que se aproximam dos serviços oferecidos na nuvem como: sincronização de scripts de desenvolvimento; autenticação automatizada e TaaS – Testing as an Service [E18, E23]. Os aplicativos baseados em web são aqueles que podem rodados via um navegador, os nativos são compilados para rodar expressamente em dispositivos móveis e os híbridos podem atuar das duas formas. Os aplicativos web tem vantagem na frente dos nativos, pois esse devem ser recompilados para cada plataforma diferente [E11]. A Tabela 5 apresenta a classificação dos estudos de acordo com a arquitetura de desenvolvimento.

3.3.3 Ferramentas e técnicas

Em diversos artigos foram apresentadas ferramentas para prototipação de interface de usuário para os dispositivos móveis. Essas ferramentas variam de simples aplicações de *wireframing* tradicionais para ferramentas de prototipação avançadas que são capazes de fornecer uma grande variedade de funcionalidades, tal como eventos de toque e gestos, interações, transições de tela. Mais importante essas ferramentas fornecem a capacidade de visualizar um protótipo do dispositivo real. A seguir uma lista com comentários sobre as principais ferramentas citadas nos estudos:

- **PhoneGap:** é um *framework open source* para criar rapidamente aplicativos

móveis multiplataforma usando HTML5, Javascript e CSS. PhoneGap já foi baixado mais de 1 milhão de vezes e está sendo usado por mais de 1.000.000 desenvolvedores. Milhares de aplicativos criados usando PhoneGap estão disponíveis em lojas de aplicativos móveis e diretórios.

- **Xcode:** a IDE está no centro da experiência de desenvolvimento da Apple. Totalmente integrado com os quadros de Cocoa e Cocoa Touch, Xcode é um ambiente extremamente produtivo para a criação de aplicativos incríveis para Mac, iPhone e iPad.
- **TouchStudio:** esse projeto foi renomeado para TouchDevelop. É um amigável ambiente de criação de aplicativo móvel. Com TouchDevelop você pode criar aplicativos em seu telefone móvel, *tablet* e PC, e compartilhar os aplicativos que você criar na nuvem.
- **App Inventor:** a ferramenta foi citada em artigos dessa revisão. Ele foi atualizado pelo MIT – Instituto de Tecnologia de Massachusetts, e agora chama-se Inventor MIT app, que permite desenvolver aplicações para dispositivos Android usando um navegador da web e quer um telefone ou emulador conectado. Você também pode usar o site para armazenar seu trabalho e manter o controle de seus projetos. App Inventor foi originalmente desenvolvido pela Google.
- **Titanium:** oferece criar aplicações nativas, híbridas, ou aplicativos web móveis em todas as plataformas de uma única base de código JavaScript. Titanium lhe permite oferecer uma experiência de usuário mais aprofundada em uma fração do tempo, mas com os mesmos resultados, de plataformas nativas. Atualmente, Titanium oferece suporte a sistemas operacionais para smartphones móveis, como o iPhone da Apple, Android, do Google, e Mobile Web. Outras plataformas, como Windows Phone, estão atualmente em desenvolvimento.
- **Balsamiq:** a ferramenta foi lançada em junho de 2008, é uma interface gráfica para criação de mockups e wireframes. É baseado em widgets pré-construídos usando o conceito de WYSIWYG para a implementação dos protótipos.
- **AnteType:** ferramenta bastante eficiente para contruir UIs responsivos para apps móveis. Dispõe de mais de 400 widgets para as principais plataformas. Com recursos de interação do AnteType, você pode facilmente demonstrar a sensação de seu protótipo. Definir ações, eventos, estados do widget e mudança de telas.
- **Axure:** oferece a oportunidade de rapidamente criar mockups simples ou altamente funcionais, protótipos ricos com lógica condicional, conteúdo dinâmico, animações, arrastar e soltar e cálculos.
- **Fluid UI:** é uma ferramenta de prototipação de app móvel amigável e muito intuitiva. Projetado com um objetivo, para poupar tempo e dinheiro no tocante a prototipação de UI / *wireframing* e levantamento de requisitos para apps. Com ele é possível rapidamente interagir com projetos em um dispositivo móvel para

ajudar a testar e design.

- **POP:** é um programa com conceito simples, primeiramente cria-se os protótipos em papel, com a câmera do aplicativo fotografa-se os esboços, finalmente utilizando recursos do POP, é incluído atribui-se botões as imagens digitalizadas e é criada então as ligações entre as telas.
- **Kendo UI:** é um *framework* de UI com foco em jQuery. Fazendo uso de tecnologia de processamento adaptável, ele emula UI de aparência nativa sem a necessidade de que desenvolvedores que tenham de mudar qualquer coisa.

Existe uma iniciativa da comunidade de Design de UX, mais especificamente Javier Simón Cuello¹, que mantém uma lista atualizada periodicamente das principais ferramentas de prototipação de interfaces. Essa lista está disponível no endereço eletrônico <http://www.prototypingtools.co>. A Tabela 5 apresenta parte da lista mantida pela comunidade supracitada.

Durante a pesquisa bibliográfica foram encontrados trabalhos de relevância no contexto de desenvolvimento orientado a modelos e processos de prototipação. Esses trabalhos apresentam linguagens de domínio específico que fornecem soluções práticas e eficientes para criação de aplicações móveis e multiplataforma, tendo cada uma suas peculiaridades, bem como técnicas de prototipação de interfaces que utilizam visão computacional ou prototipação no próprio dispositivo.

POP (Marvel, 2017), *Prototyping On Paper*, é um aplicativo que permite fotografar esboços de telas e personalizá-las para serem navegadas como se fossem um aplicativo nativo. É possível importar várias imagens de telas ao mesmo tempo e adaptá-las através de recursos básicos de edição. É possível criar pontos na tela que simulam botões oferecendo o recurso de interatividade que não existe no protótipo em papel. Em Paper in Screen (BOLCHINI; PULIDO; FAIOLA, 2009) é utilizado o mesmo conceito: coloca-se o protótipo de papel da aplicação móvel dentro do dispositivo móvel. Isso é feito através da adaptação de imagens de protótipos de baixa fidelidade (papel) para exibição na tela de um dispositivo móvel. Permitindo que os usuários se aproximem da experiência real do usuário de dispositivos móveis que estão ausentes quando se utiliza apenas as interfaces de papel para testes de usabilidade. Essa técnica de prototipação é prática, uma vez que utiliza protótipos que provavelmente foram feitos anteriormente para testes. Se um designer esboçou uma série de protótipos de papel para realizar um teste de usabilidade ou uma avaliação heurística, estes, podem ser facilmente transformados em protótipos de papel na tela. Esses protótipos de fidelidade mista compartilham algumas características importantes com a prototipação de papel. Eles ainda são relativamente fáceis de criar e descartar. Eles tem a aparência de inacabados, que permite aos participantes forneçam o *feedback* sem medo, pois percebe-se

¹ Designer de UX responsável pela atualização lista das principais ferramentas de prototipação de interface

que protótipo é de fácil implementação e pode ser feito em pouco tempo. A prototipação de papel na tela e a prototipação regular de papel têm um lugar semelhante no ciclo de vida geral do projeto de uma aplicação. Ambos permitem testes rápidos de produção e projeto iterativo (BOLCHINI; PULIDO; FAIOLA, 2009). DENIM (LIN et al., 2000) e SILK (LANDAY, 1996) são exemplos de ferramentas de prototipação que permitem criar rapidamente protótipos baseados em esboços de telas e interagir com eles no computador, incluindo também a possibilidade de substituir componentes desenhados por componentes digitais.

Sá e Carriço (SÁ; CARRIÇO, 2009) apresentam uma ferramenta que fornece suporte para prototipação *in-situ* e avaliação diretamente em dispositivos móveis. A ferramenta permite aos projetistas criarem e refinarem protótipos de fidelidade mista. Utilizando o mesmo conceito do MobiDev (PFLEGING et al., 2010) é um ambiente de desenvolvimento que permite a criação de aplicações diretamente no dispositivo móvel usando desenhos em papel como um ponto de partida para criar a interface de usuário. Os usuários desenharam esboços da interface no papel como o passo inicial de um processo de prototipação evolucionário, que diminui a entrada de texto e acelera o processo de desenvolvimento. MobiDev permite aos usuários criar interfaces de usuários de três maneiras: Inserindo o código textualmente; usando o UIBuilder ou o SketchBuilder, esses dois últimos são projetados para acelerar e facilitar o desenvolvimento em comparação com a digitação de todo o código necessário.

No estudo de Lachgar e Abdali (LACHGAR; ABDALI, 2015) é proposta uma abordagem para a concepção da interfaces de usuário para aplicações móveis e web. A ferramenta usa conceitos de desenvolvimento orientado a modelos para a geração dos protótipos. Uma aplicação é composta de várias páginas e cada página contém uma coleção de elementos. Um elemento pode representar grupos de elementos ou um componente único. Todos os componentes são definidos na linguagem de domínio específico. O processo de prototipação das interfaces é feito em quatro passos: Análise e modelagem textual das interfaces gráficas, conforme a linguagem de domínio específico; Validação do modelo por meio de Xtend 2; Transformação de modelos PIM em PSM por meio de Xtend 2; Geração de código de acordo com plataforma de destino. O WireframeSketcher (RAMON et al., 2013) é uma ferramenta de *wireframing* executada como plugin da IDE Eclipse, que ajuda os designers a criar rapidamente protótipos para *desktop*, *web* e dispositivos móveis. Ele pode ser executado como um aplicativo de desktop, bem como um plug-in para qualquer Eclipse IDE. O processo de prototipação funciona da seguinte forma: as partes interessadas e os desenvolvedores se encontram e criam uma interface inicial (com a ajuda do WireframeSketcher) que é discutido e iterativamente refinado até que seja validado. Então, o desenvolvedor coloca o modelo da interface em um projeto Eclipse e executa o *plugin* depois de a plataforma de destino. Finalmente, o código da interface final é automaticamente gerados nesse projeto.

Tabela 5 – Principais ferramentas de prototipação. (CUELLO; VITTONI, 2013)

Nome	Roda em	Protótipo para	Saída
Antetype	OS X	Qualquer	HTML, iOS App Player
App Cooker	iPad	iOS	PDF, App Player
Atomic	Web	Qualquer	HTML
Avocado	OS X	Android, iOS	Desktop
Axure	OS X, Windows	Qualquer	HTML, PDF
Briefs	OS X	iOS	App Player
Canvas Flip	Web	Android, iOS, Web	HTML
Codiqa Web	Web	Mobile	HTML
Codiqa Desktop	OS X, Windows	Mobile	HTML
Concept.ly	Web	iPad, iPhone, Web	HTML
Demonstrate	iOS	iPhone	App Player
Evolus Pencil	Linux, OS X, Windows	Qualquer	HTML, PDF
Flinto for Mac	OS X	iOS	App Player
Flinto Lite	Web	Android, iOS	HTML
Fluid UI	Web	Qualquer	HTML, App Player
Form	OS X	iOS	App Player
Framer	OS X	Android, iOS	HTML
Framework7	Web	Android, iOS	HTML
HotGloo	Web	Qualquer	HTML, PDF, PNG
Indigo Studio	OS X, Windows	Qualquer	HTML, PDF
InVision	Web	Android, iOS, Web	HTML
iRise	Web, OS X, Windows	Android, iOS, Web	HTML
Justinmind	OS X, Windows	Android, iOS, Web	HTML, App Player
Marvel	Web	Qualquer	HTML
Marvel Apps	Android, iOS	Android, iOS	HTML, App Player
Mockingbird	Web	Qualquer	HTML, PDF
Mockplus	OS X, Windows	Qualquer	HTML
Moqups	Web	Qualquer	HTML, PDF
Neonto	OS X	Android, iOS	Source code
Origami	OS X	Android, iOS	Desktop
Pidoco	Web	Qualquer	HTML, PDF, App Player
POP	Android, iOS, WP	Android, iOS, WP	HTML, App Player
Principle	OS X	iOS	App Player
Proto.io	Web	Qualquer	HTML, PDF, App Player
ProtoPie (Beta)	Mac	Android, iOS	App Player
Protoshare	Web	HTML	HTML
Prott	OS X, Web, Windows	HTML	HTML
Prott App	Android, iOS	Android, iOS	HTML, App Player
Ratchet	Any (HTML)	HTML	Android, iOS
Solidify	Web	Qualquer	HTML
UX App	Web	Qualquer	HTML, PNG
UXPin	Web	Qualquer	HTML, PDF
Weld	Web	Qualquer	HTML

3.4 Conclusão

Com os resultados obtidos nesta revisão sistemática foi permitido conhecer quais os estudos que vêm sendo desenvolvidos no tocante a prototipação de interface de usuário em dispositivos móveis. Foi realizado um levantamento estatístico dos principais tópicos do tema assim como uma análise do conteúdo selecionado.

Embora existam várias ferramentas de prototipação, elas exigem constantes adaptações e extensões para acompanhar o progresso das tecnologias móveis. Como aplicativos móveis se tornam mais sofisticados, uma prototipação mais detalhada se torna necessária. O mercado de telefonia móvel cresce e o mercado de ferramentas de prototipação vai continuar a crescer com ele. Novos sistemas operacionais e dispositivos mais eficientes são desenvolvidos, por exemplo, painéis de controle para navegação automóvel e demandam novos tipos de interfaces e interação com usuário.

4 O Processo MOP

O processo MOP (Figura 12) é permitir que a partir do reconhecimento de padrões das imagens dos protótipos de baixa fidelidade em papel gerar um modelo em conformidade com uma Linguagem Específica de Domínio, além de uma versão executável, para ser utilizada diretamente no dispositivo móvel que adquiriu a imagem. O modelo gerado poderá ser manipulado pelos projetistas com o objetivo de inserir melhorias e correções, podendo gerar uma nova versão executável ou uma versão atualizado do projeto em papel. Esse processo de prototipação pode aproveitar protótipos que eventualmente tenham sido produzidos anteriormente para testes de usabilidade ou uma avaliação heurística para gerar protótipos de alta fidelidade na tela.

4.1 Etapas

A Figura 12 ilustra o processo MOP, evidenciando as principais tarefas realizadas por cada ferramenta utilizada no processo. O roteiro para o desenvolvimento do protótipo com o processo proposto deve ser seguido as seguintes etapas:

1. **Desenhe:** O passo inicial é desenhar as telas utilizando um estêncil. As ligações das telas podem ser inseridas no próprio desenho ou na fase de refinamento;
2. **Digitalize:** utiliza-se o aplicativo para digitalizar as imagens. As imagens são enviadas pelo aplicativo ao sistema de visão computacional, onde são realizados procedimentos de reconhecimento de formas e casamento de modelos. Após o processamento digital da imagem é gerada a metamodelo, que descreve a imagem reconhecida em forma de texto. O arquivo é enviado para o editor de modelos;

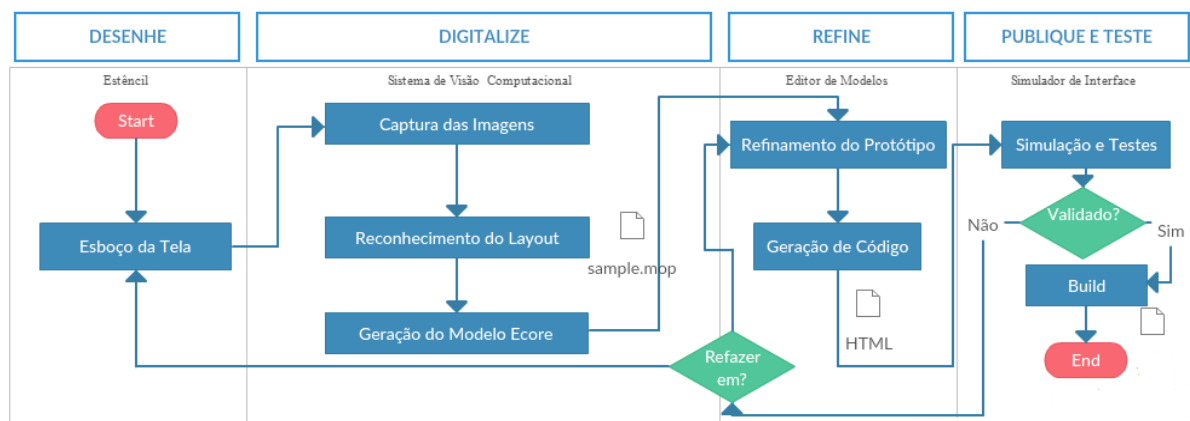


Figura 12 – Processo MOP.

3. **Refine:** com o editor de modelos pode-se refinar a interface do protótipo, trocando botões, editando *labels*, inserindo interações, dentre vários outros tipos de alterações. Após o refinamento do modelo é gerado o código fonte e disponibilizado para o Servidor Cordova;
4. **Publique:** com o Cordova CLI é possível compilar o projeto e disponibilizar o APK, para instalação e execução no próprio dispositivo móvel. Outra forma de executar é por aplicativos como Phonegap Desktop e Cordova Browser que simulam a aplicação direto do servidor Cordova.
5. **Teste:** com a interface podendo ser simulada diretamente no dispositivo móvel é bem mais fácil realizar testes de usabilidade, produzindo uma experiência completa e precisa do design.

Para dar suporte às etapas do processo, foi desenvolvido um conjunto de ferramentas que são apresentadas nas próximas seções desse capítulo.

4.2 O Estêncil

Para a criação do modelo foi necessário a definição do estêncil para Android. Com base nas orientações do Material Google Design, ou somente Material(GOOGLE, 2016). Os componentes do Material fornecem um ambiente de desenvolvimento confiável para aplicativos e sites em Android, iOS e na Web. Os componentes do Material estão em constante evolução, garantindo uma implementação consistente e perfeita em concordância aos padrões de desenvolvimento de *front-end* do Google. O Material é orientado por elementos de design baseados em impressão - como tipografia, grades, espaço, escala, cor e imagens - para criar hierarquia, significado e foco que imergem o usuário na experiência. O Material design adota ferramentas do campo do design de impressão, como grades de linha de base e modelos estruturais, incentivando a consistência entre ambientes, repetindo elementos visuais, grades estruturais e espaçamento entre plataformas e tamanhos de tela. Esses *layouts* são dimensionados para se ajustarem a qualquer tamanho de tela, que simplifica o processo de criação de aplicativos escaláveis. O manual do Material Design é disponibilizado na internet pelo Google e apresenta os componentes típicos de uma interface de usuário (GOOGLE, 2016).

4.2.1 Principais Componentes do Material

O Material tem uma vasta quantidade de componentes visuais, comportamentais e ricos em movimentos. Para a produção do MOP foram escolhidos os componentes essenciais (Figura 13) para a geração de aplicativos, dentre eles estão barras de navegação, botões,

listas, elementos de entrada e seleção, controles, caixas de diálogo e alguns ícones de ação. A seguir é apresentado maiores informações sobre essas estruturas.

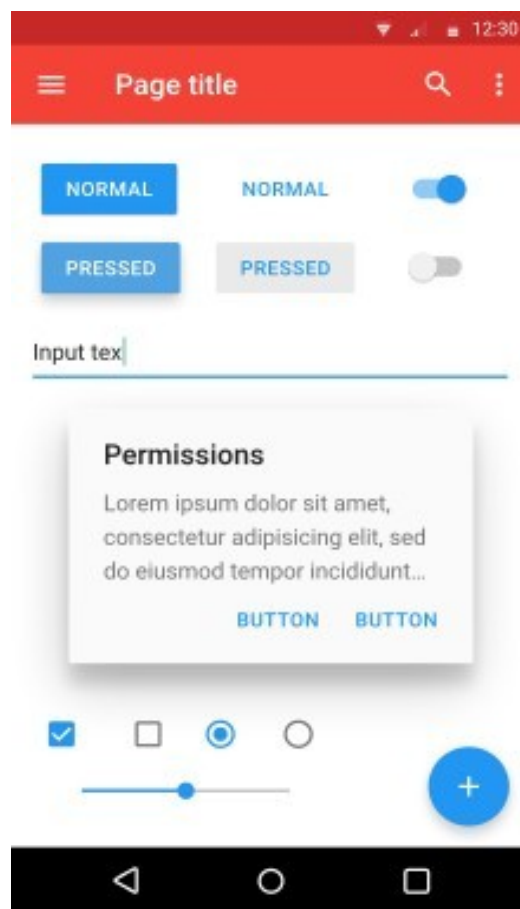


Figura 13 – Componentes essenciais.

A estrutura básica de navegação de um aplicativo é descrito na Figura 14. Essa estrutura possui uma barra superior permanente e um botão de ação flutuante, uma barra inferior opcional, que pode ter botões ação ou não, menus de navegação laterais que sobrepõem todos os outros elementos estruturais e uma área de conteúdo.

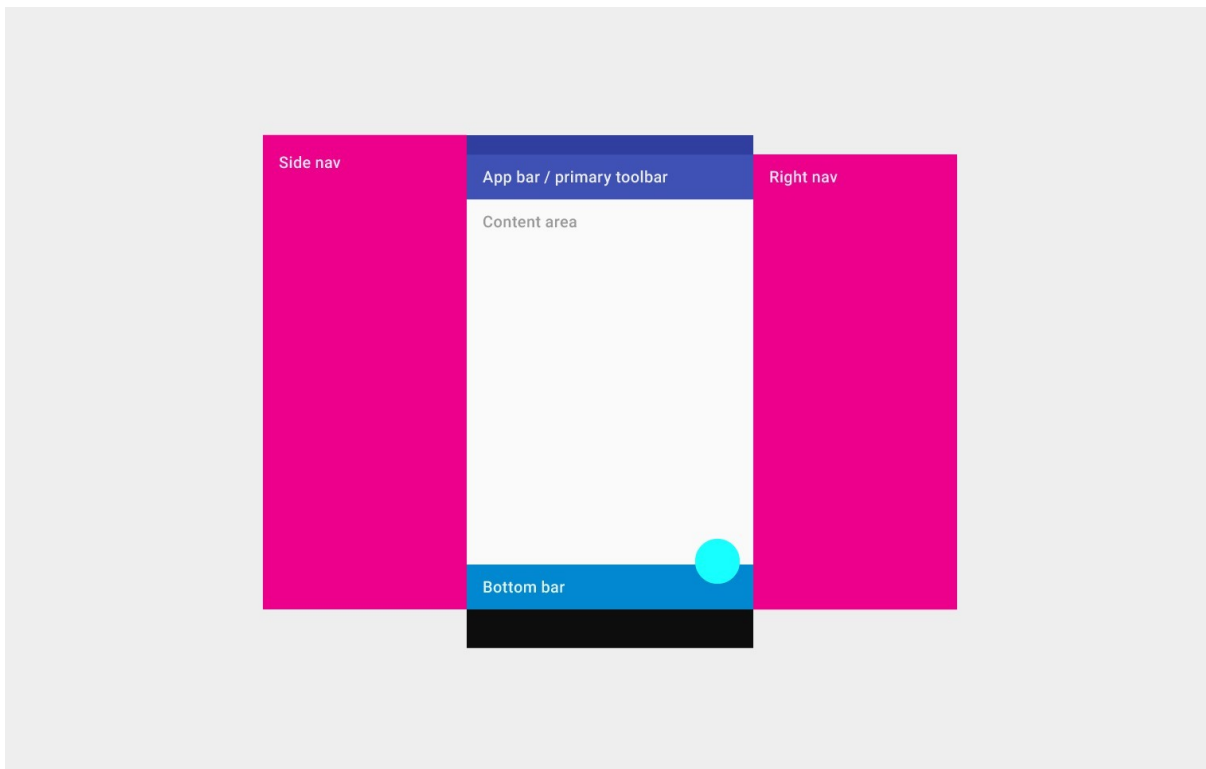


Figura 14 – Estrutura básica de navegação.

A barra superior, também conhecida como AppBar, consiste em uma barra de ferramentas, que contém o título das aplicação e, geralmente, contém ícones de ações para abrir outros menus, como o de navegação lateral.

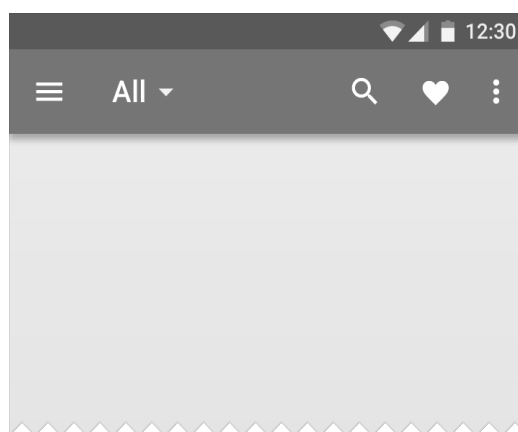


Figura 15 – Barra superior (AppBar).

A barra inferior de navegação é composta por itens na forma de etiquetas, ícones ou ambos. Ela fornece navegação rápida entre as telas do aplicativo. Além disso, pode conter de três a cinco ícones de navegação, onde cada um leva a um destino diferente.

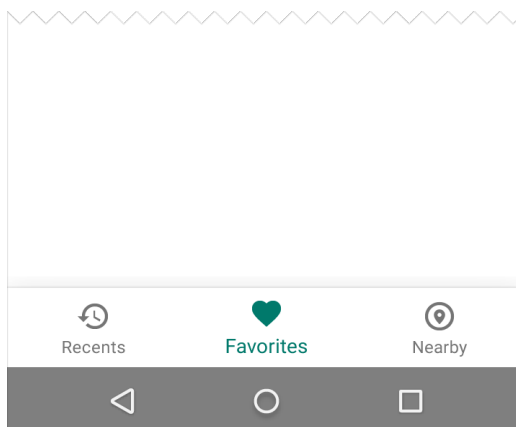


Figura 16 – Barra inferior de navegação (BottomBar).

Um botão é definido em texto e/ou uma imagem que comunica claramente qual ação ocorrerá quando o usuário clicar ou tocar nele. O Material fornece 3 tipos de botões. Dois dos botões tem o formato retangular, são eles o *Raised* e *Flat*, enquanto o FAB, *Flat Action Button*, é no formato circular com ícone dentro. O FAB pode ser usado para ações como, chamar outras telas ou caixas de diálogo.

- **FAB:** é um botão flutuante no formato circular com um ícone de ação em seu centro. Pode ser encontrado numa versão menor chamada de miniFAB;
- **Raised:** é um botão colorido, no formato retangular que paira sobre a interface;
- **Flat:** é um botão que apesar do formato retangular não contém bordas, reage aos toques preenchendo com cor. Geralmente utilizado em cartões e caixas de diálogo.

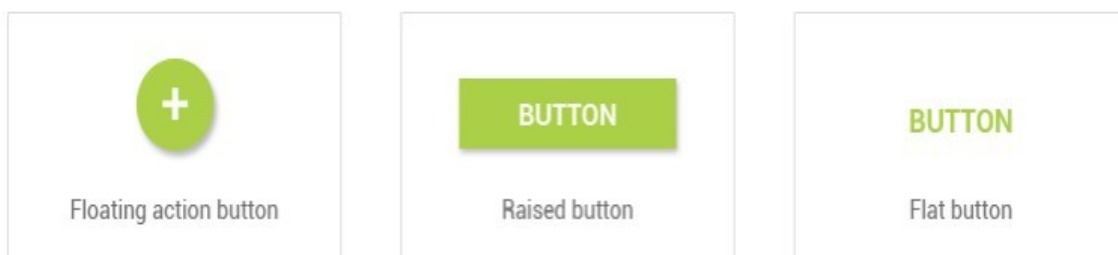


Figura 17 – Tipos de botões do Material.

Um cartão (Figura 18) é um componente que serve como um ponto de entrada para informações mais detalhadas. Os cartões podem conter uma foto, texto e um link sobre um único assunto. Eles podem exibir conteúdo contendo elementos de tamanho variável, como fotos com legendas de comprimento variável.

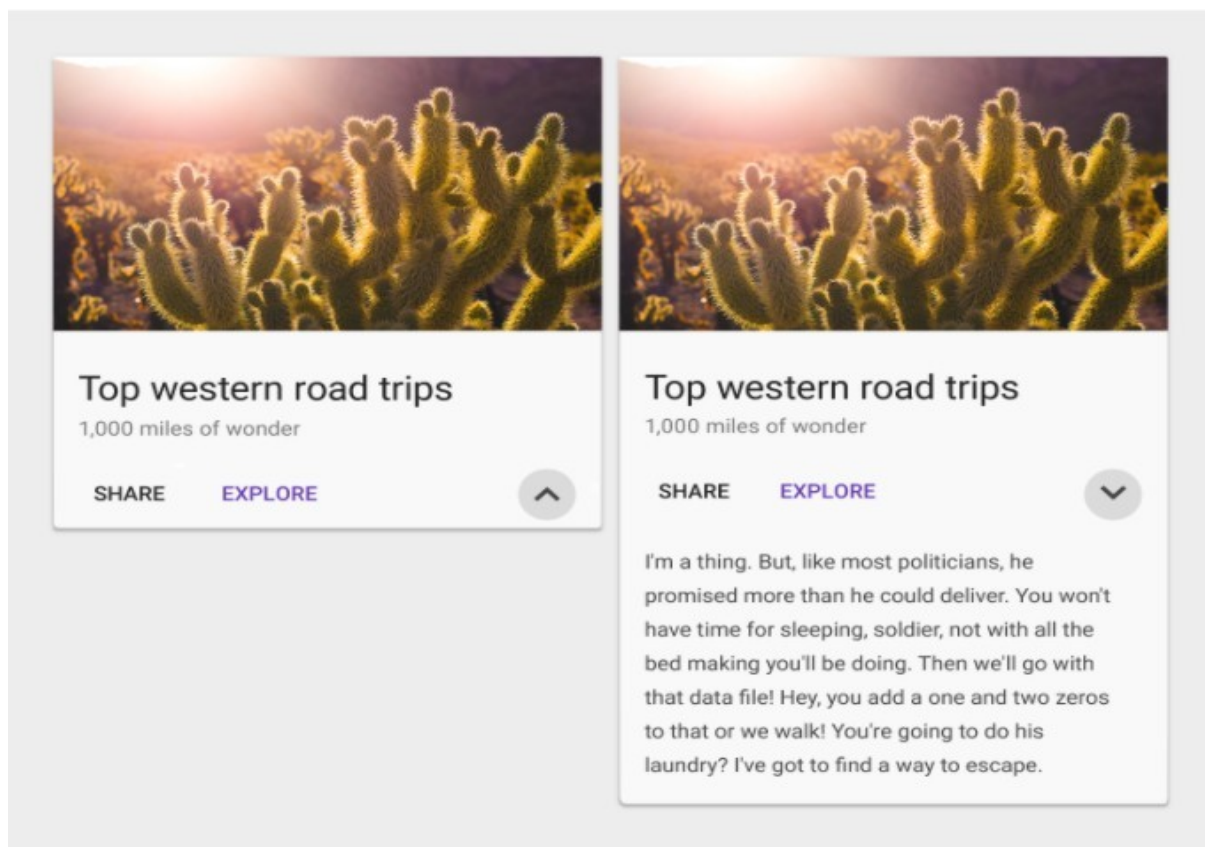


Figura 18 – Exemplo Cartão do Material.

As caixas de diálogo (Figura 19) avisam aos usuários sobre as informações críticas, exigindo que os usuários façam decisões. A aparência da caixa de diálogo força o usuário a interromper sua tarefa atual e se focar no conteúdo do diálogo.

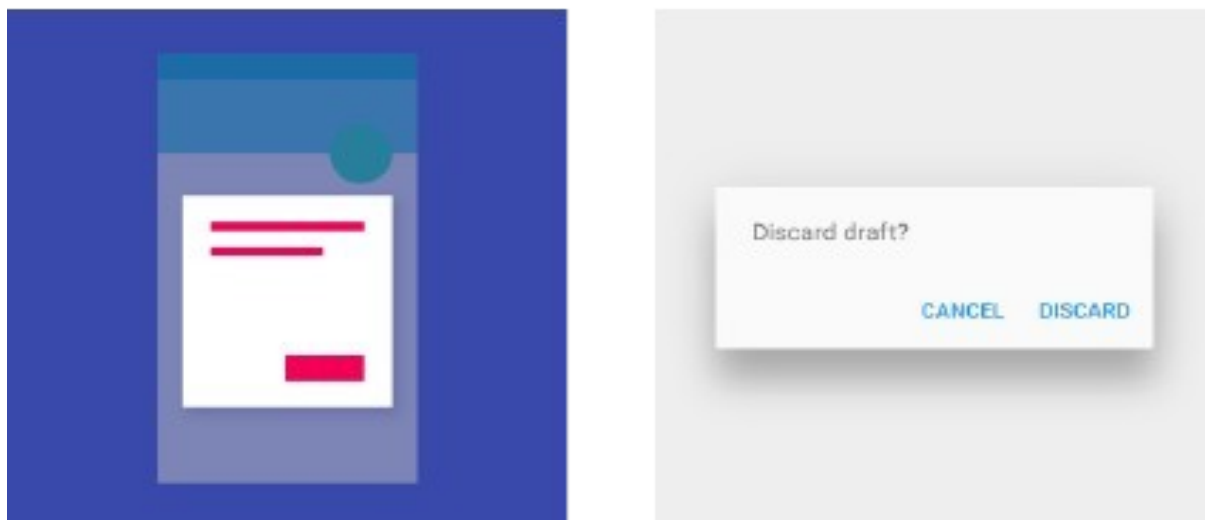


Figura 19 – Exemplo de Caixa de Diálogo do Material.

As listas, com exemplo ilustrado pela Figura 20, são compostas por um conjunto de itens do mesmo tipo dispostos em um arranjo vertical. Alguns tipos de listas contêm controles de ação. Os controles de lista são ícones que aparecem à esquerda ou à direita do

texto da lista. Eles indicam o estado de um item de lista, informações sobre um item ou servem como uma ação relacionada a este item.



Figura 20 – Exemplo de lista com controles de ações.

Existem ainda diversos componentes para entrada de dados e controle (Figura 21), que são utilizados principalmente em formulários. Abaixo segue a lista:

- **Campos de texto:** normalmente são inseridos em formulários, mas podem aparecer em outros locais, como caixas de diálogo e pesquisa. Esse tipo de campo permite a entrada de dados por texto e podem ser inseridos em linha única, com ou sem rolagem ou multi-linha, e, ainda pode ter um ícone;
- **Pickers:** são selecionadores de data e hora. Utilizam uma caixa de diálogo para selecionar uma única data ou hora.
- **Checkboxes:** permite que o usuário selecione várias opções de um conjunto;
- **Radio:** botões tipo Radio permitem que o usuário selecione uma opção dentro de um conjunto;
- **Switch:** é um botão de estado que permite alterar o estado de uma única opção de configuração;
- **Slider:** Os controles deslizantes permitem que os usuários selecionem um valor dentro de um intervalo de valores movendo o polegar.

A Figura 21 ilustra os componentes de entrada de dados e controle.

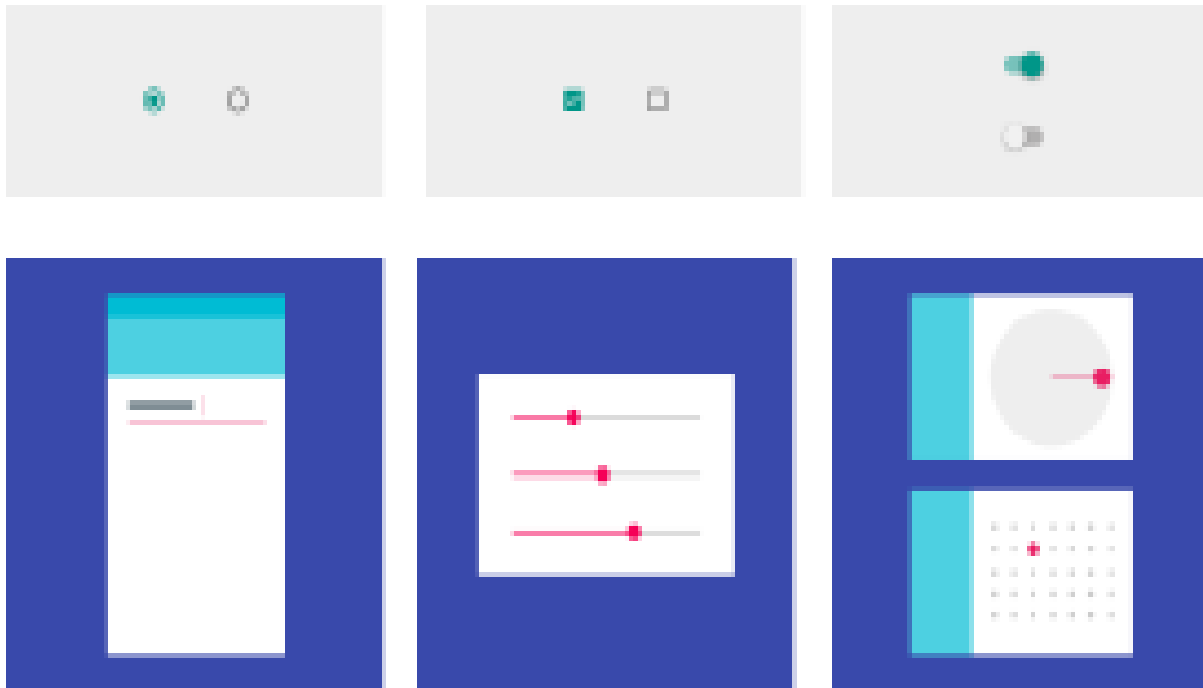


Figura 21 – Componentes de Entrada e Seleção.

4.2.2 Especificação do Estêncil

Na seção anterior foi apresentado o detalhamento dos principais componentes do Material Design, nesta seção é apresentado os componentes do estêncil que são utilizados para a confecção dos esboços das interfaces de usuário. Os elementos desse estêncil foram ajustados para facilitar a sua identificação pelo sistema de visão computacional. Abaixo segue a lista de elementos que compõem o estêncil.

- Header
- Footer
- Content
- Dialog
- Button
- Input text
- List
- Controls

O protótipo do estêncil foi projetado em acetato (Figura 22) e é formado pelos principais elementos da biblioteca de componentes do Material Design. Seu uso pode ser otimizado com a utilização de cadernos de layout (Figura 23). Esses cadernos são blocos de notas com silhuetas das telas, onde o projetista deve esboçar seu protótipo de interface utilizando estêncil.

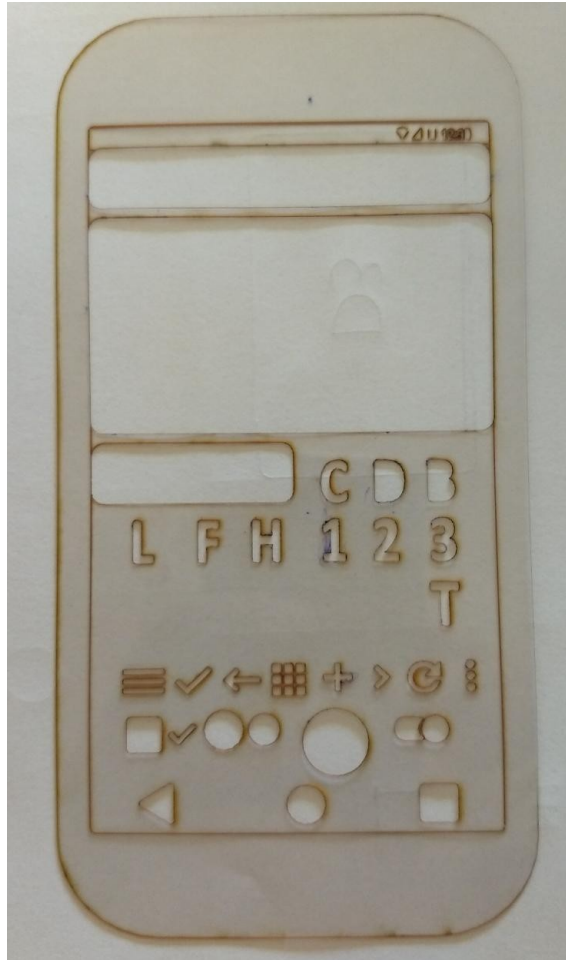


Figura 22 – Estêncil Físico.

A utilização do estêncil oferece velocidade ao desenvolvimento dos protótipos. A padronização dos ícones e regras de design proporcionam ao aplicativo de reconhecimento de objetos uma melhor interpretação do layout, facilitando o processo de prototipação.

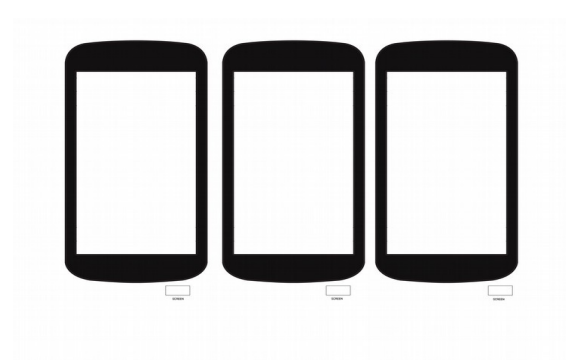


Figura 23 – Folha do caderno de layout.

Os componentes do estêncil podem ser apresentados de diversas formas, por exemplo, existem 3 formas de cabeçalho (Header), 3 formas de ilustrar os cartões (CARD), e assim sucessivamente. Durante a confecção do esboço de tela o projetista deve identificar o componente utilizando o formato correspondente, bem como uma letra seguida de um

número, exceto os controles que possuem forma já definida. Para criar uma lista composta apenas por texto, é necessário desenhar o forma com ilustra a Figura 24.



Figura 24 – Recorte da lista simples.

Dessa forma o projetista poderá criar uma enorme variedade de telas esboçando apenas formas letras e números. As variações de todos os componentes disponíveis para o esboço de tela são definidas logo abaixo.

O HEADER é uma barra de cabeçalho que contém um texto com o título do aplicativo. Também poderá conter dois tipos de ícones: o que abre o SIDEBAR e o de MORE que abre um menu suspenso de opções. Esse componente pode ser esboçado de 3 formas diferentes:

- H1 : simples, apenas com o título;
- H2 : contém o ícone que liga ao NAVSIDE;
- H3 : contém o ícone MORE que abre o menu suspenso.

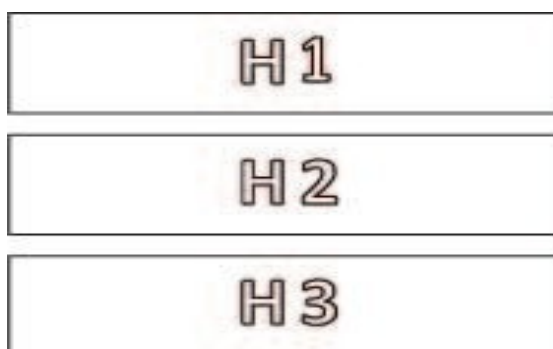


Figura 25 – Recorte do Header.

O FOOTER é inserido na parte inferior de uma tela que podem acomodar vários tipos de conteúdo. Tem as mesmas opções do HEADER, exceto o ícone para o SIDEBAR. Poderá ser usado como BOTTOMBAR, uma espécie de barra de ferramentas na parte inferior da tela.

- **F1**: simples, apenas com o título;
- **F2**: contém o ícone MORE que abre o menu suspenso;

- **F3:** BOTTOMBAR, com alguns ícones ou TABS.



Figura 26 – Corte do Rodapé

A forma do CARD no estêncil é uma quadrilátero, geralmente um retângulo com um C no interior. O CARD pode conter até cinco blocos, sendo eles, título, imagem, vídeo, texto e ações. No estêncil o CARD pode ser expresso 3 formas diferentes:

- **C1:** Simples, apenas o texto;
- **C2:** Título, texto e Ações;
- **C3:** Título, Imagem ou Vídeo, Texto e Ações.



Figura 27 – Corte do CARD

A caixa de diálogo ou DIALOG permite a verificação das ações do usuário, entrada de dados simples e alertas para fornecer informações adicionais aos usuários. A DIALOG aparece após ação de um botão e é executado por uma uma função javascript. É composto por título, conteúdo de texto e ações. O estêncil apresenta duas formas de DIALOG, a caixa de alerta e a caixa de diálogo simples.



Figura 28 – Corte do DIALOG

As listas apresentam vários itens de linha verticalmente como um único elemento contínuo. Podem conter texto, imagens e ações. Também podem ser esboçadas de 3 formas diferentes.

- **L1:** Lista simples, composta apenas por texto;
- **L2:** Lista com avatar;
- **L3:** Lista com avatar e action.



Figura 29 – Recorte da Lista

No estêncil os botões podem ser representados de três formas, duas retangulares e uma circular. O FAB é o botão no formato circular e tem seu próprio formato no definido estêncil. Abaixo os três tipos de botões definidos no estêncil.

- **B1:** Raised
- **B2:** Flat
- **FAB** (Formato próprio)



Figura 30 – Recorte dos Botões

Com o estêncil ainda é possível esboçar componentes de entrada e seleção de dados. Existem campos de entrada de textos, além de elementos de seleção de data e hora, botões tipo *radio*, *checkbox* e *switches*. Os recortes desses elementos são apresentados nas Figuras 31 e 32.



Figura 31 – Recorte dos Campos de Entrada de Texto.



Figura 32 – Recorte dos Componentes de Seleção.

4.3 App de Reconhecimento e Servidor Scilab

O aplicativo Android e o servidor Scilab juntos fazem a função do sistema de visão computacional. O aplicativo foi desenvolvido com o *framework* Apache Cordova e tem as funções de aquisição de imagens e conexão com o restante do sistema visão computacional. O servidor Scilab faz o restante do processo de reconhecimento, realizando procedimentos de segmentação; representação e descrição; reconhecimento e interpretação. Focando em técnicas de Casamento de Modelos (*Template Matching*), SURF(BAY; TUYTELAARS; GOOL, 2006) e SIFT(LOWE, 1999). Na metodologia proposta, o algoritmo é aplicado tela a tela. É feita a leitura da imagem a ser analisada, a detecção e extração de suas características e comparadas a cada um dos modelos existente na biblioteca. De acordo com os resultados é criado um arquivo de texto que conterá o **sample.mop**, contendo o modelo em formato XML, que deverá ser importado pelo Editor de Modelos onde é possível refinamento. O código abaixo exemplifica o a saída do arquivo gerado pelo sistema de visão computacional.

```

1  <? xmlversion= " 1.0 "encoding= " UTF-8 " ? >
2  <mop:App xmi:version= " 2.0 "xmlns:xmi= " http: // www.omg.org / XMI "
   xmlns:xsi= " http: // www.w3.org /2001/ XMLSchema-instance"xmlns:mop= "
   http: // www.mop.org / mop " >
3    <screen >
4      <uicomponent xsi:type= " mop:AppBar "/ >
5      <uicomponent xsi:type= " mop:List " >
6        <simplecomponent xsi:type= " mop:ListItem "/ >
7        <simplecomponent xsi:type= " mop:ListItem "/ >
8        <simplecomponent xsi:type= " mop:ListItem "/ >
9      </ uicomponent >
10     <uicomponent xsi:type= " mop:FAB " / >
11   < / screen >
12   <screen >
13     <uicomponent xsi:type= " mop:AppBar "/ >
14     <uicomponent xsi:type= " mop:Card "/ >
15   < / screen >
16   <screen >
17     <uicomponent xsi:type= " mop:AppBar "/ >
18     <uicomponent xsi:type= " mop:Card "/ >
19   < / screen >
20 </ mop:App >

```

O aplicativo oferece uma interface com um total de 4 telas. A tela inicial apresenta uma lista de cartões, onde cada cartão é um projeto do usuário. Essa tela contém um botão de ação flutuante que canto inferior direito que quando clicado abre outra tela contendo um formulário de criação de um novo projeto.

A tela do projeto segue o mesmo padrão da tela inicial, a diferença é que na lista

exposta, cada cartão é um esboço de feito a partir do estêncil. Para adicionar um novo esboço é necessário clicar no botão de ação, que permite a aquisição da esboço a partir da galeria de imagens do dispositivo ou capturar a foto com a câmera do próprio dispositivo. Na barra de título do aplicativo existe um botão para fazer o *upload* do projeto para o servidor Scilab, onde ocorrerá o reconhecimento das interfaces desenhadas no papel.

4.4 Editor Gráfico de Modelos

Existem diversas ferramentas que auxiliam na geração bancadas de modelagem. Para a criação do Editor de Modelos foi utilizado o Obeo Designer, é uma IDE baseada no Eclipse que permite criar rapidamente editores gráficos de modelos para domínio específicos. Assim como o Eclipse, o Obeo Designer possui perspectivas de desenvolvimento de acordo as atividades.

O projeto do Editor tem três projetos principais, sendo eles, o projeto de modelagem, o projeto de especificação de ponto de vista e o projeto de geração de código. Cada um desses projetos é implementado utilizando uma perspectiva.

4.4.1 Projeto de Modelagem

No projeto de modelagem (*mop.model*), foi gerado o metamodelo baseado no Modelo de Domínio do MOP. O metamodelo foi descrito de acordo com diagrama de classes ilustrado na figura 33. A especificação do metamodelo MOP é apresentada no Apêndice A.

O modelo de domínio define que uma aplicação (*app*) pode ter diversas telas (*screens*) , que são um conjunto de componentes de interface. Esses componentes podem ser simples ou grupo de componentes. Os principais os componentes são: barras, listas, *cards*, *dialogs*, botões, menus dentre outros. Para cada componente foram criado nós, para elementos que contêm outros elementos e *containers*, para elementos contêm elementos, inclusive *containers*. O objetivo principal desse projeto é gerar o código fonte em Java do metamodelo, que serve de ligação com a bancada de modelos. O Obeo gera automaticamente esse código a partir do diagrama de classes como ilustra a Figura 34, no caso do MOP foram criados outros dois projetos o **mop.model.edit** e o **mop.model.editor**.

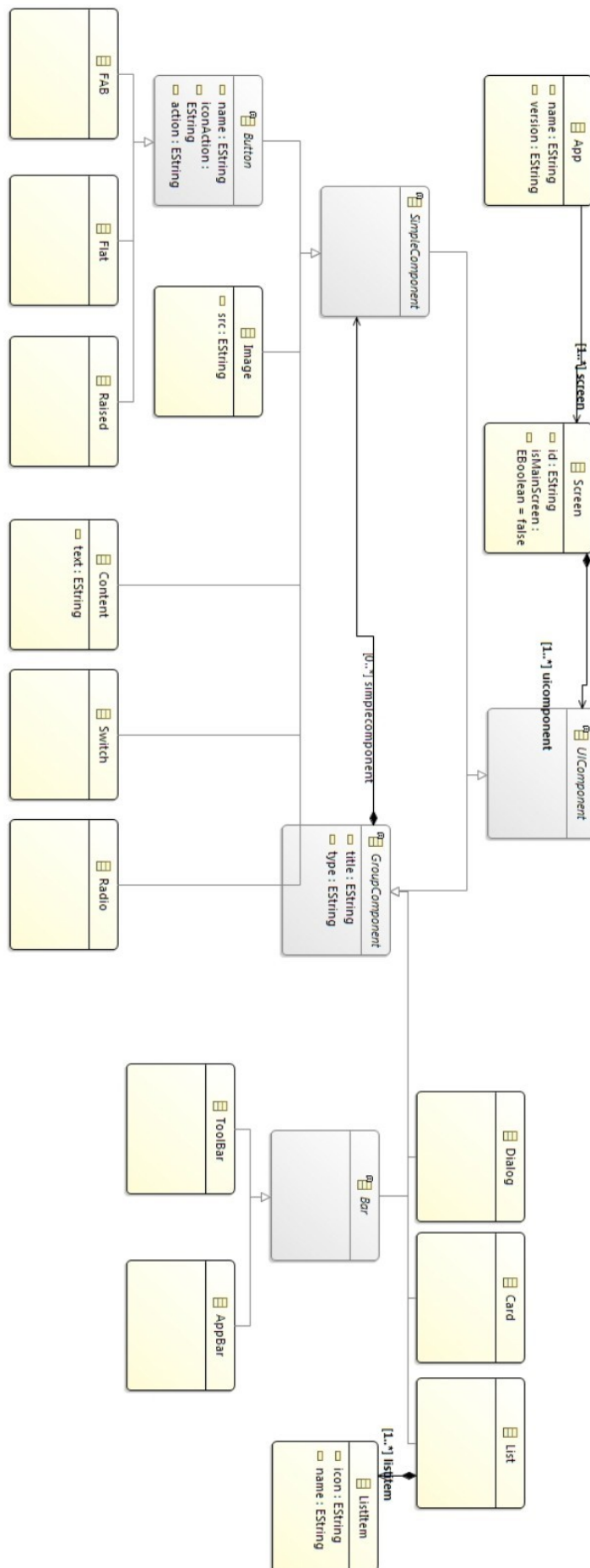


Figura 33 – Metamodelo Proposto.

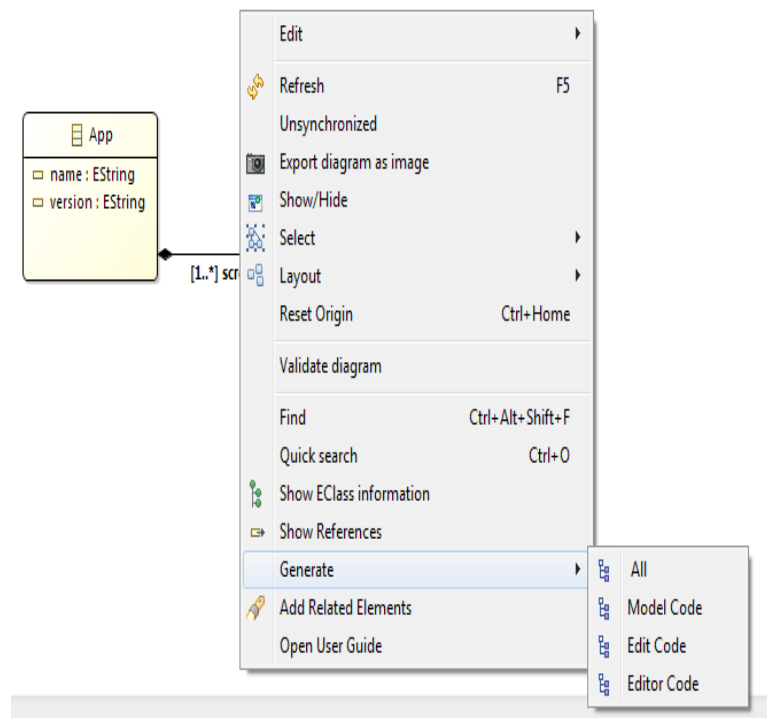


Figura 34 – Geração do Código Fonte do Metamodelo.

Para realizar testes do metamodelo é necessário criar um projeto de modelagem em uma nova instância do Obeo e criar um arquivo `sample.mop`. O arquivo pode ser editado de forma textual, em formato de árvore e a partir do paleta da bancada de modelos. O arquivo gerado pelo sistema de visão computacional, mencionado na Seção 4.3, deve ter essa mesma extensão e especificação.

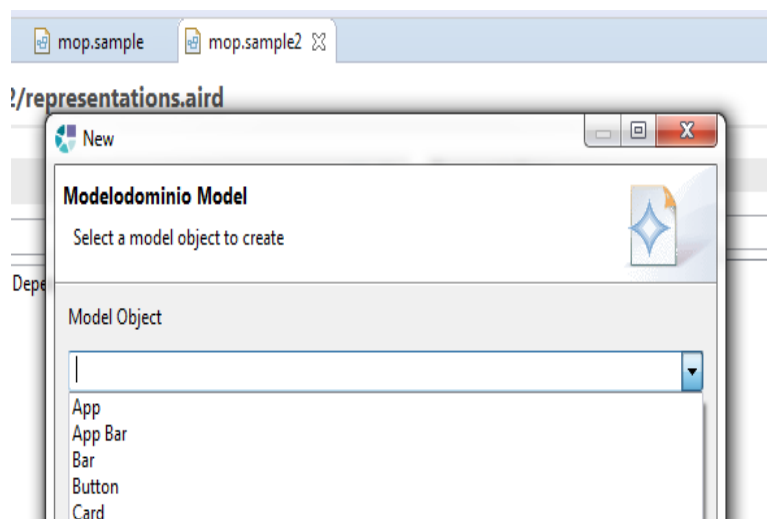


Figura 35 – Criação de novo arquivo de modelo

4.4.2 Projeto de Especificação de Ponto de Vista

A bancada de modelagem foi desenvolvida a partir do projeto de especificação de ponto de vista, utilizando Sirius (Eclipse Foundation, 2017). Sirius é um projeto do

ambiente de desenvolvimento Eclipse que permite que o desenvolvedor crie facilmente seu próprio editor de modelagens gráficas aproveitando as tecnologias de Modelagem do Eclipse, incluindo EMF e GMF. Ele fornece uma ferramenta genérica para engenharia de arquitetura baseada em modelos que pode ser facilmente adaptada para atender necessidades específicas. Baseado em uma abordagem de ponto de vista, Sirius possibilita muitos recursos para lidar com arquiteturas complexas em domínios específicos. O Editor de Modelos roda como uma instância do Obeo Designer.

No **mop.design** foi criado o MOPViewPoint, que contém todas as definições do metamodelo em forma de um diagrama. No MOP Diagram foram criados o Nós, Containers e a Paleta de Edição. Cada elemento do diagrama tem propriedades relacionadas a classe do metamodelo correspondente, dentre as propriedades que podem ser definidas estão: id, classe de domínio, comportamento e estilo de formato. A Figura 36 demonstra o ambiente de edição do diagrama (*Sirius Specification Editor*).

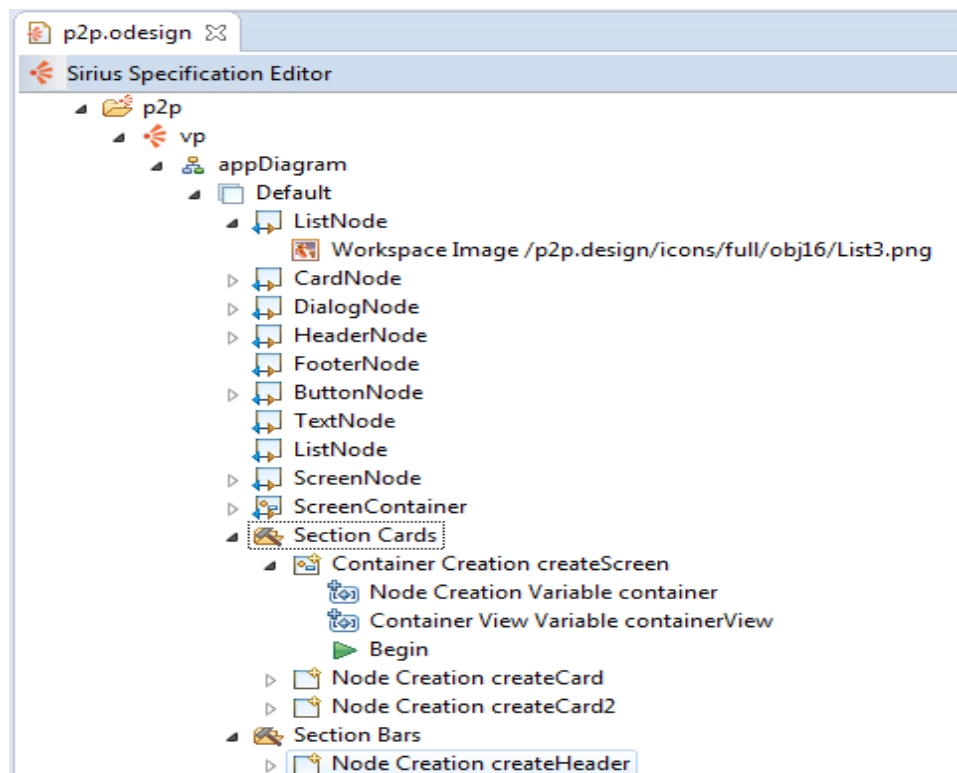


Figura 36 – Ilustração do Sirius Specification Editor

O projeto responsável pela geração de código foi desenvolvido com o Acceleo, que resultou no **mop.gencode.html**. O projeto é um gerador de código HTML baseado nas especificações metamodelo MOP, definido no **mop.model**, a partir de um arquivo **exemplo.mop**, resultado de um refinamento no Editor de Modelos. Durante o seu desenvolvimento foram seguidos os passos listados a seguir.

- Criação do *Acceleo Generation Project*;
- Criação do módulo de geração de código;

- Vinculação do módulo ao metamodelo MOP;
- Definição do *template* usando AQL;
- Criação do assistente **Generate MOP to HTML**

O assistente oferece um atalho para que o arquivo **index.html** seja criado apenas em um clique. Para acessar esse atalho é necessário clicar com o botão direito em arquivo *.mop e clicar em Generate MOP to HTML no menu Acceleo Model to Text.

4.4.3 Refinamento

Para iniciar um refinamento, é necessário criar um projeto de modelagem no Editor de Modelos e inserir o arquivo **sample.mop** resultante do reconhecimento de padrões na pasta do projeto.

O refinamento é feito no Editor de Modelos onde é possível inserir novos componentes, alterar textos e ações. Atualmente, o processo suporta um tipo de gesto, o clique único. Dentro do Editor de Modelos, é possível definir as transições entre telas, para isso são utilizados os botões. Cada botão pode ser ligado a uma tela e quando clicado alterna entre elas.

Ao final da edição o arquivo index.html pode ser gerado pelo assistente Generate MOP to HTML que deve ser exportado para o projeto Cordova pré-configurado com o Material Design Lite.

4.5 Simulação no Apache Cordova

O Apache Cordova (Apache Cordova, 2017) é um *framework* de desenvolvimento móvel de código aberto que permite utilizar tecnologias web padrão - HTML5, CSS3 e JavaScript para desenvolvimento nos mais diversos tipos de plataformas. Os aplicativos são executados em *wrappers* segmentados para cada plataforma e dependem de APIs compatíveis com padrões para acessar as capacidades de cada dispositivo, como câmeras, sensores, dados, status da rede, dentre outros.

Dentro da proposta, o Apache Cordova tem a função de compilar ou simular o código gerado pelo editor de modelos.

O código gerado trata-se de um arquivo em HTML, index.html, que contém todas as telas e é inserido em um projeto pré-configurado. O projeto pode ser compilado pelo Cordova CLI, ou simulado utilizando o Cordova Browser.

Com aplicativo instalado no dispositivo móvel é possível realizar diversas avaliações, como verificar se a interface está de acordo com os requisitos do cliente ou identificar qualquer problemática que possa interferir na interatividade.

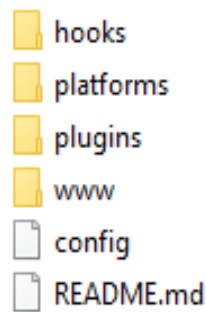


Figura 37 – Estrutura de Projeto Cordova Padrão.

4.5.1 Estrutura de pastas do projeto

O projeto pré-configurado traz a estrutura padrão do projeto em Cordova como é ilustrado na figura 37.

A lista a seguir descreve os principais elementos do projeto Cordova.

- **hooks:** são *scripts* usados para customizar os comandos Cordova. Os *hooks* são escritos em Javascript. Por exemplo, pode-se usar o hook *before-platform-add* para ser executado antes de adicionar um plataforma (*cordova platform add*);
- **platforms:** para desenvolver aplicativos Cordova, é necessário instalar SDKs para cada plataforma. Esta pasta contém as configurações necessárias para compilar projeto para a plataforma destino, nesse está configurado para compilar em Android;
- **plugins:** um *plugin* é um *script* utilizado acessar funções da plataforma destino, permitindo que o aplicativo use recursos do dispositivos além do que está disponível para aplicativos da web pura. Esta pasta contém os principais *plugins* para acessar as funções nativas de um dispositivo Android;
- **www:** é nessa pasta que será inserido o index.html. Será detalhada a seguir;
- **config.xml:** é o arquivo de configuração do Cordova, nele estão as todas as informações do projeto;
- **README.md:** esse é um arquivo padrão contendo informações sobre o *framework*.

A pasta **www** (Figura 38) contém os todos os elementos de interface e interatividade do protótipo. A estrutura é composta pelas pastas: *css*, *js*, *lib* e o arquivo *index.html* como pode ser visualizado na estrutura da figura a seguir.

- **css:** essa pasta contém o arquivo *css* padrão do Cordova;
- **js:** aqui está o arquivo *my-app.js* padrão do Cordova
- **lib:** dentro dessa pasta está a estrutura do Material Design, contém todos os tipos de *css*, o arquivo *material.min.js* e uma pasta de imagens contendo todos



Figura 38 – Estrutura da Pasta www.

os ícones de ação utilizados pelo MOP.

4.5.2 Explicando a interatividade

O arquivo *index.html* gerado pelo Editor de Modelos contém todas as interfaces e suas devidas interações. O código das telas é apresentado dentro da tag *<main>*, todas essas tags são da classe *screen* e têm um id próprio que será utilizado para alterar na visualização. Somente uma tag *<main>* pode assumir a classe *screen_active* por vez e seu conteúdo será apresentado na tela do dispositivo. O código abaixo ilustra a composição do código das telas.

```

1 <main id= " screen1 " class= " mdl-layout__contentscreen " >
2 <!--Conteúdo da Tela 1 -->
3 </main >
4
5 <main id= " screen2 " class= " mdl-layout__contentscreen_active " >
6 <!--Conteúdo da Tela 2 -->
7 <!--A Tela 2 esta ativa -->
8 </main >

```

Cada ação definida no Editor de Modelos é gerenciado pela uma função javascript. O script alterna entre as telas de acordo com os cliques nos botões ou links. A função muda o ID de cada *<main>* para indicar qual tela deve ser apresentada. Abaixo é apresentado o código da função.

```

1 function alternaTela ( id ){
2     var val;
3     var a = document.getElementsByClassName ( ' screen ' );
4     for ( val of a ) {
5         val.className= " screen " ;
6     }
7     var NAME = document.getElementById ( id ) ;
8     NAME.className= " screenscreen-active " ;
9 }

```

4.5.3 Simulação ou Compilação

Com o projeto pré-configurado disponível é possível gerar o APK, utilizando o Cordova CLI, ou simular no dispositivo ou em um navegador web, com o uso do Phonegap Desktop.

O PhoneGap Desktop juntamente com o Phonegap Developer App facilitam bastante os testes, com eles é possível ver rapidamente um protótipo em ação. O Phonegap Desktop é uma espécie de servidor que disponibiliza o projeto na rede. O PhoneGap Developer App é uma App que pode ser instalado no aparelho e vincula o projeto ao dispositivo. Ele automaticamente traz a versão mais recente do código e roda como se fosse a App final. Nesse caso para testar uma versão nova do protótipo não é preciso reinstalar o APK.

A partir desse ponto a interface pode ser simulada diretamente no dispositivo móvel sendo bem mais fácil realizar testes de usabilidade, produzindo uma experiência completa e precisa do design.

4.6 Conclusão

Esse capítulo apresentou como foram criadas as ferramentas que utilizadas no processo de prototipação MOP, bem como as funções que cada um tem dentro do processo desde o reconhecimento do *sketch* a simulação no Cordova, passando pelo refinamento e a geração de código.

Apesar do roteiro ter sido apresentado na Seção 4.1 e na Figura 12 abaixo segue uma lista das etapas do processo linear de prototipação, levando em consideração as ferramentas apresentadas nas seções desse capítulo:

1. o primeiro é passo é utilizar o estêncil e o caderno de layout para esboçar as telas, com base nos componentes definidos na seção 4.2.2
2. com o aplicativo digitalizar o esboço através da câmera do dispositivo;
3. exportar as imagens para o sistema de visão computacional, onde o algoritmo reconhece os componentes e gera o arquivo `sample.mop`;
4. No Editor de Modelos cria-se uma projeto do tipo *Modeling Project*, onde é feito o refinamento e a geração do arquivo `index.html`;
5. finalmente o arquivo `index.html` é inserido no projeto pré-configurado com o Material Design Lite e simulado a partir do Servidor Cordova, onde são realizados os testes.

Na capítulo a seguir é apresentado um estudo de caso que traz o passo a passo da criação de um protótipo utilizando o processo de prototipação MOP.

5 Estudo de Caso

Neste capítulo, é apresentado um estudo de caso que ilustra o desenvolvimento de um protótipo utilizando o processo MOP. O aplicativo teste escolhido para o desenvolvimento deste estudo de caso é um simples cadastro de lembretes, esboçado em 3 telas. A primeira tela contém a lista de lembretes e um botão de ação flutuante, a segunda uma caixa de diálogo para cadastro de novos lembretes e a outra com o cartão que exibe a descrição do lembrete. As seções a seguir apresentam os passos para a criação desse protótipo.

5.1 Desenhe

Utilizando o estêncil e o caderno de layout foi feito o esboço das telas de acordo com componentes da DSL. A Figura 39 ilustra os esboços em papel que serviram de base para a criação do protótipo.

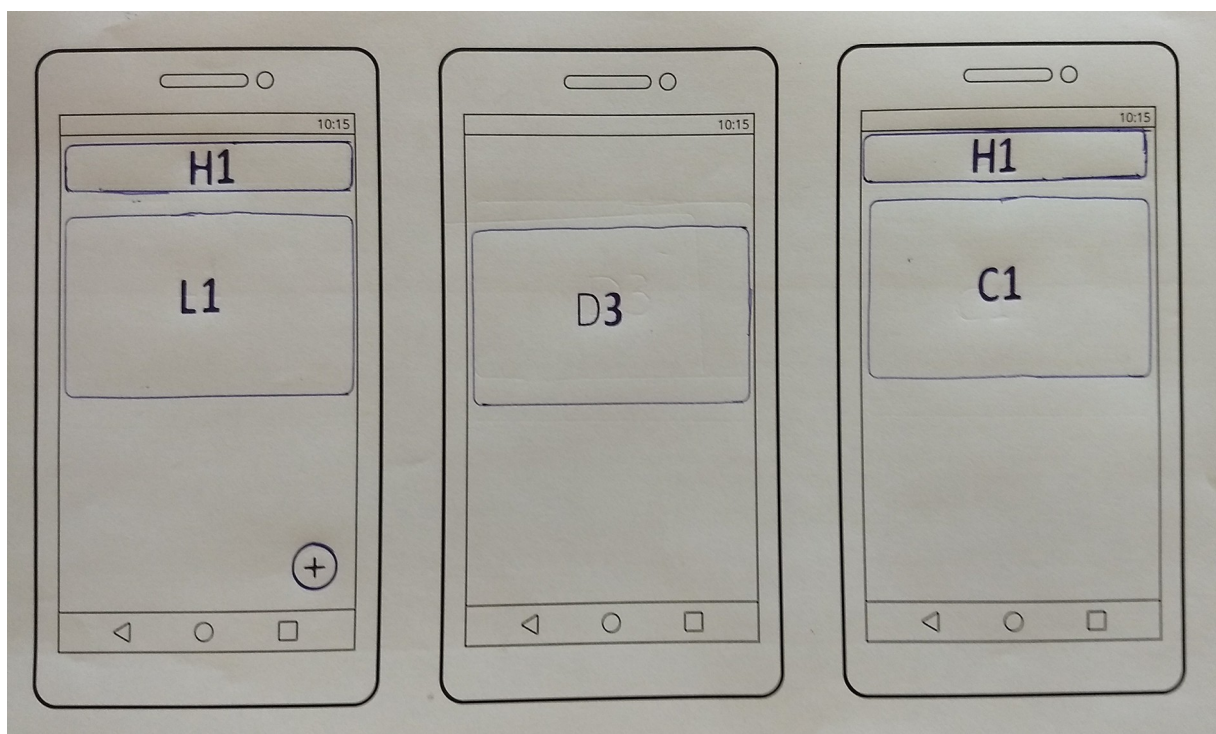


Figura 39 – Protótipo em papel.

5.2 Digitalize

As telas foram digitalizadas uma a uma com auxílio do aplicativo de reconhecimento através da câmera do dispositivo. A primeira imagem digitalizada é correspondente a tela inicial. As imagens foram passadas para o sistema de visão computacional que

processou cada imagem, identificando os contornos e o reconhecimento ótico de caracteres e comparou com a biblioteca de modelos e as regras de design. Na primeira imagem as formas reconhecidas foram: a tela em si, já impressa no caderno de layout, a barra superior (H1), a lista simples (L1) e o botão flutuante de ação. Na imagem da segunda tela foi reconhecido a caixa de diálogo com título e 3 campos (D3). Os objetos reconhecidos na terceira imagem foram a barra superior e o cartão simples (C1). Após o reconhecimento foi gerado o arquivo `sample.mop`, com toda estrutura do modelo para que no Editor seja possível refinar o protótipo. O arquivo `sample.mop` resultado do reconhecimento do layout contém o código logo abaixo.

```

1  <? xmlversion= " 1.0 "encoding= " UTF-8 " ? >
2  <mop:App xmi:version= " 2.0 "xmlns:xmi= " http: // www.omg.org / XMI "
   xmlns:xsi= " http: // www.w3.org /2001/ XMLSchema-instance"xmlns:mop= "
   http: // www.mop.org / mop " >
3    <screen >
4      <uicomponent xsi:type= " mop:AppBar " / >
5      <uicomponent xsi:type= " mop:List " >
6        <simplecomponent xsi:type= " mop:ListItem "/ >
7        <simplecomponent xsi:type= " mop:ListItem "/ >
8        <simplecomponent xsi:type= " mop:ListItem "/ >
9      < / uicomponent >
10     <uicomponent xsi:type= " mop:FAB " / >
11   </ screen >
12   <screen >
13     <uicomponent xsi:type= " mop:AppBar " / >
14     <uicomponent xsi:type= " mop:Card " / >
15   </ screen >
16   <screen >
17     <uicomponent xsi:type= " mop:AppBar " / >
18     <uicomponent xsi:type= " mop:Card " / >
19   </ screen >
20 </ mop:App >

```

5.3 Refine

No Editor de Modelos, foi criado um projeto do tipo *Modeling Project*, que foi nomeado `mop.sample.lembretes`, onde o foi inserido o arquivo `sample.mop`. Com a estrutura do projeto de refinamento montado, foram alteradas as *labels*, os tipos de ícones, e inseridos botões de ações nos *cards* e *dialogs*, utilizando a seção de propriedades da bancada de modelagem. Na Tela 1 foi alterado o título do app na barra superior, o nomes dos itens e o ícone da lista passou a ser o calendário (*event*). Também foram inseridas ações nos itens da lista para acessar outras telas e no botão para abrir a caixa de diálogo. Na Tela 2 foram alterada as *labels* dos campos de texto. Na Tela 3 a alteração foi apenas no conteúdo

textual do cartão. Na bancada de modelagem utilizando a paleta de ferramentas foi criada uma quarta tela no mesmo formato da terceira, apenas o conteúdo do cartão diferente. A tela pode ser acessada com um clique no segundo item da lista. A Figura 40 apresenta a bancada de modelagem.

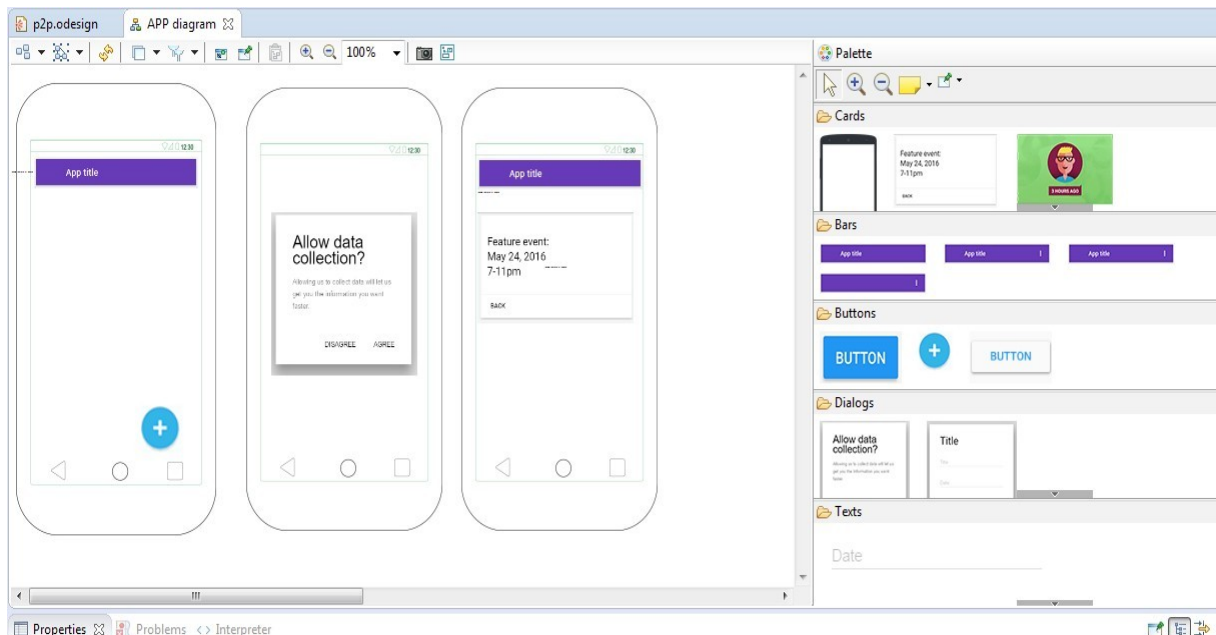


Figura 40 – Refinamento.

Após todas as edições realizadas na fase de refinamento o código **sample.mop** também foi atualizado e seu código é apresentado logo abaixo.

```

1  <? xmlversion= " 1.0 "encoding= " UTF-8 " ? >
2  <mop:App xmi:version= " 2.0 "xmlns:xmi= " http: // www.omg.org / XMI "
   xmlns:xsi= " http: // www.w3.org / 2001/ XMLSchema-instance"xmlns:mop= "
   http: // www.mop.org / mop " >
3    <screen id= " screen1 isMainScreen= " true " >
4      <uicomponent xsi:type= " mop:AppBar "title= " MOP"type= " H1 " />
5      <uicomponent xsi:type= " mop:List " >
6        <simplecomponent xsi:type= " mop:ListItem "icon= " event "name= "
           Lembrete 1" action= " screen3 "/ >
7        <simplecomponent xsi:type= " mop:ListItem "icon= " event "name= "
           Lembrete 2 " action= " screen4 "/ >
8        <simplecomponent xsi:type= " mop:ListItem "icon= " event "name= "
           Lembrete 3" action= " null "/ >
9      </ uicomponent >
10     <uicomponent xsi:type= " mop:FAB "name= ""iconAction= " add "action=
           " " />
11   < / screen >
12   <screen id= " screen3 " >
13     <uicomponent xsi:type= " mop:AppBar "title= " MOP"type= " H1 " />
14     <uicomponent xsi:type= " mop:Card "title= " "type= " C1 " >

```

```
15     <simplecomponent xsi:type= " mop:Content "text= " Evento1 :  
16         31/07/2017 09:00 "/ >  
17     < / uicomponent >  
18 </ screen >  
19 <screen id= " screen4 " >  
20     <uicomponent xsi:type= " mop:AppBar "title= " MOP type= " H1 " / >  
21     <uicomponent xsi:type= " mop:Card "title= "" type= " C1 " >  
22         <simplecomponent xsi:type= " mop:Content "text= " Evento2 :  
23             10/09/2017 18:00 "/ >  
24     < / uicomponent >  
25 </ screen >  
26 </ mop:App >
```

5.4 Publique e teste

Após a inferência do layout e o refinamento no Editor de Modelo o código foi gerado a partir do assistente de Generate MOP to HTML e salvo na pasta src-gen com o nome de index.html. O arquivo HTML foi inserido no projeto pré-configurado no servidor Cordova e simulado no próprio dispositivo utilizando o aplicativo Cordova Browser ou PhoneGap Developer. A Figura 41 apresenta as telas geradas. O código do arquivo **index.html** pode ser visualizado no Apêndice B.

5.5 Conclusão

O protótipo foi gerado com êxito comprovando a eficácia do processo de prototipação MOP. Apesar do sucesso ainda é perceptível a necessidade melhorias no processo. A principal problemática na geração do protótipo foi o tempo de execução do algoritmo de reconhecimento das interfaces. O sistema de visão computacional deve ser otimizado para dar celeridade ao processo. O Editor de Modelos deve ser melhorado de forma que facilite o processo de edição do protótipo. O capítulo a seguir traz maiores informações sobre as lições aprendidas durante o desenvolvimento desta dissertação.

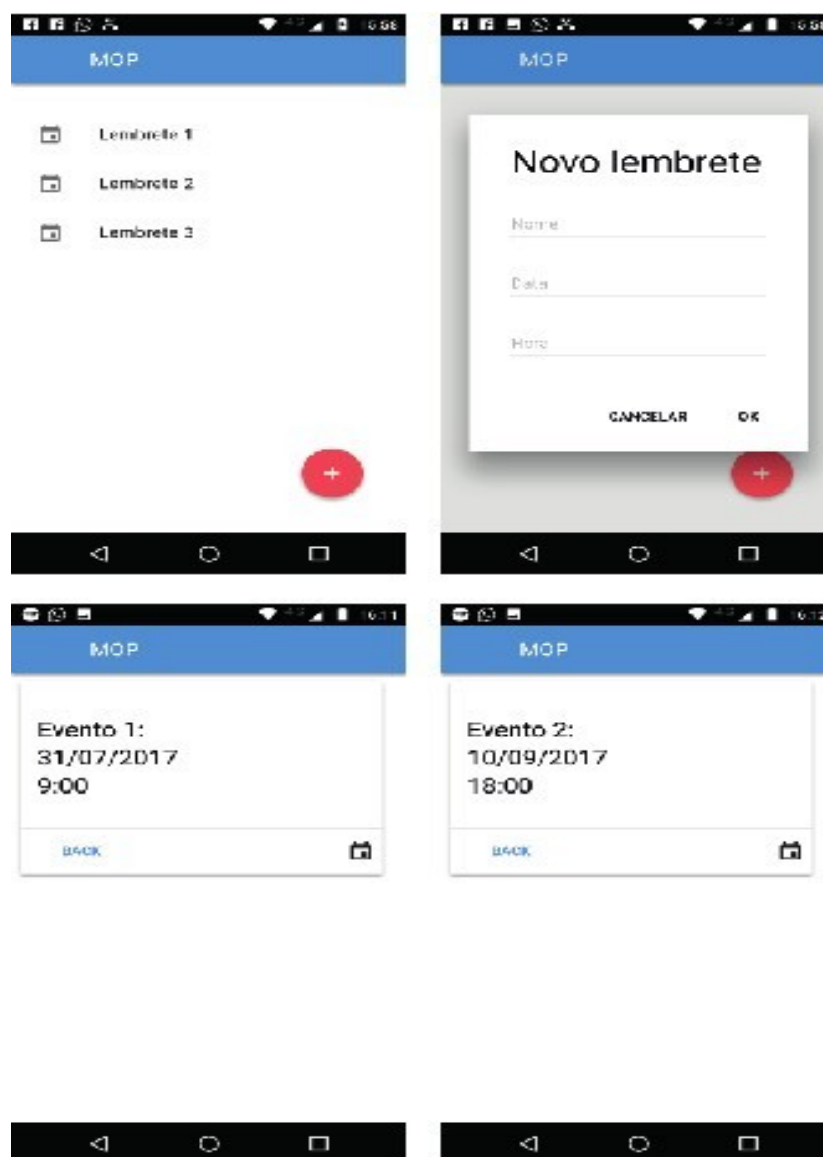


Figura 41 – Resultado.

6 Conclusão

Nesta dissertação, foi apresentado o processo de prototipação MOP, um estudo que adota o conceito de desenvolvimento orientado a modelos para gerar protótipos de aplicações móveis para Android a partir de protótipos em papel. O processo MOP pode ser utilizado como ferramenta no auxílio a fase de projeto de um processo de engenharia de software. Tem como principal objetivo facilitar produção de protótipo de alta fidelidade, utilizando pouca ou nenhuma codificação, a partir do papel.

O processo foi desenvolvido dentro do escopo da Engenharia de Software, utilizado conhecimentos das disciplinas de Processamento Digital de Imagens, Engenharia Orientada a Modelos e conceitos de Design de Interface e Experiência de Usuário. Durante a pesquisa pôde-se verificar que a prototipação uma forma eficiente de avaliar aspectos junto ao cliente, diminuindo custo de tempo e recursos antes de iniciar a codificação. O processo de prototipação MOP permite criar um protótipo precocemente mapeando modelos de *sketch* em telas de alta similaridade ao software final, assim agilizando a fase de projeto de software. O processo MOP evidencia o melhor das técnicas de prototipação de baixa e alta fidelidade. Como o projeto se inicia no papel, podem ser gerados rapidamente, não requerem habilidades específicas e necessitam de poucos recursos para serem esboçados. Após um breve refinamento o protótipo de alta fidelidade é disponibilizado, podendo ser realizados testes com um grau elevado de interatividade e a experiência do usuário bem mais realista. O protótipo gerado apresenta características de prototipação descartável e evolucionária, pois fica a critério da equipe de desenvolvimento a utilização ou não do código fonte a produção do software final.

Este estudo apresentou uma prova de conceito atestando a viabilidade do processo de prototipação MOP, apesar de sua efetividade, o processo ainda precisa de melhorias. Os componentes de interface são limitados. A ligação entre o Editor de Modelos e as outras ferramentas não está sendo feita de forma automatizada por ser executado em ambiente *desktop*. O algoritmo de reconhecimento da interface leva bastante tempo em sua execução. A geração de código está limitada a código HTML.

Sendo assim, em estudos futuros, pretende-se inserir todos os elementos de interface, suporte a gestos, otimizar o sistema de visão computacional com o o objetivo de acelerar o processo de extração de características dos esboços de tela e efetuar o reconhecimento de esboços feitos a mão livre, sem o uso de estêncil, ainda realizar um estudo minucioso do desempenho do sistema de visão computacional.

Devido ao Editor de Modelos ser uma ferramenta fora da *web*, a sua integração com as outras ferramentas não tem fluidez, o ideal é que em trabalhos futuros a bancada

de modelagem já possa ser disponibilizada em uma plataforma na internet. Transformar o processo MOP num serviço na web é premissa inicial que não pode ser alcançada por esta dissertação.

Como a geração de código ainda é limitada pretende-se efetuar integrações com outros *frameworks* de interfaces como o IONIC (Ionic, 2017), FRAMEWORK7 (Framework7, 2017), Materialize (Materialize, 2017) e geração de código nativo Android e iOS. Outro trabalho é otimizar o aplicativo para que possa realizar a prototipação diretamente no dispositivo, abrangendo reconhecimento de interface, refinamento e simulação em seu escopo.

Referências

- AMELLER, D. *Considering Non-Functional Requirements in Model-Driven Engineering*. Dissertação (Mestrado) — Llenguatges i Sistemes Informàtics (LSI), June 2009. Disponível em: <<http://upcommons.upc.edu/pfc/handle/2099.1/7192>>. Citado 3 vezes nas páginas 15, 43 e 44.
- Apache Cordova. *Cordova Documentation*. 2017. Disponível em: <<https://cordova.apache.org/docs/>>. Citado na página 84.
- ARAÚJO, S. A. d. *Casamento de padrões em imagens digitais livre de segmentação e invariante sob transformações de similaridade*. Tese (Doutorado) — Universidade de São Paulo, 2009. Citado 2 vezes nas páginas 15 e 53.
- BAHR, B. Rapid creation of sketch-based, native android prototypes with blended prototyping. In: *Mobile HCI 2013—Workshop on Prototyping to Support the Interaction Designing in Mobile Application Development*. [S.l.: s.n.], 2013. Citado na página 58.
- BAY, H.; TUYTELAARS, T.; GOOL, L. V. Surf: Speeded up robust features. *Computer vision—ECCV 2006*, Springer, p. 404–417, 2006. Citado na página 79.
- BÉZIVIN, J. et al. Bridging the ms/dsl tools and the eclipse modeling framework. In: *Proceedings of the International Workshop on Software Factories at OOPSLA*. [S.l.: s.n.], 2005. v. 5, p. 1–19. Citado na página 46.
- BIOLCHINI, J. et al. Systematic review in software engineering. *System Engineering and Computer Science Department COPPE/UFRJ, Technical Report ES*, v. 679, n. 05, p. 45, 2005. Citado 2 vezes nas páginas 15 e 55.
- BÖHM, S.; IGLER, B. A tool-based approach for structuring feedback for user interface evaluations of mobile applications. 2013. Citado na página 59.
- BOLCHINI, D.; PULIDO, D.; FAIOLA, A. Feature paper in screen prototyping: an agile technique to anticipate the mobile experience. *interactions*, ACM, v. 16, n. 4, p. 29–33, 2009. Citado 2 vezes nas páginas 62 e 63.
- BOOCH, G. et al. The unified modeling language. *Unix Review*, v. 14, n. 13, p. 5, 1996. Citado na página 43.
- BRADSKI, G. R. Real time face and object tracking as a component of a perceptual user interface. In: IEEE. *Applications of Computer Vision, 1998. WACV'98. Proceedings., Fourth IEEE Workshop on*. [S.l.], 1998. p. 214–219. Citado 2 vezes nas páginas 50 e 51.
- BROWN, T. et al. Design thinking: uma metodologia poderosa para decretar o fim das velhas ideias. *Rio de Janeiro: Elsevier*, 2010. Citado 3 vezes nas páginas 31, 32 e 33.
- CHEN, P. P.-S. The entity-relationship model—toward a unified view of data. *ACM Transactions on Database Systems (TODS)*, Acm, v. 1, n. 1, p. 9–36, 1976. Citado na página 43.

- COIFMAN, B. et al. A real-time computer vision system for vehicle tracking and traffic surveillance. *Transportation Research Part C: Emerging Technologies*, Elsevier, v. 6, n. 4, p. 271–288, 1998. Citado na página 51.
- CUELLO, J.; VITTONI, J. *Designing mobile apps*. [S.l.]: José Vittone, 2013. Citado 2 vezes nas páginas 17 e 64.
- Eclipse Foundation. *Sirius Documentation*. 2017. Disponível em: <<http://www.eclipse.org/sirius/doc/>>. Citado 2 vezes nas páginas 47 e 82.
- FILHO, O. M.; NETO, H. V. *Processamento digital de imagens*. [S.l.]: Brasport, 1999. Citado 3 vezes nas páginas 48, 49 e 51.
- FOWLER, M.; PARSONS, R. *DSL-Linguagens Específicas de Domínio*. [S.l.]: Porto Alegre: Bookman, 2013. Citado na página 45.
- Framework7. *Framework7 Documentation*. 2017. Disponível em: <<https://framework7.io/docs/>>. Citado na página 96.
- GIOVANNI, P. D.; SEBILLO, M.; VITIELLO, G. Integrating mobile design patterns within a user interface development environment. Citado na página 59.
- GOMES¹, S. L. et al. Reconhecimento ótico de caracteres para reconhecimento das sinalizações verticais das vias de trânsito. *Revista Sode bras*, v. 9, p. 101, 2014. Citado na página 50.
- GONZALEZ, R. C.; WOODS, R. E. *Processamento digital de imagens. tradução: Cristina Yamagami e leonardo piamonte*. [S.l.]: Pearson Prentice Hall, São Paulo, 2010. Citado 4 vezes nas páginas 15, 49, 51 e 52.
- GOOGLE. *Material Design*. 2016. Disponível em: <<https://material.google.com/>>. Citado na página 68.
- GOTHELF, J. *Lean UX: Applying lean principles to improve user experience*[S.l.]: "O'Reilly Media, Inc.", 2013. Citado 2 vezes nas páginas 31 e 32.
- GREMILLION, M. E. B.; ZIEBA, K. *The Guide to Prototyping Process & Fidelity*. 2016. <https://www.uxpin.com/studio/ebooks/>. Citado na página 35.
- HÖYSNIEMI, J.; HÄMÄLÄINEN, P.; TURKKI, L. Wizard of oz prototyping of computer vision based action games for children. In: ACM. *Proceedings of the 2004 conference on Interaction design and children: building a community*. [S.l.], 2004. p. 27–34. Citado na página 58.
- IGLER, B.; BRAUMANN, T.; BÖHM, S. A blackboard-like architecture for the development of evolving high fidelity mobile application prototypes. Citado na página 59.
- Ionic. *Ionic Documentation*. 2017. Disponível em: <<https://ionicframework.com/docs/>>. Citado na página 96.
- KIEFER, F.; HESS, S. Towards using context personas to support prototyping of mobile business apps. 2013. Citado na página 59.

- KLEPPE, A. G.; WARMER, J. B.; BAST, W. *MDA explained: the model driven architecture: practice and promise*. [S.l.]: Addison-Wesley Professional, 2003. Citado na página 46.
- LACHGAR, M.; ABDALI, A. Modeling and generating the user interface of mobile devices and web development with dsl. *Journal of Theoretical & Applied Information Technology*, v. 72, n. 1, 2015. Citado na página 63.
- LANDAY, J. A. Silk: sketching interfaces like crazy. In: ACM. *Conference companion on Human factors in computing systems*. [S.l.], 1996. p. 398–399. Citado na página 63.
- LARSEN, F. H. *From Sketches to Functional Prototypes-Extending Wireframe Sketcher with Prototype Generation*. Dissertação (Mestrado) — NTNU, 2014. Citado na página 46.
- LEICHTENSTERN, K.; ANDRÉ, E.; REHM, M. Using the hybrid simulation for early user evaluations of pervasive interactions. In: ACM. *Proceedings of the 6th Nordic Conference on Human-Computer Interaction: Extending Boundaries* [S.l.], 2010. p. 315–324. Citado na página 59.
- LIN, J. et al. Denim: finding a tighter fit between tools and practice for web site design. In: ACM. *Proceedings of the SIGCHI conference on Human Factors in Computing Systems*. [S.l.], 2000. p. 510–517. Citado 2 vezes nas páginas 31 e 63.
- LOWE, D. G. Object recognition from local scale-invariant features. In: IEEE. *Computer vision, 1999. The proceedings of the seventh IEEE international conference on*. [S.l.], 1999. v. 2, p. 1150–1157. Citado na página 79.
- Marvel. *POP - Prototyping on paper*. 2017. Disponível em: <<https://popapp.in>>. Citado 2 vezes nas páginas 25 e 62.
- Materialize. *Materialize Documentation*. 2017. Disponível em: <<http://materializecss.com/>>. Citado na página 96.
- MATHWORKS, I. T. *Computer Vision System Toolbox™ User's Guide*. [S.l.]: MathWorks, 2017. Citado na página 53.
- MATHWORKS, I. T. *Image Processing Toolbox™ User's Guide*. [S.l.]: MathWorks, 2017. Citado na página 53.
- MENS, T.; GORP, P. V. A taxonomy of model transformation. *Electronic Notes in Theoretical Computer Science*, Elsevier, v. 152, p. 125–142, 2006. Citado na página 46.
- MILLER, J. et al. Mda guide. *Object Management Group*, 2003. Citado 3 vezes nas páginas 43, 44 e 45.
- MUSSET, J. et al. Acceleo user guide. See also <http://acceleo.org/doc/obeo/en/acceleo-2.6-user-guide.pdf>, v. 2, 2006. Citado na página 48.
- NIELSEN, J. *Usability engineering*. [S.l.]: Elsevier, 1994. Citado na página 43.
- PFLEGING, B. et al. Mobidev: a mobile development kit for combined paper-based and in-situ programming on the mobile phone. In: ACM. *CHI'10 Extended Abstracts on Human Factors in Computing Systems*. [S.l.], 2010. p. 3733–3738. Citado na página 63.

- PREECE, J.; ROGERS, Y.; SHARP, H. *Design de interação*. [S.l.]: bookman, 2005. Citado na página 32.
- PRESSMAN, R.; MAXIM, B. *Engenharia de Software-8ª Edição*. [S.l.]: McGraw Hill Brasil, 2016. Citado 7 vezes nas páginas 15, 25, 31, 32, 33, 34 e 36.
- RAMON, O. S. et al. Gui generation from wireframes. In: *14th Int. Conference on Human-Computer Interaction Interaccion'2013*. [S.l.: s.n.], 2013. Citado na página 63.
- ROSE, L. M.; KOLOVOS, D. S.; PAIGE, R. F. Eugenia live: a flexible graphical modelling tool. In: ACM. *Proceedings of the 2012 Extreme Modeling Workshop*. [S.l.], 2012. p. 15–20. Citado na página 47.
- RUI, Y. et al. Relevance feedback a power tool for interactive content-based image retrieval. *IEEE Transactions on circuits and systems for video technology*, IEEE, v. 8, n. 5, p. 644–655, 1998. Citado na página 50.
- SÁ, M. D.; CARRIÇO, L. A mobile tool for in-situ prototyping. In: ACM. *Proceedings of the 11th International Conference on Human-Computer Interaction with Mobile Devices and Services*. [S.l.], 2009. p. 20. Citado na página 63.
- SNYDER, C. *Paper Prototyping: The Fast and Easy Way to Design and Refine User Interfaces*. Morgan Kaufmann, 2003. (ITPro collection). ISBN 9781558608702. Disponível em: <<https://books.google.com.br/books?id=YgBojJsVLGMC>>. Citado na página 25.
- SOMMERVILLE, I. et al. *Engenharia de software*. [S.l.]: Addison Wesley São Paulo, 2003. v. 6. Citado 3 vezes nas páginas 31, 32 e 34.
- SPECIFICATION, O. A. *MOF Model to Text Transformation Language*. 2008. Citado na página 47.
- STEINBERG, D. et al. *EMF: eclipse modeling framework*. [S.l.]: Pearson Education, 2008. Citado 2 vezes nas páginas 46 e 47.
- WARFEL, T. Z. *Prototyping: a practitioner's guide*. [S.l.]: Rosenfeld media, 2009. Citado na página 35.

Apêndices

APÊNDICE A – Especificação do Metamodelo MOP v1.0

As tabelas a seguir descrevem cada classe do Modelo de Domínio do processo de prototipação MOP.

Tabela 6 – Especificação do Modelo de Domínio.

Componente	Metaclasse	Descrição
App	Class	Representa o protótipo
Screen	Class	Representa cada tela do protótipo
UIComponent	Abstract Class	Representa um componente de interface
GroupComponent	UIComponent	Representa um grupo de componentes
SimpleComponent	UIComponent	Representa um componente simples
ProgressBar	SimpleComponent	Representa uma barra de progresso
Label	SimpleComponent	Representa um label
Radio	SimpleComponent	Representa um controle tipo radio
Content	SimpleComponent	Representa um conteúdo textual
Image	SimpleComponent	Representa uma imagem
Switch	SimpleComponent	Representa um controle tipo switch
Button	SimpleComponent	Representa um botão
FlatButton	Button	Representa um botão tipo Flat
FAB	Button	Representa um botão de ação flutuante
Card	GroupComponent	Representa um cartão
Dialog	GroupComponent	Representa uma caixa de diálogo
Bar	GroupComponent	Representa uma barra
AppBar	Bar	Representa um barra tipo cabeçalho
ToolBar	Bar	Representa uma barra de rodapé
List	GroupComponent	Representa uma lista
ListItem	SimpleComponent	Representa um item da lista

Tabela 7 – Classe App

App		
Representa o prototipo		
Atributo	Tipo	Descrição
name	String	Especifica o nome do protótipo
version	String	Especifica a versão do protótipo

Tabela 8 – Classe Screen

Screen		
Representa cada tela do prototipo		
Atributo	Tipo	Descrição
id	String	Especifica o código único da tela
isMainScreen	String	Especifica a Screen inicial do protótipo

Tabela 9 – Classe UIComponent

UIComponent
Classe abstrata que representa cada elemento que compõe a tela do prototipo

Tabela 10 – Classe SimpleComponent

SimpleComponent
Classe abstrata que referencia os componentes simples de cada Screen

Tabela 11 – Classe GroupComponent

GroupComponent
Classe abstrata que referencia os grupos de componente de cada Screen

Tabela 12 – Classe Button

Button		
Classe abstrata que referencia os botões de cada Screen		
Atributo	Tipo	Descrição
name	String	Especifica o nome do botão
iconAction	String	Especifica o ícone do botão de ação flutuante
action	String	Especifica o id da Screen que deve tornar ativa

Tabela 13 – Classe FAB

FAB
Especifica um botão de ação flutuante. É uma subclasse de Button

Tabela 14 – Classe Flat

Flat
Especifica um botão do tipo flat. É uma subclasse de Button

Tabela 15 – Classe Raised

Raised
Especifica um botão do tipo Raised. É uma subclasse de Button

Tabela 16 – Classe Content

Content		
Especifica o conteúdo textual de componente. É uma subclasse de SimpleComponent		
Atributo	Tipo	Descrição
text	String	Especifica o conteúdo textual de componente

Tabela 17 – Classe Label

Label		
Especifica um elemento tipo label. É uma subclasse de SimpleComponent		
Atributo	Tipo	Descrição
text	String	Especifica o conteúdo textual de componente

Tabela 18 – Classe Radio

Radio
Especifica um componente de controle tipo Radio. É uma subclasse de SimpleComponent

Tabela 19 – Classe Switch

Switch
Especifica um componente de controle tipo Switch. É uma subclasse de SimpleComponent

Tabela 20 – Classe ProgressBar

ProgressBar
Especifica um componente tipo ProgressBar. É uma subclasse de SimpleComponent

Tabela 21 – Classe Image

Image		
Especifica um componente tipo Imagem. É uma subclasse de SimpleComponent		
Atributo	Tipo	Descrição
src	String	Especifica o caminho da URL da imagem

Tabela 22 – Classe GroupComponent

GroupComponent
Classe abstrata que referencia os grupos de componente de cada Screen

Tabela 23 – Classe Card

Card
Especifica um componente de Card. É uma subclasse de GroupComponent

Tabela 24 – Classe Dialog

Dialog
Especifica um componente de Caixa de Diálogo. É subclasse de GroupComponent

Tabela 25 – Classe List

List
Especifica um componente de List. É uma subclasse de GroupComponent

Tabela 26 – Classe ListItem

ListItem		
Especifica um componente de controle tipo Item de lista		
Atributo	Tipo	Descrição
icon	String	Especifica o iconAction correspondente
name	String	Especifica o conteúdo textual do componente
action	String	Especifica o id da Screen que deve tonar ativa

Tabela 27 – Classe Bar

Bar
Especifica um componente de Bar. É uma subclasse de GroupComponent

Tabela 28 – Classe AppBar

AppBar		
Especifica um componente de controle tipo AppBar. É uma subclasse de Bar		
Atributo	Tipo	Descrição
title	String	Especifica o título da cabeçalho
type	String	Especifica o tipo de barra

Tabela 29 – Classe ToolBar

ToolBar
Especifica um componente de tipo barra de ferramentas. É subclasse de Bar

APÊNDICE B – Código Resultante do Protótipo

```

1  <!doctype html >
2  <!--
3      MOP based in Material Design Lite
4  -->
5  <html lang= " en " >
6
7  <head >
8      <meta charset= " utf-8 " >
9      <meta http-equiv= " X-UA-Compatible " content= " IE=edge " >
10     <meta name= " description " content= " Aportfolio template that uses
11         Material Design Lite. " >
12     <meta name= " viewport " content= " width=device-width ,
13         initial-scale=1.0 , minimum-scale=1.0 " >
14     <title > MOP </title >
15     <link rel= " stylesheet " href= " https: // fonts.googleapis.com / css ?
16         family=Roboto:regular , bold , italic , thin , light , bolditalic , black
17         , medium & amp;lang=en " >
18     <link rel= " stylesheet " href= " https: // code.getmdl.io /1.3.0/
19         material.blue-red.min.css " />
20     <link rel= " stylesheet " href= " styles.css " >
21     <link rel= " stylesheet " href= " mop.css " >
22     <link rel= " stylesheet " href= " https: // fonts.googleapis.com / icon ?
23         family=Material + Icons " >
24
25 </head >
26 <body >
27     <div class= " mdl-layoutmdl-js-layout mdl-layout--fixed-header " >
28         <header class= " mdl-layout__header " >
29             <div class= " mdl-layout__header-row " >
30
31                 <!--Title -->
32                 <span class= " mdl-layout-title " > MOP< / span >
33                 <!--Add spacer ,to align navigation to the right -->
34                 <div class= " mdl-layout-spacer " > </div >
35                 <!--Navigation. We hide it in small screens. -->
36             </div >
37         </header >
38
39         <main id= ' screen1 ' class= " mdl-layout__content " >
40             <div class= " mdl-gridportfolio-max-width " >

```

```

35         <ul class= " demo-list-iconmdl-list " >
36         <li onclick= " ativaTela ( ' screen3 class= " mdl-list__item " >
37             <span class= " mdl-list__item-primary-content " >
38                 <i class= " material-icons mdl-list__item-icon " > event< / i>
39                 Lembrete 1
40             </ span >
41         < / li >
42         <li onclick= " ativaTela ( ' screen3 class= " mdl-list__item " >
43             <span class= " mdl-list__item-primary-content " >
44                 <i class= " material-iconsmdl-list__item-icon " > event< / i>
45                 Lembrete 2
46             </ span >
47         </ li >
48         <li onclick= " ativaTela ( ' screen4 class= " mdl-list__item " >
49             <span class= " mdl-list__item-primary-content " >
50                 <i class= " material-icons mdl-list__item-icon " > event< / i>
51                 Lembrete 3
52             </ span >
53         </ li >
54         < / ul >
55
56     </div >
57     <button id= " show-dialog type= " button class= " mdl-button
58         mdl-js-button mdl-button--fab mdl-js-ripple-effect
59         mdl-button--colored mdl-shadow--4dp mdl-color--accent
60         fab-bottom " >
61         <i class= " material-icons "role= " presentation " > add< / i>
62         <span class= " visuallyhidden " > Add< / span >
63     < / button >
64     <dialog class= " mdl-dialog " >
65         <h4 class= " mdl-dialog__title " > Novolembrete< / h4 >
66         <div class= " mdl-dialog__content " >
67             <div class= " mdl-textfieldmdl-js-textfield
68                 mdl-textfield--floating-label " >
69                 <input class= " mdl-textfield__input "
70                     pattern= "[ A-Z , a-z]*,"type= " text "
71                     id= " Name " >
72                 <label class= " mdl-textfield__label "for=
73                     " Name " > Nome< / label >
74                 <span class= " mdl-textfield__error " >
75                     Letters and spaces only< / span >
76             </div >
77             <div class= " mdl-textfieldmdl-js-textfield
78                 mdl-textfield--floating-label " >
79                 <input class= " mdl-textfield__input "
80                     pattern= "[ A-Z , a-z]*,"type= " text "
81                     id= " Name " >

```

```

71         <label class= " mdl-textfield__label "for=
           " Name " > Data< / label >
72         <span class= " mdl-textfield__error " >
           Letters and spaces only< / span >
73     </div >
74     <div class= " mdl-textfieldmdl-js-textfield
           mdl-textfield--floating-label " >
75         <input class= " mdl-textfield__input "
           pattern= " [ A-Z , a]* ,"type= " text "
           id= " Name ">
76         <label class= " mdl-textfield__label "for=
           " Name " > Hora< / label >
77         <span class= " mdl-textfield__error " >
           Letters and spaces only< / span >
78     </div >
79 </div >
80 <div class= " mdl-dialog__actions " >
81     <button type= " button "class= " mdl-button " > OK< / button >
82     <button type= " button "class= " mdl-buttonclose " > Cancelar<
           / button >
83 </div >
84 </ dialog >
85 </main >
86
87
88
89 <main id= " screen3 "class= " mdl-layout__content " >
90
91     <div class= " mdl-gridportfolio-max-width " >
92         <div class= " demo-card-eventmdl-card mdl-shadow--2dp " >
93             <div class= " mdl-card__title mdl-card--expand " >
94                 <h4 >
95                     Evento 1: <br >
96                     31/08/2017 <br >
97                     21:00
98                 </ h4 >
99             </div >
100            <div class= " mdl-card__actions mdl-card--border " >
101                <a class= " mdl-buttonmdl-button--colored mdl-js-button
                    mdl-js-ripple-effect " >
102                    Back
103                </ a>
104            <div class= " mdl-layout-spacer " > </div >
105                <i class= " material-icons " > event< / i>
106            </div >
107        </div >
108    </div >

```

```

109 </main >
110
111 <main id= " screen4 class= " mdl-layout__content " >
112
113   <div class= " mdl-gridportfolio-max-width " >
114     <div class= " demo-card-eventmdl-card mdl-shadow--2dp " >
115       <div class= " mdl-card__title mdl-card--expand " >
116         <h4 >
117           Evento 2: <br >
118           31/08/2017 <br >
119           21:00
120         < / h4 >
121       </div >
122       <div class= " mdl-card__actions mdl-card--border " >
123         <a class= " mdl-buttonmdl-button--colored mdl-js-button
124           mdl-js-ripple-effect " >
125           Back
126         < /a >
127       <div class= " mdl-layout-spacer " > </div >
128       <i class= " material-icons " > event< / i>
129     </div >
130   </div >
131
132 </main >
133 </div >
134
135 <script >
136   var dialog = document.querySelector ( ' dialog ' ) ;
137   var showDialogButton = document.querySelector ( '# show-dialog ' );
138   if ( ! dialog.showModal ) {
139     dialogPolyfill.registerDialog ( dialog );
140   }
141   showDialogButton.addEventListener ( ' click ',function (){
142     dialog.showModal () ;
143   }) ;
144   dialog.querySelector ( '. close ' ) . addEventListener ( ' click ',
145     function (){
146       dialog.close () ;
147     }) ;
148 </script >
149
150 <script >
151 function ativaTela ( id ) {
152   var val;
153   var a = document.getElementsByClassName ( ' screen ' );

```

```
154     for ( valof a ){
155         val.className= " screen " ;
156     }
157     var NAME = document.getElementById ( id ) ;
158     NAME.className= " screenscreen-active ";
159 }
160 </script >
161 </div >
162
163     <script src= " https: // code.getmdl.io /1.3.0/ material.min.js " >
        </script >
164 </body >
165
166 </html >
```