UNIVERSIDADE FEDERAL DA PARAÍBA

CENTRO DE INFORMÁTICA

CURSO DE GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

ANÁLISE COMPARATIVA ENTRE PADRÕES ARQUITETURAIS NA PLATAFORMA ANDROID

ALEXANDRE FREITAS CESAR

JOÃO PESSOA

2019

UNIVERSIDADE FEDERAL DA PARAÍBA

CURSO DE GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

ANÁLISE COMPARATIVA ENTRE PADRÕES ARQUITETURAIS NA PLATAFORMA ANDROID

ALEXANDRE FREITAS CESAR

JOÃO PESSOA

CENTRO DE INFORMÁTICA

2019

Alexandre Freitas Cesar

ANÁLISE COMPARATIVA ENTRE PADRÕES ARQUITETURAIS NA PLATAFORMA ANDROID

Monografia apresentada ao curso Ciência da Computação do Centro de Informática, da Universidade Federal da Paraíba, como requisito para a obtenção do grau de Bacharel em ciência da computação.

Orientador: Raoni Kulesza

Setembro de 2019

ANÁLISE COMPARATIVA ENTRE PADRÕES ARQUITETURAIS NA PLATAFORMA ANDROID

ALEXANDRE FREITAS CESAR

Trabalho de Conclusão de Curso de Ciência da Computação intitulado Análise comparativa entre padrões arquiteturais na plataforma Android de autoria de Alexandre Freitas Cesar, aprovada pela banca examinadora constituída pelos seguintes professores:

Prof. Raoni Kulesza

Prof.

Prof.

Coordenador(a) do Departamento CI/UFPB

João Pessoa, 2 de setembro de 2019

Centro de Informática, Universidade Federal da Paraíba

Rua dos Escoteiros, Mangabeira VII, João Pessoa, Paraíba, Brasil CEP: 58058-600

Fone: +55 (83) 3216 7093 / Fax: +55 (83) 3216 7117

AGRADECIMENTOS

"A mente que se abre a uma nova ideia jamais voltará ao seu tamanho original".

Albert Einstein

RESUMO

texto

Palavras-chave: ; ; ; ; ;

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

ABNT	*Associação Brasileira de Normas Técnicas*
GUI	*Graphical User Interface*
MVC	*MODEL-VIEW-CONTROLLER*
MVP	*MODEL-VIEW-PRESENTER*
MVVM	*MODEL-VIEW-VIEW-MODEL*
ATAM	*Architecture tradeoff analysis method*

LISTA DE SÍMBOLOS

Zt	altura de um elemento
Rp	altura máxima do pico
Zp	altura de pico
Rt	amplitude do perfil
R^2	coeficiente de determinação
L	comprimento de amostragem
Rms	desvio quadrático médio
Rku	fator de achatamento ou curtose
LM	linha média
Md	mediana
Rv	profundidade máxima do vale
Zv	profundidade de vale
Ra	rugosidade média ou amplitude média
MPD	textura média do perfil

SUMÁRIO

1 INTRODUÇÃO

O Android foi apresentado ao mundo em 2005, e ao longo desses anos de existência, a plataforma alcançou um impressionante sucesso, tornando-se o sistema operacional móvel mais instalado. Durante esse tempo, foram lançadas várias versões diferentes do sistema operacional, com o Android sempre se tornando mais desenvolvido. Durante a evolução do desenvolvimento Android para aplicações móveis, o Google não forneceu um padrão de arquitetura para aplicar no desenvolvimento de aplicações. Nesse quesito, a plataforma oferece aos desenvolvedores uma liberdade muito grande para desenvolver aplicações, no entanto essa liberdade pode ser tornar um problema quando o objetivo é desenvolver uma aplicação dentro de uma empresa.

1.1 Definição do Problema

As aplicações atuais são construidas usando arquiteturas específicas, no qual diferentes arquiteturas oferecem vantagens diferentes que vêm com diferentes incapacidades. A aplicação de uma arquitetura depende muitas vezes dos requisitos específicos do projeto e propriedades do seu Framework. Os padrões de arquitetura geralmente tem propriedades diferentes para atender diferentes tipos de necessidade, algumas arquiteturas mostram melhor desempenho em grandes projetos, mas requerem muito tempo e mão de obra para ser implementada. No entanto, outras arquiteturas oferecem grande flexibilidade que pode ser difícil de satisfazer testes.

O padrão de arquitetura é um campo de conhecimento que foi desenvolvido durante os anos para melhorar a qualidade dos sistemas e para encontrar soluções. Este conhecimento vai ser importante no futuro, porque novos sistemas exigem novas abordagens para o desenvolvimento. Geralmente as novas arquiteturas são projetadas para solucionar problemas que já existem em arquiteturas antigas, mas o entendimento das vantagens e desvantagens da velha arquitetura com a nova arquitetura deve ser avaliada pela pesquisa. A arquitetura quase sempre é representada por um conjunto de propriedades e uma comparação pode ser feita em cima dessas propriedades, no entanto não basta apenas comparar suas propriedades, é necessario analisar os resultados, no qual esta análise pode ser usada para tomar decisões sobre qual arquitetura usar em um projeto.

1.2 Objetivo geral

A intenção deste trabalho é comparar os principais padrões arquiteturais usados pelos desenvolvedores para construir aplicações na plataforma Android. O presente trabalho tem como principais objetivos:

1.2.1 Objetivo específico

- Conceituar os padrões de arquitetura mais usados.
- Apresentar as características dos padrões arquiteturais.
- Qual arquitetura deve ser usada para desenvolver uma aplicação considerando possíveis limitações.
- Qual arquitetura se encaixa melhor em diferentes tipos de projeto.
- Análise das métricas nas aplicações a partir do código fonte.

1.3 Motivação

O Android é o sistema operacional móvel mais utilizado do mundo, em uma pesquisa feita em 2013 foi mostrado que 71% dos programadores para aplicações moveis desenvolviam para Android.Deste modo, uma investigaçao nesta área de desenvolvimento de software será favorável para os desenvolvedores de aplicações móveis. É possível notar que novas aplicações Android estão constatemente sendo lançadas no mercado com mais desempenho.

A arquitetura mais tradicional utilizada em aplicações Android é o MVP, mas existe uma arquitetura nova chamada FLUX, que foi criado pelo FACEBOOK. Ela foi adaptada para o desenvolvimento Android

obs: Melhorar está seção

1.4 Grupos Alvo

Nesta seção é mostrado o passo 1 do ATAM que consiste de apresentar quem são os stakeholders da pesquisa. O grupo alvo são os desenvolvedores de aplicações Android que buscam um melhor desempenho da sua aplicação e otimização do trabalho em equipe para produzir um aplicativo em menor tempo. Conhecendo os padrões arquiteturais baseado nas métricas importantes, os desenvolvedores poderão ter o conhecimento sobre qual arquitetura deve escolher para solucionar dificuldades do projeto e de sua equipe. A decisão da escolha da arquitetura poderá afetar os custos do desenvolvimento da aplicação Android, manutenção e o tempo necessário para concluir o desenvolvimento.

O TCC descreve o método para comparar arquiteturas móveis, então os pequisadores também são o grupo alvo desta pesquisa.

1.5 Organização do trabalho

O Capítulo 2 descreve a identificação dos padrões arquitetuturais existentes e conceitos importantes para entender o funcionamento do Android. O Capítulo 3 descreve o método utilizado na pesquisa para fazer a análise dos padrões arquiteturais.

O Capítulo 4 descreve os cenários de caso de uso utilizados na pesquisa para escolher as ações que são necessárias para testes.

O Capítulo 5 descreve as aplicações utilizadas para testes de desempenho de CPU, memória e casos de teste.

O Capítulo 6 descreve a apresentação dos resultados.

O Capítulo 7 descreve qual o padrão se apresentou melhor diante dos cenários analisados em tempo de execução, memória e casos de testes.

2 CONCEITOS GERAIS E REVISÃO DA LITERATURA

Neste capítulo é descrito os principais conceitos do Android e o passo 4 do ATAM, no qual os padrões arquiteturais tradicionais são identificados e conceituados.

2.1 Constituição de uma aplicação Android

O sistema operacional Android é contruído sobre alguns dos principais componentes e camadas. Os componentes tem o objetivo de ajudar a desenvolver aplicações robustas, testavéis e sustentáveis. O Google publicou esses componentes com o intuito de auxiliar no desenvolvimento de aplicativos para o Android. Nesta seção será descrito três conceitos importantes neste TCC: Activity, Fragment e Recursos.

2.1.1 Activity

É a maneira de interagir com o usuário, ou seja, uma activity é basicamente uma interface de uma aplicação, para cada nova tela que é criada é necessário uma activity. As activitys quando são abertas utilizam o conceito de pilha, exemplo: quando se tem uma tela e é acessado uma nova tela, a tela nova fica sobre a tela antiga. Dentro da activity é carregado um arquivo xml que compõe a interface.

2.1.1.1 Ciclo de vida

A maioria dos componentes de apps Android têm ciclos de vida ligados a eles, que são gerenciados diretamente pelo próprio sistema.

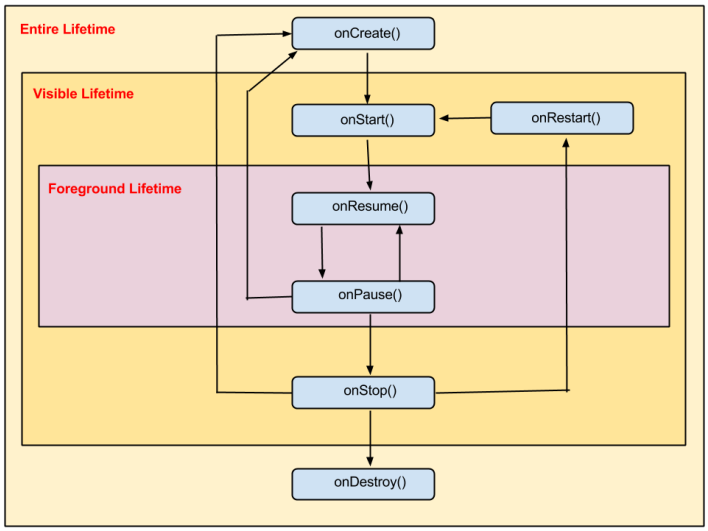

Figura 1 – Ciclo de vida de uma activity

O ciclo de vida é dividido em 3 níveis: Entire lifetime, Visible lifetime e Foreground lifetime. O Entire lifetime aborda todo o ciclo do created até o destroyed. O Visible lifetime aborda só a parte que o usuário pode ver a sua atividade que é do start até o stop; O Foreground lifetime aborda todo contexto que permeti interagir com a atividade. Quando o usuário executa a aplicação, a atividade começa no ONCREATE que é chamado apenas uma única vez, o usuário ainda não pode ver nada na tela, logo depois a atividade vai para o ONSTART no qual inicia a apresentação da tela, mas não pode interagir com a tela, no ONRESUME o usuário já está em execução e é possível interagir.

2.1.2 Fragment

É uma mini activity que possui o seu próprio ciclo de vida e estão interligados a uma activity, mas não é um componente do sistema, pois não tem assinatura no AndroidManifest. O funcionamento é parecido como as Activity's funcionam. A diferença entre ambos é que a Activity é um componente do Android e tem uma conexão com o processo da aplicação, já o Fragment fica sobre o encargo direto da Activity.

2.1.3 Recursos

Os recursos são os arquivos adicionais e conteúdo estático que o código usa. Existe vários tipos de recursos como: Layout, bitmaps, dimensão e etc.

2.2 Padrões de arquitetura

No decorrer de toda a história do desenvolvimento Android, o Google não entregou sua posição sobre qual o padrão de arquitetura de software que deveria utilizar para desenvolver aplicativos Android. Quando lançaram oficialmente as primeiras versões do Android, o objetivo era usar o MVC para o desenvolvimento das aplicações, mas rapidamente surgiram problemas nesse padrão arquitetural, então foi preciso criar novas abordagens como os padrões MVP, MVVM, MVC.

2.2.1 Arquitetura multicamadas

O sistema multicamadas não tem uma definição clara de quais são as camadas que certamente devem ter em um aplicativo Android, nem mesmo o número mínimo e máximo. Este sistema faz uso de objetos distribuídos(permite a operação com objetos remotos) associado à utilização de interfaces para executar seus procedimentos, podendo esta aplicação Android ser dividida em várias partes, cada uma bem definida, com suas características e responsável por determinadas funções. Em um aplicativo nesta forma,

pelo menos três camadas são necessárias: apresentação, regras de negócios e banco de dados.

2.2.2 MVC

O MVC foi criado na década de 70 como forma de separar a lógica de apresentação da lógica de negócios, reduzindo a ligação entre classes. No ano de 2009, quando saiu as primeiras versões do Android, a ideia era usar o padrão arquitetural MVC para o desenvolvimento dos projetos. O incentivo para usar o MVC foi os desenvolvedores conseguirem paralelizar o desenvolvimento e conseguir fazer o que chamamos de reuso de código. Esta abordagem separa sua aplicação em um nível de composição com 3 conjuntos de funções. Sendo eles:

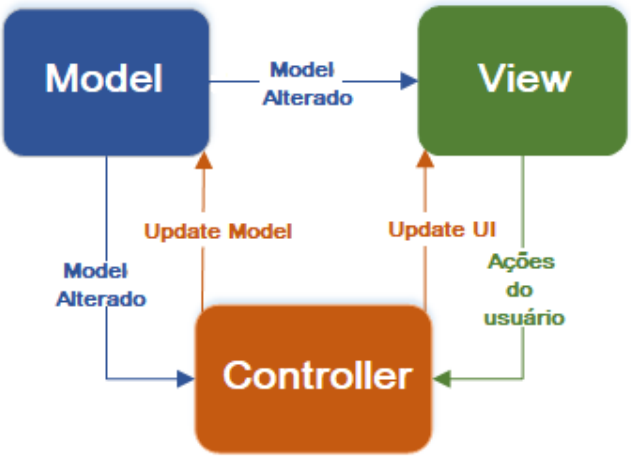

Figura 2 – Arquitetura MVC

A Camada View é responsável por apresentar informações de forma gráfica para o usuário e faz a comunicação com o controller; A Camada Controller é responsável por fazer a ligação entre a interface gráfica e a lógica de negócios, alterando atributos e mostrando as modificações na interface gráfica; A Camada Model é onde a lógica de negócio é definida, incluindo a persistência de dados.

O MVC consegue separar o Model e a View. Deste jeito o Model pode ser testado, pois não está vinculado a nada e a View não tem muito para testar em um nível de teste unitário. No entanto, o Controller não vai ficar disponível para testes, pois várias partes que deveriam estar dentro do controller foram passadas para o activity.

2.2.3 MVP

O MVP surgiu na década de 90, no qual a união da APPLE, HP e IBM fez com que esse padrão arquitetural surgisse para complementar necessidades que o padrão MVP

não cobria.

O MVP possibilita uma divisão melhor para camadas quando o objetivo é separar camadas superiores de camadas inferiores, ou seja, possibilita que a camada diretamente relacionada com a interface do usuário somente se comunique com a camada diretamente abaixo dela. Esta abordagem separa sua aplicação em um nível de composição com 3 conjuntos de funções. Sendo eles:

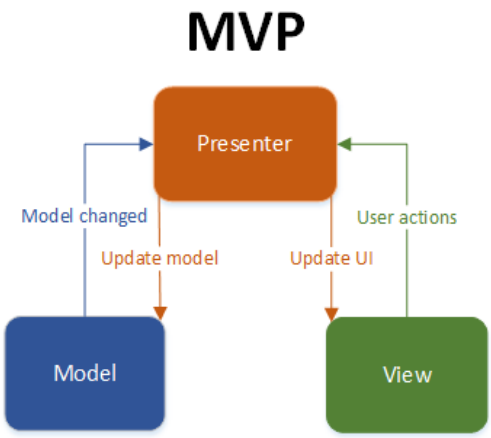

Figura 3 – Arquitetura MVP

A camada View é responsável por responder a saída ou entrada de dados, no qual a saída vem da camada Presenter, e a entrada vem normalmente do usuário; A camada Presenter é responsável por responder as invocações da camada de visualização e invocações da camada de modelo e pode também invocar ambas as camadas. O Presenter pode incluir lógica de negócio e pode ser responsável pela formatação dos dados; A camada model: é responsável por fornecer os dados e pode conter lógica de negócio para domínio do problema.

A grande vantagem do padrão arquitetural MVP é que ele descreve três níveis de abstração, o que torna mais fácil para depurar a aplicação Android. A separação da lógica de negócios e lógica de acesso a dados também permite o teste que vai ser implementado. A camada presenter faz referência ao Controller do MVC, exceto por ele não estar vinculado ao View, apenas a uma interface, com isto ele não mais gerencia o tráfego de solicitações recebidas, como é feito no Controller.

2.2.4 MVVM

O MVVM é uma pequena evolução do MVP em uma parte e uma decadência em outro. Neste modelo o ViewModel não está ciente do que ocorre no View, mas este está informado do que ocorre no ViewModel. No caso do Model ambos estão cientes do

que ocorre em cada um. O nome se dá porque ele adiciona propriedades e operações ao Model para atender as necessidades do View, portanto ele cria um novo modelo para a visualização.

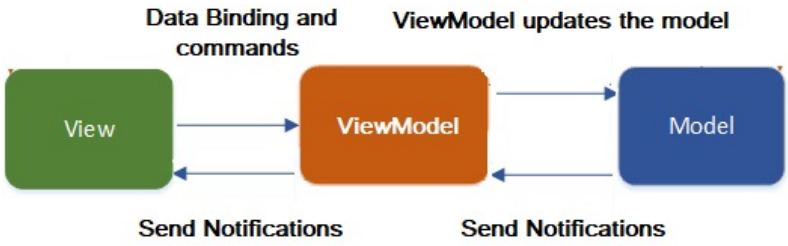

Figura 4 – Arquitetura MVVM

A camada View liga-se a variáveis Observable e ações expostas pelo ViewModel de forma flexível; A camada viewModel: A responsabilidade da ViewModel no contexto do MVVM, é disponibilizar para a View uma lógica de apresentação. A View Model não tem nenhum conhecimento específico sobre a view, ou como ela implementada, nem o seu tipo. A ViewModel implementa propriedades e comandos, para que a View possa preencher seus controles e notifica a mesma, caso haja alteração de estado; seja através de eventos ou notificação de alteração. A ViewModel é peça fundamental no MVVM, por que é ela quem vai coordenar as iterações da View com o Model, considerando que ambos não terem conhecimento um do outro. E além de tudo isto, a ViewModel, também pode implementar a lógica de validação, para garantir a consistência dos dados; A camada model é responsável por fornecer os dados e pode conter lógica de negócio para domínio do problema.

2.2.5 FLUX

A arquitetura do Flux é usada pelo Facebook para construir seus aplicativos da Web do lado do cliente e também pode ser utilizado em aplicações moveis. Ele é dividido em 3 partes: Dispatcher, View e Store.

A camada View é a interface do aplicativo. Cria ações em resposta às interações do usuário. A camada Dispatcher. A camada Dispatcher é um hub central através do qual passam todas as ações e cuja responsabilidade é fazer com que elas cheguem a todos os stores. A camada store é responsável por manter o estado de um determinado domínio de aplicativo. Eles respondem a ações de acordo com o estado atual, executam a lógica de negócios e emitem um evento de mudança quando são concluídas. Esse evento é usado pela visão para atualizar sua interface.

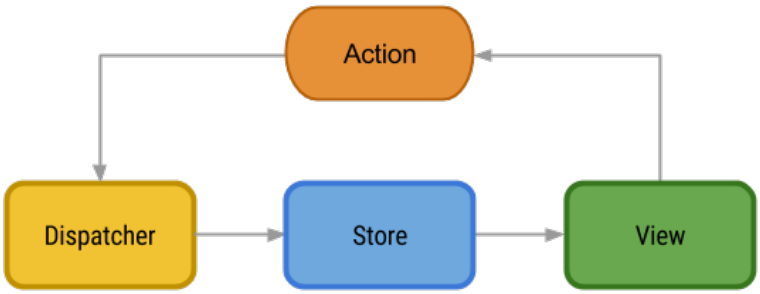

Figura 5 – Arquitetura Flux

2.3 Casos de teste

Os casos de teste são divididos em 3 categorias, são eles:

- Primeira categoria é os testes de unidade cujo o objetivo é garantir que cada componente do sistema está funcionando de maneira correta.
- Segunda categoria é os testes de integração cujo objetivo é saber se os componentes funcionam corretamente juntos.
- Terceira categoria é os teste de interface que garante que a GUI é exibida no momento certo e é sensível às ações dos usuários.

3 METODOLOGIA

Os método ATAM foi adotado nesta pesquisa afim de analisar comparativamente a qualidade do código e os detalhes na estrutura de implementações da mesma aplicação utilizando padrões arquiteturais diferentes. O método ATAM analisa quão bem a arquitetura de software satifaz objetivos particulares de qualidade. Este método é baseado no método de análise de arquitetura de software(SAAM) e é considerado uma evolução deste método.O método é formado por 4 fases e 7 passos, sendo eles:

3.1 Fase de apresentação

- Passo 1 - Apresentação do ATAM para os stakeholders.
- Passo 2 - Descrever os objetivos de négocio que estão motivando o desenvolvimento e quais serão os requisitos arquiteturalmente relevantes.
- Passo 3 - Apresentar a arquitetura proposta.

3.2 Fase de investigação e análise

- Passo 4 - Identificar as abordagens arquiteturais tradicionais.
- Passo 5 - Neste passo, os cenários de uso comuns e importantes do sistema são identificados. Quais os tipos de problemas o sistema deve resolver? Este passo é realizado com base no sistema específico no contexto do mundo real.
- Passo 6 - Neste passo, os requisitos são coletados para critérios de avaliação. Os requisitos são características mais importantes para o sistema estabelecido pelas partes interessadas.

3.3 Fase de teste

- Passo 7 - Brainstorm e priorização de cenários:
- Passo 8 - Realizar a etapa 6 novamente, mas aqui os cenários mais relevantes do passo 7 são considerados para serem testados para análise do padrão de arquitetura.

3.4 Fase de relátorio

- Passo 9 - Apresentar resultados, neste passo é mostrado um relátorio com base nas informações recolhidas(estilos, cenários, questões específicas de atributos, a árvore de utilidades, riscos, pontos de sensibilidade, compensações).

Nesta pesquisa vai ser utilizada uma pesquisa prática, a pesquisa é realizada em conjunto de testes. Os testes são baseados nas principais métricas exigidas pelas aplicações atuais como: desempenho, capacidade de teste e modificabilidade. A principal propriedade de uma arquitetura de projeto é a manutenção. O desempenho deve ser medido pois os requisitos da arquitetura exigem um gerenciamento de memória e processamento adequados para que a interface do usuário não sofra atrasos.

4 CENÁRIOS DE USO

Um passo necessário do método ATAM é o levantamento dos cenários de uso, ele é um conjunto de ações básicas que é necessário para completar um caso de uso, são as ações são descritas do ponto de vista da aplicação. As ações básicas que vão ser realizadas durantes os testes são as seguintes: clicar no botão, acessar armazenamento de dados e memória, entrar em uma nova tela, selecionar os itens da lista e etc. As aplicações mais conhecidas do Android geralmente são formadas por cenários de caso de uso pequenos que consistem em até cinco ações básicas.

4.1 Aplicativo Youtube Music

É uma das aplicações criadas pelo Google. Esta aplicação permite que o usuários procurem e escutem músicas. Este cenário de uso é o mais utilizado no Youtube Music. As ações básicas:

- Tela principal aberta
- Carregar músicas recentemente escutadas
- Clicar no ícone "lupa"para pesquisar uma música
- Clicar no ícone da música
- Abrir nova tela para transmitir a música

4.2 Aplicativo Tarefas

É uma das aplicações criadas pelo Google. Esta aplicação permite que o usuários registrem tarefas. As ações básicas:

- Tela principal aberta
- Clicar no botão "+"para registrar uma tarefa
- Escrever uma nova tarefa
- Clicar no botão para registrar detalhes
- Clicar no botão salvar
- Abrir nova tela para listar tarefas salvas

Outro cenário de caso de uso é editar a sua tarefa. As ações básicas:

- Tela principal aberta
- Clicar na tarefa
- Abrir nova tela para editar tarefa
- Editar nome ou descrição da tarefa
- Confirmar edição
- Abrir nova tela para listar tarefas salvas

Outro cenário de caso de uso é ver o feed do usuário. As ações básicas:

- Tela principal aberta
- Tarefas registradas na tela principal
- Rolar para ver tarefas

Outro cenário de caso de uso é deletar as tarefas. As ações básicas:

- Tela principal aberta
- Clicar na tarefa
- Abrir nova tela com descrição e nome da tarefa
- clicar no ícone da lixeira para deletar
- Voltar para tela principal

4.3 Aplicativo Medium

Esta aplicação permite que o usuário pesquise para encontrar artigos explicando vários assuntos. O cenário de uso mais popular é a pesquisa. Este cenário é composto pelas seguintes ações básicas:

- Tela principal aberta
- Digitar o texto da pesquisa
- Clicar na lupa para pesquisar
- Abrir nova tela para listar sua pesquisa

- Clicar no ícone da imagem da pesquisa para ver o texto
- Abrir nova tela para exibir a pesquisa

É importante notar que os cenários de uso mais populares consistem de ações básicas, ou seja, A maioria dos cenários de caso de uso tem como finalidade permiti que o usuário alcance os seus objetivos mais rapidamente. Isto significa dizer que os cenários de caso de uso pequenos são de interesse para pesquisadores e desenvolvedores.

5 APLICAÇÕES ANDROID

Os casos de uso são implementados na forma de aplicativos Android. Estas aplicações requerem nível mínimo API Android 14, que corresponde ao Android 4.0.1 - 4.0.2 Ice Cream Sandwich, em torno dos 99% está o mercado que funciona com a versão 4.0 até a 9.0 que é do Android mais atual. A aplicação mostrada na figura 6 foi implementada pelo Google utilizando a linguagem Java e Kotlin. Os testes foram realizados utilizando uma ferramenta do Android studio chamada Profile performance que é capaz de medir o tempo de uso dos métodos da aplicação em execução, ela também pode medir o consumo de memória e energia.

5.1 Aplicativo TODO

Este aplicativo foi desenvolvido pelo Google para mostrar diferentes abordagens arquitetônicas para o desenvolvimento de aplicativos no sistema operacional Android. Esta aplicação está disponível nos padrões MVP e MVVM.

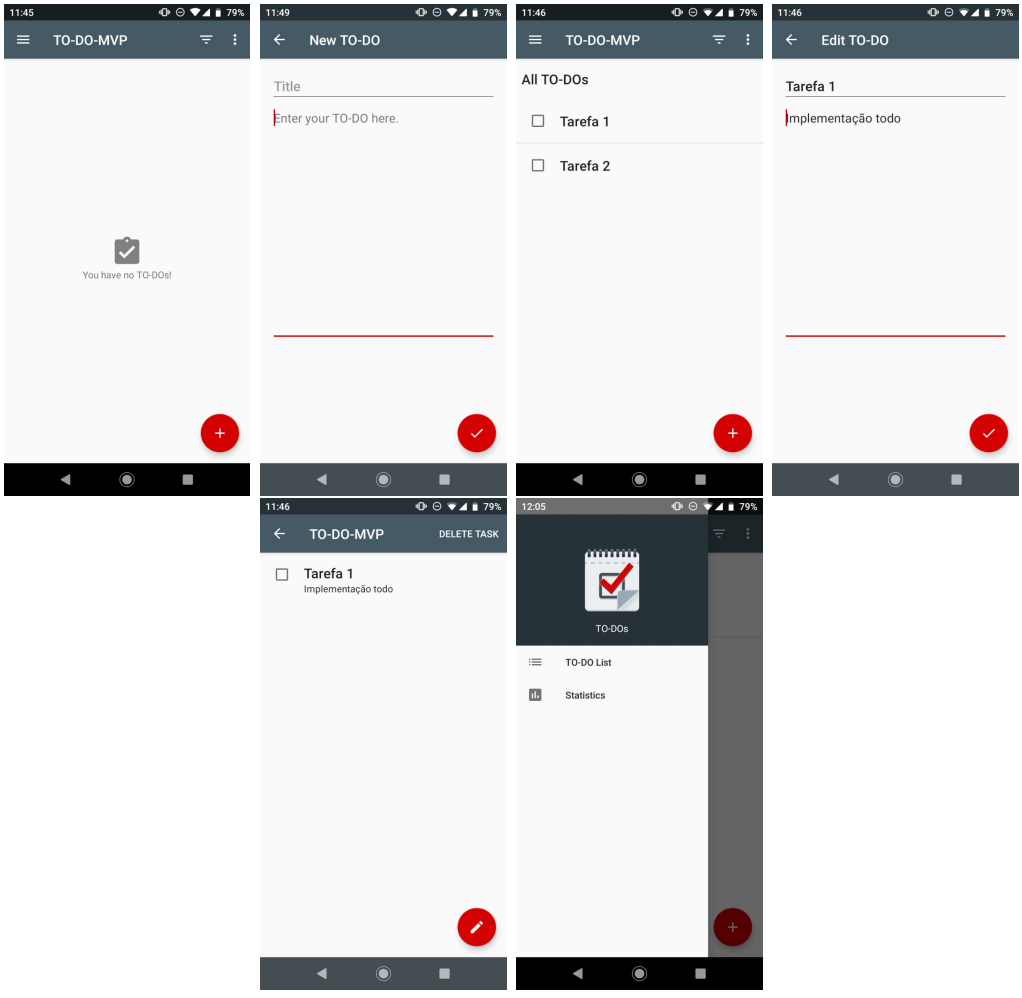

Figura 6 – Aplicativo TODO GOOGLE

A figura 6 mostra a interface do usuário das telas em caso de uso. A interface de usuário utiliza uma abordagem simples que é para focar no principal que é a arquitetura. A navegação é feita utilizando menu latetal e botões de cadastramento e visualização. O botão vermelho é utilizado para chamar atenção do usuário sobre sua funcionalidade principal que é cadastrar e editar. O menu é utilizado para mostrar estatísticas e voltar para o menu principal com a lista de tarefas.

5.2 Aplicativo Movie Finder

Este aplicativo foi desenvolvido pelo DigiGene para comparar diferentes abordagens arquitetônicas para o desenvolvimento de aplicativos no sistema operacional Android. Esta aplicação está disponível nos padrões MVC, MVP, MVVM e sem padrão.

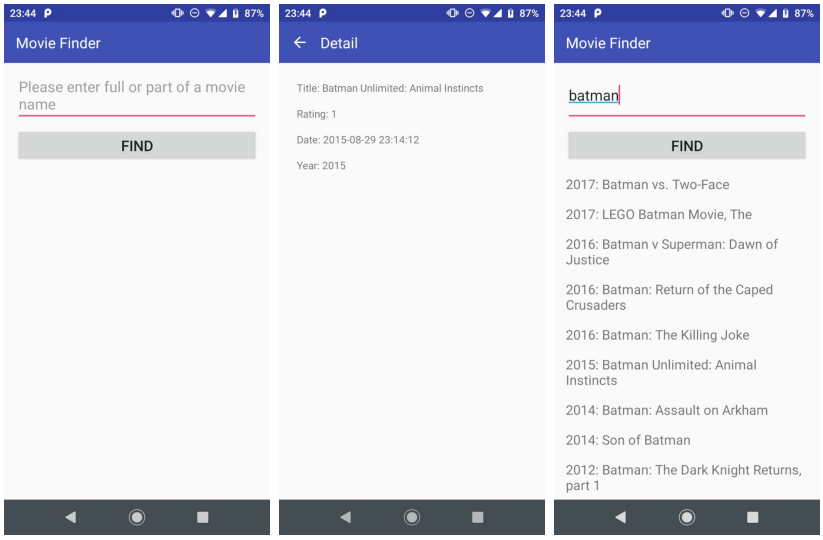

Figura 7 – Aplicativo Movie Finder

A figura 7 mostra a interface do usuário das telas em caso de uso. A interface de usuário utiliza uma abordagem simples que é para focar no principal que é a arquitetura. A navegação é feita utilizando botões. O botão finder é utilizado para encontrar filmes. Clicar no nome do filme que está na lista é mostrado uma descrição.

6 RESULTADOS

Este capítulo é referente ao passo 9 que é o mais importante do ATAM, neste capítulo é mostrado uma descrição com base nas informações recolhidas como: estilos, cenários, questões específicas dos atributos, a árvore de utilidades, riscos, pontos de sensibilidade. A implementação do relátorio consistiu de um conjunto de casos de uso, que executa operações básicas nas aplicações Android levantadas. Os casos de uso levantados vão servir para representar os cenários de uso que foram levantados no capítulo 4. Estes cenários tem um conjunto de ações simples como: adicionar, remover, deletar, procurar e listar. As ações que exigem internet como no cenário de pesquisa deve exigir mais tempo, pois depende também do tempo de acesso a internet. As ações que são apenas cálculo de hardware devem ter o acesso rápido, pois não depende do requisito da internet. Neste capítulo também será mostrado tabelas com medições, obtidos nos testes. Os dados vão ser comparados para responder objetivos referentes às perguntas do TCC. Os padrões arquiteturais vão ser avaliados no contexto das medidas e analisadas para responder objetivos do TCC e seguir os passos do ATAM.

Os padrões de arquiteturas são avaliados de acordo com as métricas principais levantadas como: desempenho e capacidade de teste. Profile é uma ferramenta utilizada para medir desempenho e uso de memória, A ferramenta é capaz de otimizar e mostrar problemas de desempenho presentes, animações que nunca terminanam, problemas de congelamento, alto consumo de energia. Uma otimização do uso da CPU em um aplicativo Android pode deixar a aplicação mais rápida e suave.

obs Colocar texto sobre o appium

6.1 Inspecionar atividades de CPU para comparar desempenho

O desempenho é uma métrica muito importante para avaliar os padrões de arquitetura. Uma arquitetura com um desempenho comprometido pode levar o usuário a ter um experiência de uso ruim devido a problemas de congelamento e lentidão quando o aplicativo abre novas telas, então o desempenho deve ser considerado uma métrica principal para uma análise de arquitetura.

O uso da ferramenta Profile do Android studio vai servir para inspecionar o uso de CPU das atividades em tempo real, com está ferramenta o tempo de execução das principais funções do aplicativo vão ser capturadas em microsegundos e milisegundos e estes dados vão ser utilizados para comparação dos padrões arquiteturais MVC, MVP e MVVM.

6.1.1 Descrição do Experimento com aplicação TODO da Google

O experimento tem como objetivo comparar arquiteturas pelo desempenho dos casos de uso construídos com padrões arquiteturais que foram levantados. O desempenho é medido em tempo real em termos de tempo de execução, memória total utilizada, memória gráfica, memória utilizada pelo código Java.

Os casos de uso do Google Samples vão ser utilizados para avaliar os padrões arquiteturais MVP e MVVM. As avaliações vão ser em cima das funções: criar uma tarefa, editar uma tarefa, excluir uma tarefa e listar as tarefas.

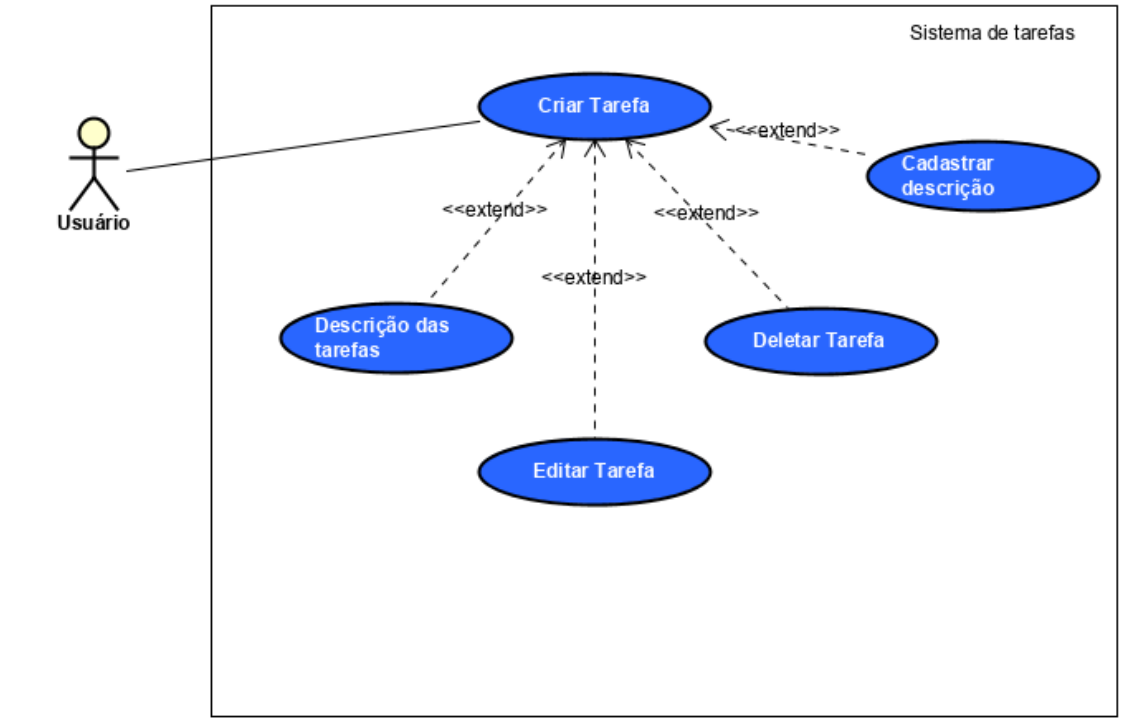

Figura 8 – Diagrama de caso de uso do TODO

Tabela 1 – Resultado do uso de CPU para adicionar tarefa com o tempo medido em μs

Padrões Arquiteturais - μs	Média		Mediana		Primeiro
	5 testes	10 testes	5 testes	10 testes	1 teste
MVP	16.563	14.489	15.094	13.223	28.992
MVVM	12.593	12.107	12.960	11.561	14.660

A Tabela 1 mostra a média e mediana do tempo de execução em tempo real da função criar tarefa que foi implementada para os ambos padrões arquiteturais MVP e MVVM. Os casos de uso foram capturados durante 5 e 10 testes criando tarefas.

A Figura 8 mostra o desempenho de cpu para criação da tarefa em tempo real para os padrões arquiteturais MVP e MVVM que foi medido utilizando a ferramenta profile do Android studio.

Figura 9 – Desempenho do criar tarefa MVP x MVVM

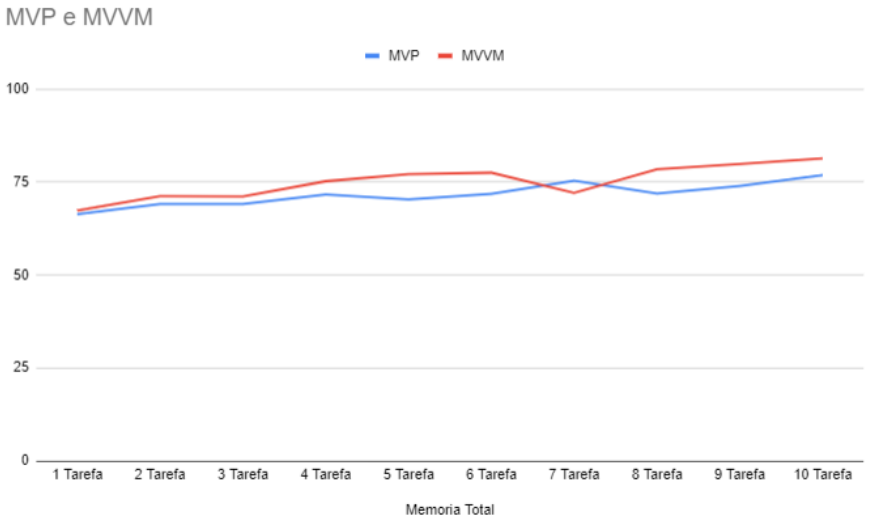

Figura 10 – Desempenho da memória em mb - MVP x MVVM

A Figura 9 mostra o desempenho da memória para criação da tarefa em tempo real para os padrões arquiteturais MVP e MVVM que foi medido utilizando a ferramenta profile do Android studio.

A Figura 10 mostra o desempenho dos gráficos para criação da tarefa em tempo real para os padrões arquiteturais MVP e MVVM que foi medido utilizando a ferramenta profile do Android studio.

A Figura 11 mostra o desempenho da CPU para listar tarefas em tempo real para os padrões arquiteturais MVP e MVVM que foi medido utilizando a ferramenta profile do Android studio.

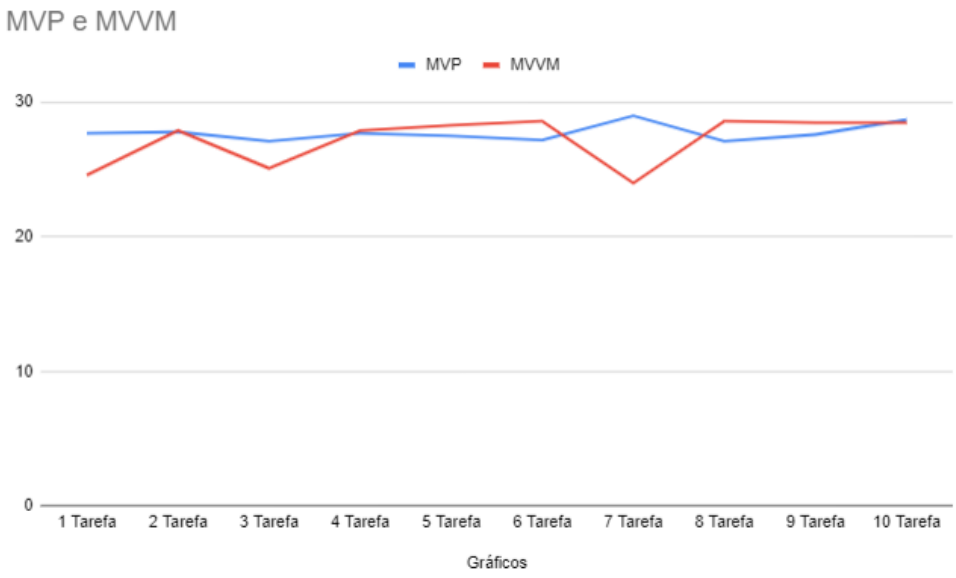

Figura 11 – Desempenho dos gráficos em mb - MVP x MVVM

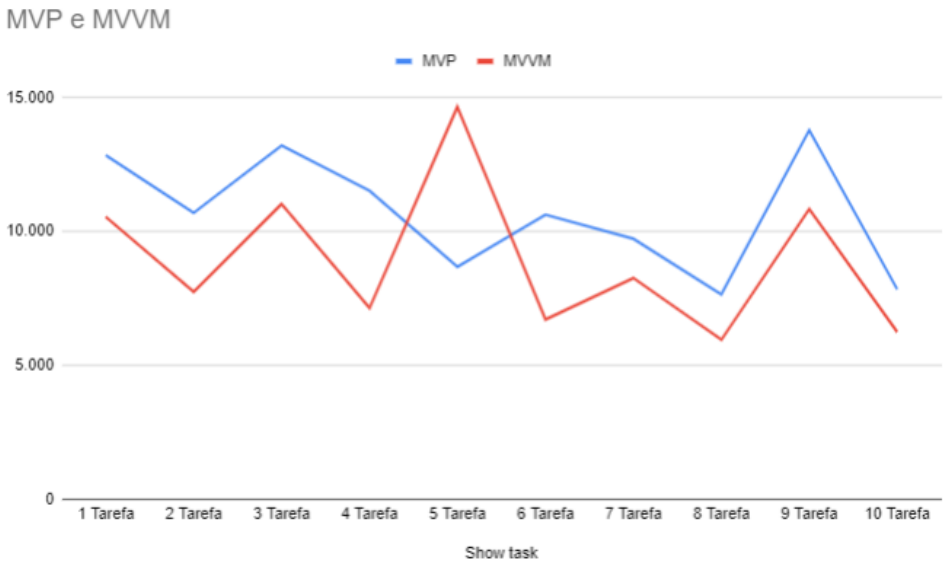

Figura 12 – Desempenho do showTask em microsegundos - MVP x MVVM

6.1.1.1 Delete task

texto texto texto

6.1.1.2 Edit task

texto texto texto

6.1.2 Descrição do Experimento com aplicação Finder

O aplicativo Movie Finder vai ser utilizado para avaliar os padrões arquiteturais MVC,MVP e MVVM. As avaliações vão ser em cima das funções: Pesquisar e lista.

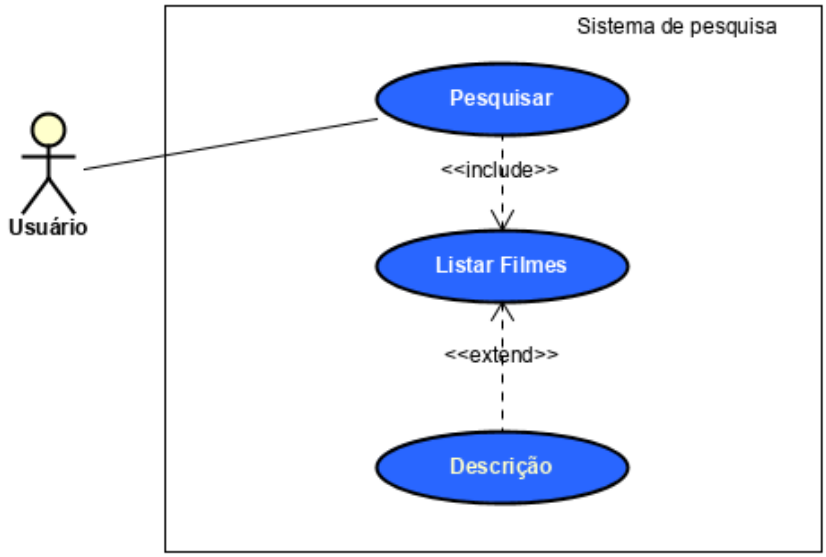

Figura 13 – Diagrama de caso de uso do Movie Finder

7 CONCLUSÃO E TRABALHOS

7.1 Conclusão

7.2 Trabalhos futuros

Durante o desenvolvimento do TCC foi percebido pontos que podem ser melhorados e novos pontos para acrescentar na pesquisa que são de interesse na comunidade do Android, são eles:

- Levantar mais cenários poderia dar melhores resultados e ter um noção maior sobre qual padrão arquitetural pode abranger mais cenários que simulam os aplicativos nos dias de hoje.
- Obter aplicações mais complexas implementadas com diferentes arquiteturas, mas com mesma função.
- Utilizar diferentes padrões GOF para analisar o comportamento do desempenho e a facilidade para produzir o código com diferente padrões arquiteturais.
- O TCC poderia incluir mais métricas para comparar.
- Utilizar diferentes dispositivos Android para saber o desempenho deles diante dos padrões arquiteturais.
- Investigar outros padrões arquiteturais e poder encontrar melhores resultados. Novas arquiteturas estão surgindo ao longo dos anos e comparar elas com arquiteturas tradicionais pode criar uma pesquisa forte com grandes resultados na comunidade.

REFERÊNCIAS

[1] NUNES, Filipe. Android MVC x MVP x MVVM qual Pattern utilizar — Parte 1. 2017. Disponível em: https://medium.com/@FilipeFNunes/android-mvc-x-mvp-x-mvvm-qual-pattern-utilizar-parte-1-3defc5c89afd. Acesso em: 19 ago. 2019

[2] LOU, T. A comparison of Android Native App Architecture MVC, MVP and MVVM. 2016. Disponível em: https://pure.tue.nl/ws/portalfiles/portal/48628529/Lou_2016.pdf Acesso em: 19 ago. 2019

[3] Documentação para desenvolvedores de apps.Disponível em: https://developer.android.com/docs. Acesso em: 19 ago. 2019

[4] Profile your app performance.2017 Disponível em: https://developer.android.com/studio/profile/. Acesso em: 19 ago. 2019

[5] Testar o aplicativo. 2017 Disponível em: https://developer.android.com/studio/test. Acesso em: 19 ago. 2019

[6] Fragment. Disponível em : https://developer.android.com/reference/android/app/ent. Acesso em: 19 ago. 2019

[7] android.app. Disponível em https://developer.android.com/reference/android/app/package-summary. Acesso em: 19 ago. 2019

[8] HUMENIUK, Vladyslav. Android Architecture Comparison: MVP vs. VIPER. 2019. Disponível em: http://lnu.diva-portal.org/smash/get/diva2:1291671/FULLTEXT01.pdf Acesso em: 19 ago. 2019

[9] CAVALCANTE, Henri. Flux — A arquitetura do Facebook para desenvolvimento FrontEnd. 2015 Disponível em: https://medium.com/@henricavalcante/flux-a-arquitetura-do-facebook-para-desenvolvimento-frontend-2bf7192c8f77. Acesso em: 19 ago. 2019

[10] CECHINEL, Alexandre. Avaliação do framework Angular e das bibliotecas React e Knockout para o desenvolvimento do Frontend de aplicações Web. Disponível em: https://repositorio.ufsc.br/bitstream/handle/123456789/182199/TCC%20PROJETOS%202%20-%20ALEXANDRE%20CECHINEL.pdf?sequence=1.
Acesso em: 27 ago. 2019

REFERÊNCIAS