UNIVERSIDADE
FEDERAL RURAL
DE PERNAMBUCO

Yasmmin Maria Monteiro Claudino

Como o uso de *Play Feature Delivery* no *Android* pode ajudar na sustentabilidade digital

Recife

Setembro de 2023

Yasmmin Maria Monteiro Claudino

Como o uso de *Play Feature Delivery* no *Android* pode ajudar na sustentabilidade digital

Artigo apresentado ao Curso de Bacharelado em Sistemas de Informação da Universidade Federal Rural de Pernambuco, como requisito parcial para obtenção do título de Bacharel em Sistemas de Informação.

Universidade Federal Rural de Pernambuco – UFRPE
Departamento de Estatística e Informática
Curso de Bacharelado em Sistemas de Informação

Orientador: Gabriel Alves De Albuquerque Junior

Recife
Setembro de 2023

Como o uso de *Play Feature Delivery* no *Android* pode ajudar na sustentabilidade digital

Yasmmin Maria Monteiro Claudino[1], **Gabriel Alves de Albuquerque Júnior** [2]

[1]Departamento de Estatística e Informática – Universidade Federal Rural de Pernambuco
Rua Dom Manuel de Medeiros, s/n, - CEP: 52171-900 – Recife – PE – Brasil

yasmmin.claudino@ufrpe.br, gabriel.alves@ufrpe.br

Resumo. *Com o crescente acesso à internet via dispositivos móveis entre as classes menos favorecidas no Brasil, a sustentabilidade digital ganha crescente importância. O objetivo deste estudo é avaliar o impacto da utilização da tecnologia Play Feature Delivery no consumo de dados móveis em dispositivos Android. Foi aplicado um teste estatístico de hipótese paramétrica t para avaliar diferenças significativas entre as médias de quantidade de dados gastos em Megabytes ao fazer o download de duas aplicações. O resultado correspondeu a um valor de t aproximadamente de $65{,}55$ e a rejeição da hipótese nula. Esta descoberta não apenas sublinha a importância técnica para desenvolvedores Android, mas também ressalta sua relevância ao mostrar a melhora no uso dos dados móveis, especialmente em uma era onde a democratização do acesso à informação é vital. A pesquisa reforça a ideia de que a sociedade deve adaptar-se a recursos digitais, otimizando o uso de dados. Para alcançar essas conclusões, duas aplicações móveis focadas em orientações sobre primeiros socorros foram desenvolvidas e analisadas. A principal vantagem observada foi a diminuição no consumo de dados móveis, validando a eficácia da Play Feature Delivery em comparação com aplicações convencionais.*

Abstract. *With the increasing access to the internet via mobile devices among the less privileged classes in Brazil, digital sustainability becomes of growing importance. The aim of this study is to assess the impact of the Play Feature Delivery technology on mobile data consumption on Android devices. A parametric t-test was applied to evaluate significant differences between the averages of data amounts spent in Megabytes when downloading two applications. The result corresponded to a t-value of approximately 65.55 and the rejection of the null hypothesis. This finding not only underscores the technical importance for Android developers but also highlights its relevance in showing an improvement in mobile data usage, especially in an era where the democratization of access to information is vital. The research reinforces the idea that society should adapt to digital resources, optimizing data usage. To reach these conclusions, two mobile applications focused on first aid guidelines were developed and analyzed. The main advantage observed was the reduction in mobile data consumption, validating the efficacy of Play Feature Delivery compared to conventional applications.*

1. Introdução

O uso de aparelhos celulares para acesso a informações vem crescendo ao longo dos últimos anos, segundo uma pesquisa realizada pela empresa *Bain & Company* no ano de

2020, 97% dos usuários no Brasil acessam a internet por meio de um celular, e 51% a acessam exclusivamente dessa forma. Ao se tratar dos acessos por classes sociais, o corte nas classes D/E, o dispositivo móvel possui um papel ainda mais significativo, com 83% dos usuários possuindo apenas o *smartphone* como a única forma de acesso [Moura and Camargo 2020] como pode ser observado na Figura 1.

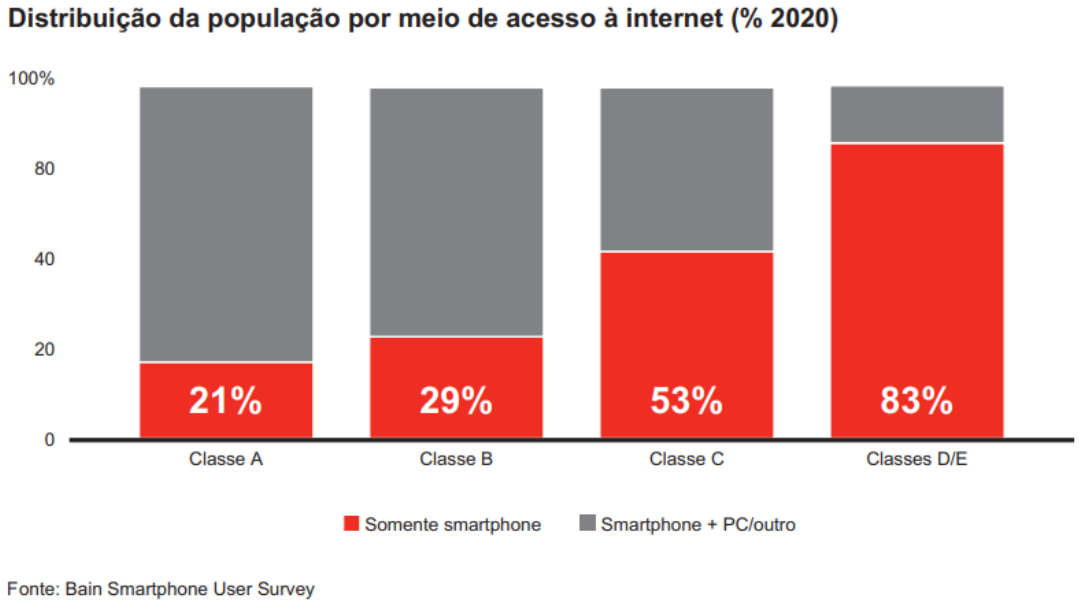

Figura 1. Distribuição da população por meio de acesso à internet (% 2020).
Fonte: Brain Smartphone User Survey

A quantidade de usuários *Android* no Brasil, em maio de 2023 concentrava mais de 78% do mercado de sistemas operacionais móveis no Brasil. Já o *iOS* da Apple detinha uma participação de quase 21,5 por cento e era o único outro sistema com uma participação superior a dez por cento naquele momento [Laricchia 2023]. Além do mais, o valor médio pago em celulares com o sistema operacional *Android*, custa em média de R$ 1,091, enquanto aparelhos com o *iOS* chega a R$ 2,513 [Moura and Camargo 2020].

À medida que se observa a expansão do acesso à internet e a digitalização de informações em escala global, é ressaltado um conceito emergente denominado Sustentabilidade Digital. Ela tem como principal objetivo discutir, criar, usar e regularizar recursos digitais a fim de maximizar seu valor para sociedade atual e no futuro. Além do mais, a Sustentabilidade Digital reforça que os bens digitais devem ser disponibilizados gratuitamente para serem compartilhados na maior extensão possível, possibilitando o potencial para inovação e valor integral para a sociedade [Stuermer 2014].

Diante deste cenário, a prevalência do uso de aparelhos móveis, especialmente entre as classes sociais mais baixas, destaca a importância crítica da sustentabilidade digital. Portanto, é necessário estabelecer as condições adequadas em um nível regulatório para que os recursos digitais livremente acessíveis sejam promovidos e os canais de distribuição, como a *Internet*, sejam igualmente acessíveis a todos [Stuermer 2014].

Além do mais, de acordo Marcus Dapp [Dapp 2013] os recursos digitais são gerenciados de forma sustentável se sua utilidade para a sociedade é maximizada, de modo

que as necessidades digitais das gerações contemporâneas e futuras sejam igualmente atendidas. As necessidades digitais são atendidas de forma ideal se os recursos estiverem acessíveis para o maior número de pessoas e forem reutilizáveis com restrições mínimas. Recursos digitais abrangem conhecimento e artefatos culturais representados em forma digital, por exemplo, texto, imagem, áudio, vídeo ou software.

Em dispositivos *Android*, o tamanho da aplicação influencia diretamente no consumo de dados móveis do usuário. Esta correlação é particularmente evidente em regiões com conectividade limitada. Este trabalho busca investigar como o uso de *Play Feature Delivery* pode otimizar o consumo do uso dos dados móveis em um dispositivo *Android*, promovendo assim, um acesso mais abrangente à informação, um dos pilares fundamentais na definição da sustentabilidade digital.

1.1. Objetivo geral

O objetivo central deste estudo é desenvolver e realizar uma comparação entre duas aplicações, distintas no que tange ao consumo de dados móveis, com o propósito de evidenciar a eficácia da incorporação do *Play Feature Delivery* na promoção da sustentabilidade digital. A aplicação enfatiza dicas de primeiros socorros, apresentando uma lista de potenciais acidentes. Ao selecionar um item da lista, o usuário pode acessar o conteúdo correspondente. A avaliação da eficácia será baseada em métricas como tamanho de *download* do aplicativo. Através deste parâmetro, espera-se demonstrar que uma aplicação que emprega a *Play Feature Delivery* pode proporcionar um acesso à informação mais ágil e confiável, em especial em contextos de rede adversos, quando comparada a uma aplicação que não o utiliza.

1.1.1. Objetivos específicos

1. Implementar duas aplicações com funcionalidades idênticas focadas em dicas de primeiros socorros, diferenciando-as pela integração ou ausência do *Play Feature Delivery*;
2. Disponibilizar na *Google Play* para avaliação prática do consumo de dados móveis;
3. Analisar e comparar métricas de consumo de dados entre as duas aplicações para determinar a eficácia do *Play Feature Delivery* no acesso eficiente à informação em diferentes condições de rede.
4. Analisar como o *Play Feature Delivery* pode contribuir para a sustentabilidade digital em termos de consumo de dados.

O artigo estrutura-se da seguinte forma: A seção Trabalhos Relacionados discute a sustentabilidade digital, o uso de dispositivos móveis pela população e as potenciais melhorias com a *Play Feature Delivery*. Referencial Teórico aprofunda-se nos temas mencionados no artigo. Em Materiais e Métodos, são descritas as aplicações desenvolvidas e a metodologia adotada. A seção Resultados apresenta os achados com base nos cenários propostos. Por fim, Considerações Finais reflete sobre as principais descobertas e implicações do estudo.

2. Trabalhos Relacionados

Nesta seção serão abordados artigos cujo a análise envolva o desempenho de aplicações *mobiles* com foco no sistema operacional *Android*.

A *Play Feature Delivery* é uma tecnologia em destaque devido à sua capacidade de modularizar e otimizar o tamanho de aplicativos. No estudo conduzido por Leo Ely [Ely 2019], o foco principal foi a redução do tamanho dos aplicativos por meio dessa técnica. Ely desenvolveu duas versões de uma aplicação móvel: uma com o recurso Play Feature Delivery e outra sem, para compará-las. No entanto, sua análise restringiu-se somente a esse aspecto, sem avaliar outras potenciais vantagens. Em contrapartida, Aleksandrov [Aleksandrov 2020] expandiu sua pesquisa para compreender as consequências práticas do download sob demanda em aplicações móveis reais, explorando possíveis melhorias na experiência do usuário. Embora os dois estudos ofereçam *insights* importantes sobre a Play Feature Delivery, ambos não consideram aspectos essenciais como a eficiência no consumo de dados móveis

Em uma vertente diferente, a convergência da digitalização com a sustentabilidade tem se tornado um tópico relevante. George et al. [George et al. 2021] focaram em como as ferramentas digitais podem ser utilizadas para abordar desafios sociais e promover modelos de negócios sustentáveis. Essa perspectiva ecoa os achados de Zhang [Zhang et al. 2022], que examinou o impacto da transformação digital na sustentabilidade dos negócios. Ambos os estudos destacam a importância de abordagens digitais inovadoras no enfrentamento de questões sociais e ambientais.

Já no artigo de [Rosário and Dias 2022], ele reforça que os processos de transição digital demonstraram uma enorme capacidade de desenvolver e implementar soluções sustentáveis, que permitem resolver vários problemas, como pobreza, altas taxas de extinção de espécies e falta de igualdade de oportunidades, mas que no entanto, não há muito estudos entre transição digital e a sustentabilidade. Dessa forma, o trabalho buscou demonstrar as potenciais contribuições da transição digital para aspectos de sustentabilidade ambiental, econômica e social. Ele reforça que tecnologias como IA (Inteligência Artificial), análise de *big data*, *IoT* (Internet das Coisas), mídias sociais e tecnologias móveis podem ser utilizadas para desenvolver e implementar soluções de sustentabilidade.Dessa forma, empresas, governos e instituições não governamentais podem aumentar a sustentabilidade ao adotar e integrar tecnologias digitais em diversas atividades, processos e sistemas.

Todos os trabalhos apresentados acima trazem diferentes estratégias que abordam o uso de sustentabilidade digital ou o uso da *Play Feature Delivery*. Leo Ely [Ely 2019], trouxe em seu trabalho o estudo na documentação do formato envio e devolvimento de aplicações para o sistema operacional *Android*, enfatizando a modularização dos recursos. Ele se assemelha a este artigo por comparar duas aplicações contendo a PFD, e outra que não tinha. No entanto, o diferencial é o foco de como essa *feature* pode trazer benefícios para usuários com menos recursos econômicos. Já Aleksandrov, visa entender como a temática abordada neste estudo pode melhorar a experiência do usuário ao transformar uma aplicação já existente a suportar a *Play Feature Delivery*, mas sem focar nos benefícios econômicos poderiam acarretar. Já ao se tratar de sustentabilidade digital, os trabalhos de George et al. [George et al. 2021], [Zhang et al. 2022] e [Rosário and Dias 2022] buscam entender como ela pode ajudar as empresas, governos

e instituições não governamentais podem aumentar a sustentabilidade ao adotar e integrar tecnologias digitais em diversas atividades. Destacando a importância de abordagens digitais inovadoras no enfrentamento de questões sociais e ambientais.

No entanto, o cruzar as temáticas e realizar uma análise mais profunda, nota-se uma lacuna evidente: a ausência de pesquisas que investigam os impactos da *Play Feature Delivery* sob a perspectiva da sustentabilidade digital.

3. Referencial Teórico

Neste referencial teórico, abordaremos conceitos-chave que fundamentam a temática central deste trabalho. Inicialmente, exploraremos a sustentabilidade digital, esclarecendo sua definição e importância no cenário tecnológico atual. Em seguida, direcionaremos nosso foco para o sistema operacional *Android*, dando especial atenção à sua funcionalidade de entrega de conteúdo, conhecida como *Play Feature Delivery*.

3.1. Sustentabilidade Digital

De acordo com o Fundo Mundial para a Natureza, a definição clássica de desenvolvimento sustentável é aquela que atende às necessidades do presente sem comprometer a capacidade das gerações futuras de atenderem às suas próprias necessidades [WWF 2023]. Ao se tratar do contexto digital, a definição para Sustentabilidade Digital é enriquecida por mais seis características: justiça intergeracional, capacidade regenerativa, redução de riscos, capacidade de absorção, e valor acrescido ecológico e econômico [Stuermer 2014]. Sendo assim, seu principal enfoque é debater como criar, utilizar e regular recursos digitais para maximizar seu valor para a sociedade contemporânea e futura. Neste âmbito, a sustentabilidade digital refere-se à construção de uma infraestrutura – tanto social quanto técnica – que preserve o valor dos dados sem uma degradação significativa [Bradley 2007].

Para um maior entendimento, se faz a necessidade de referenciar as seis características trazidas por Stuermer, sendo elas:

- **Justiça intergeracional:** Bens digitais, como dados, conteúdo e software, devem ser disponibilizados de forma que sua usabilidade a longo prazo seja garantida. Isso diz respeito não apenas aos dados em si, mas também ao conhecimento necessário para sua interpretação e uso. O pré-requisito para isso é uma arquitetura de informação transparente;
- **Capacidade regenerativa:** É um pré-requisito para a sustentabilidade digital que o conhecimento implícito (conhecimento tácito) sobre um bem digital resida não apenas dentro de uma pessoa ou de uma única organização. O conhecimento tácito deve ser distribuído entre muitos atores, permitindo que eles compartilhem suas inovações uns com os outros livremente;
- **Uso econômico de recursos:** Se as pessoas são excluídas do acesso à informação digital, elas precisam recriá-la para utilizá-la. Isso contradiz a ideia de uso econômico de recursos. Assim, deve ser assegurada a reutilização e distribuição técnica e legal irrestrita dos recursos digitais;
- **Redução de risco:** Os bens digitais, portanto, devem ser projetados de forma que não criem dependências em relação aos seus fabricantes, sejam confiáveis e possam ser interpretados corretamente por todos os usuários. Pré-requisito para isso é sua arquitetura de informação e transparente;

- **Capacidade de absorção:** A sociedade deve ser capaz de absorver recursos digitais para poder usá-los e adaptá-los de maneira adequada a novas necessidades e requisitos. Isso se refere a questões como estruturação compreensível, documentação, descoberta e filtragem de informações;
- **Valor acrescentado ecológico e econômico:** Os bens digitais devem ser disponibilizados gratuitamente para serem compartilhados na maior extensão possível, possibilitando o potencial para inovação e valor integral para a sociedade. Portanto, é necessário estabelecer as condições adequadas em um nível regulatório para que os recursos digitais livremente acessíveis sejam promovidos e os canais de distribuição, como a Internet, sejam igualmente acessíveis a todos.

3.2. *Android*

O sistema operacional *Android* é fundamentado em um *Kernel Linux* modificado [Butler 2010]. Trata-se de uma plataforma de código aberto para dispositivos móveis, com um projeto liderado pelo Google [Source 2023]. Nessa plataforma, os desenvolvedores podem criar aplicativos disponíveis para *download* na *Google Play*, otimizando o cotidiano dos usuários. Ainda que diversas linguagens de programação sejam compatíveis com seu desenvolvimento, *Java* e *Kotlin* são as mais predominantes.

Um programa *Android* executa um arquivo chamado $build.gradle$, segundo a documentação oficial do Gradle, ele serve como uma especificação declarativa do processo de construção para um projeto específico. Ele descreve os parâmetros, dependências, *plugins* e tarefas necessárias para a construção e configuração de um projeto. Dadas as complexidades envolvidas no ciclo de vida do desenvolvimento de *software*, o $build.gradle$ fornece um meio estruturado e extensível para definir como o código fonte é transformado em artefatos executáveis e distribuíveis [Documentation 2023].

Para o desenvolvimento de um aplicativo na plataforma *Android*, componentes essenciais como $Activities$, $Fragments$, $Services$ e $Resources$ são incorporados, sendo detalhados a seguir. De acordo com a documentação oficial *Developers*, uma $Activity$ fornece a interface visual para interação do usuário com o software. Em contraste, um $Fragment$ atua como um subcomponente modular dentro desse contexto, podendo ser reutilizado em diversas configurações, promovendo assim um design de interface mais adaptável e modular. Os $Services$ referem-se a funções executadas em segundo plano, sem interação contínua do usuário; um exemplo é um executor de música, que toca mesmo sem o app estar aberto. Por sua vez, $Resources$ compreendem arquivos estáticos, como layouts, imagens e textos, que serão exibidos tanto em interfaces principais quanto em subcomponentes modulares.

Na arquitetura padrão de um aplicativo dessa plataforma, todas as informações e funcionalidades estão centralizadas no módulo principal. A Figura 2 ilustra a dependência do módulo principal com esses componentes mencionados anteriormente.

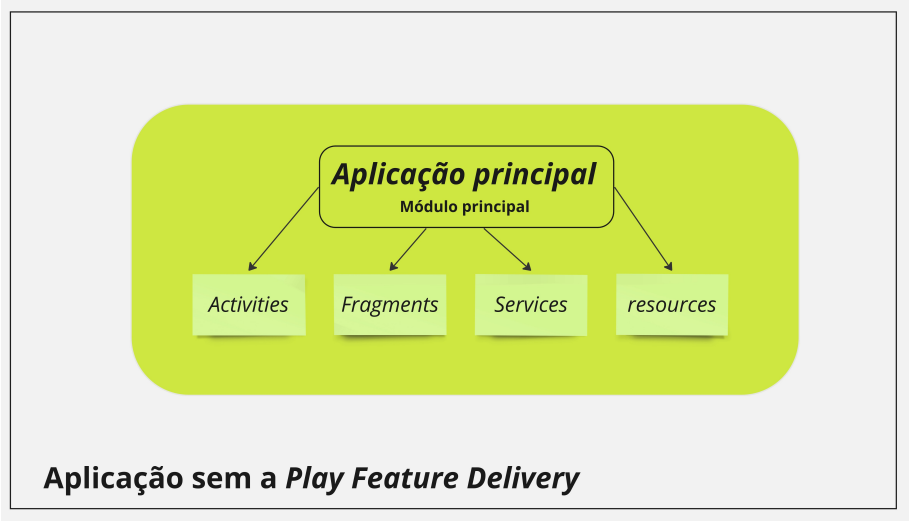

Figura 2. Ilustração da arquitetura adotada por uma aplicação tradicional.
Fonte: Autoria própria.

3.3. *Play Feature Delivery*

A *Play Feature Delivery* é um recurso que possibilita que os conteúdos sejam disponibilizados de maneira condicional ou transferidos por *download* conforme a demanda, priorizando os recursos essenciais para o funcionamento básico do aplicativo. Assim, ganha-se flexibilidade para definir como e quando diferentes aspectos da aplicação são enviados para os dispositivos [Developers 2023]. Para entender como funciona essa modularização pode ser consultado a Figura 3. Ela ilustra o fluxograma de como o usuário baixa a aplicação com os recursos básicos e sob demanda pode baixar os módulos. Esta abordagem distingue os recursos básicos da aplicação dos recursos dinâmicos.

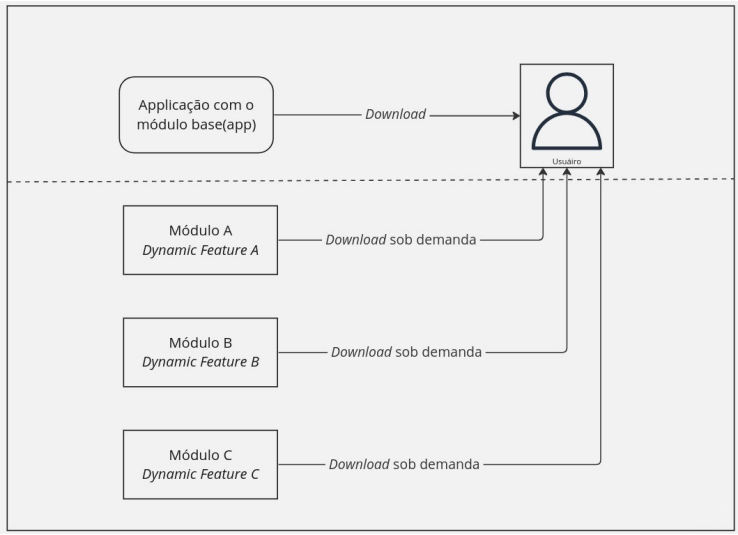

Figura 3. Ilustração fluxo adotada por uma aplicação que utiliza a *Play Feature Delivery*.
Fonte: Autoria própria.

Esse recurso divide a aplicação em Módulos de Recursos Dinâmicos (MRD), que representam funcionalidades opcionais. Na primeira instalação de um aplicativo, os usuários carregam apenas o módulo base. Posteriormente, os MRD podem ser baixados de acordo com a demanda do usuário. Para acessar esses módulos, é necessário que o usuário os solicite dentro da aplicação, que por sua vez enviará um pedido à API da *Google Play* para a instalação no dispositivo móvel.

Para que a *Play Feature Delivery* entregue esses conteúdos, ela utiliza uma classe do AOSP (*Android Open Source Project*), *SplitInstallManager*. Essa classe, representa uma parte crucial para o funcionamento do *Android App Bundles* e do sistema de entrega modular do *Google Play*, a *Play Feature Delivery*. A classe *SplitInstallManager* é uma estratégia para otimizar a entrega de aplicativos, facilitando a instalação dinâmica de módulos de um aplicativo em tempo de execução.

Dessa forma, a *Play Feature Delivery* permite que apenas o conteúdo essencial para o funcionamento da aplicação seja instalado, para assim, reduzir os recursos de transferência de dados e espaço de armazenamento.

3.4. Métricas de Desempenho

Nesta seção, discutem-se as métricas de desempenho que serão empregadas na subsequente seção de resultados para a realização de comparações entre as duas aplicações em estudo. As métricas em questão serão determinadas com base em cálculos estatísticos que incluem a média aritmética, o desvio padrão e o teste de hipóteses.

3.4.1. Média Aritmética e Desvio Padrão

A média aritmética de um conjunto de números n é calculada somando-se todos os valores individuais e dividindo-se pelo número total de elementos no conjunto. Este cálculo é formalmente expresso pela fórmula a seguir

$$\mu = \frac{\sum_{i=1}^{n} x_i}{n} \tag{1}$$

Onde:

μ Representa a média aritmética do conjunto de dados.
x_i Denota o valor individual na sequência de dados.
n Indica o total de observações ou entradas na sequência.

O desvio padrão é uma medida que expressa o grau de dispersão de um conjunto de dados, ou seja, o desvio padrão indica o quanto um conjunto de dados é uniforme, o quanto mais próximo de 0 for o desvio padrão, mais preciso são os dados [Oliveira et al. 2019]. A formula para o DP é apresentada a seguir:

$$s = \sqrt{\frac{\sum_{i=1}^{n}(x_i - \bar{x})^2}{n-1}} \tag{2}$$

Onde:

s Representa o desvio padrão da amostra.
n Refere-se ao número total de observações na amostra.
x_i Corresponde à i-ésima observação no conjunto de dados.
$\bar{x}$ É a média aritmética da amostra, dada por:

$$\bar{x} = \frac{1}{n} \sum_{i=1}^{n} x_i$$

$(x_i - \bar{x})^2$ Esta expressão calcula o quadrado da diferença entre a observação individual e a média da amostra.
$\sum_{i=1}^{n}(x_i - \bar{x})^2$ Representa a soma total dos quadrados das diferenças entre as observações e a média.
$n - 1$ Este é o divisor utilizado quando calculamos o desvio padrão de uma amostra, garantindo uma estimativa não tendenciosa do desvio padrão da população.

3.4.2. Teste de Hipóteses

O teste de hipóteses serve como um procedimento para avaliar a validade de uma afirmação, sempre associado a um determinado grau de risco de erro. Essencialmente, ele estabelece critérios para aceitar ou refutar uma hipótese com base em dados obtidos de uma amostra, implicando, assim, a possibilidade de conclusões equivocadas [Hirakata et al. 2019]. Existem dois tipos principais: paramétricos e não-paramétricos. Os testes paramétricos fundamentam-se em parâmetros específicos da amostra, como média e desvio padrão. Neste estudo, a ênfase recai sobre os testes paramétricos.

Segundo Hirakata, existem duas hipóteses, H_0 e H_1, no qual a H_0 é assumida como verdadeira até que se prove o contrário. A segunda é chamada de hipótese alternativa H_1, que representa uma afirmação de que o parâmetro de interesse difere daquele definido na hipótese nula, de modo que as duas hipóteses sejam complementares.

Para ser realizado o teste de hipótese, é preciso estabelecer um objetivo e desfecho. Após esse passo um, é preciso formular as hipóteses nula e alternativa, H_0 e H_1. Por exemplo:

- Hipótese Nula (H_0): $U_{\text{valorA}} \leq U_{\text{valorB}}$
- Hipótese Alternativa (H_1): $U_{\text{valorA}} > U_{\text{valorB}}$

Após decidir o nível de significância α, os dados são coletados e a estatística de teste é calculada. Esta estatística será comparada com uma distribuição teórica para calcular o valor-p [Hirakata et al. 2019].

A estatística t é calculada para duas amostras independentes na seguinte fórmula:

$$t = \frac{\bar{x}_1 - \bar{x}_2}{\sqrt{\frac{s_1^2}{n_1} + \frac{s_2^2}{n_2}}}$$

O valor-p é a probabilidade de obter um resultado tão extremo quanto o observado, assumindo que a hipótese nula H_0 é verdadeira. Se o valor-p for menor que o nível de significância α, então será rejeitada a hipótese nula em favor da hipótese alternativa H_1. Erros no teste de hipóteses:

- Erro Tipo I: Rejeitar H_0 quando H_0 é verdadeira.
- Erro Tipo II: Não rejeitar H_0 quando H_1 é verdadeira.

A decisão final será de rejeitar ou não a hipótese nula, baseando-se no valor-p e no nível de significância α.

4. Materiais e Métodos

Nesta seção, aborda os procedimentos metodológicos adotados durante a condução da pesquisa que embasa este estudo, desde o inicio até a fase de resultados. Para um melhor entendimento, foi desenvolvido um fluxograma, como pode ser visto na Figura 4.

Em um primeiro momento, foi elaborado um protótipo para oferecer uma visualização preliminar do design da aplicação. Em seguida, desenvolveu-se a versão tradicional da aplicação, seguida da implementação da funcionalidade *Play Feature Delivery*. Após a construção das duas versões da aplicação, uma série de testes que utilizaram métricas estatísticas para avaliar o desempenho de cada uma.

Detalhes adicionais sobre cada uma dessas etapas serão abordados nas subseções subsequentes desta seção.

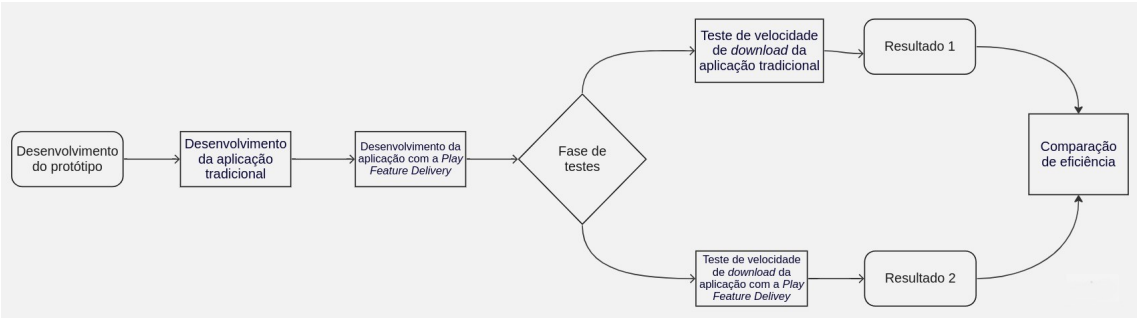

Figura 4. Fluxo para o desenvolvimento dos materiais e métodos para se obter o resultado para este trabalho.

Fonte: Autoria própria.

4.1. Fase de Prototipação

Para definir como seria a aplicação a ser implementada neste trabalho, foi optado pela criação de um protótipo. Ele é uma versão inicial de um desenvolvimento de Software que traz a clareza quais são as principais funcionalidades que uma aplicação deve possuir, ajudando a entender as necessidades para solucionar o problema. Para isso, foi utilizado o Figma, uma ferramenta voltada para a construção de produtos significativos, que permite a criação de protótipos para diversas plataformas visando coletar *feedbacks*. As principais características do protótipo desenvolvido foram:

- **Design simplificado:** Foco em uma interface de usuário intuitiva, mas básica, para entender como seria o fluxo da aplicação;
- **Validação da ideia:** Utilização da prototipação para determinar os requisitos essenciais do desenvolvimento da aplicação.

Nesta etapa do estudo, foram desenvolvidas duas interfaces gráficas, conforme ilustrado na Figura 5. A interface à esquerda representa a página inicial da aplicação. Nela, ele poderá buscar por qual situação ele precisa de dicas. Já interface à direita exibe a página com dicas selecionadas, apresentando um conjunto sequencial de orientações que o usuário pode seguir para enfrentar ou amenizar a situação em questão.

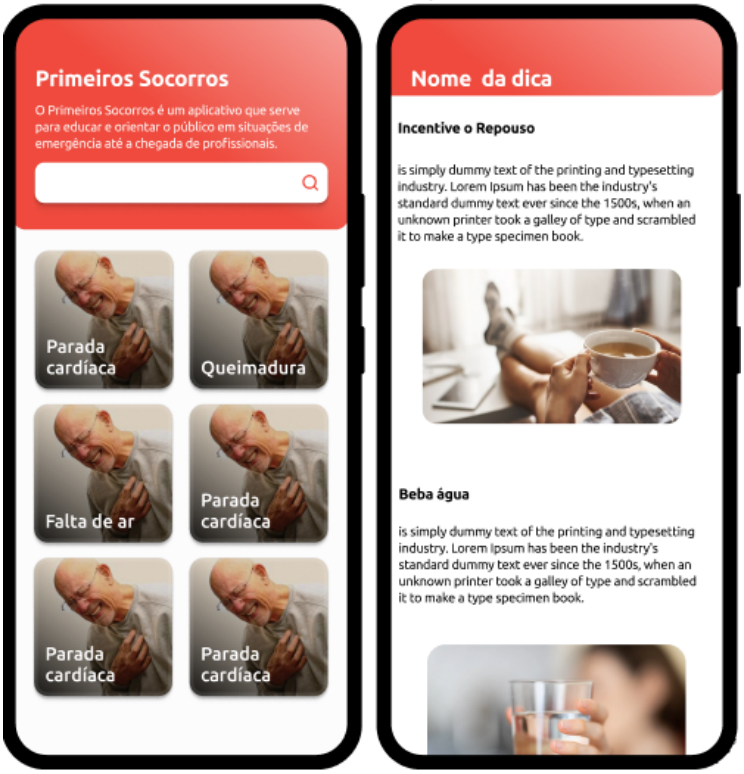

Figura 5. Prototipação da aplicação através do Figma.
Fonte: Autoria própria.

A etapa em questão desempenhou um papel fundamental no desenvolvimento do projeto. No início, não estava evidente como conceber uma aplicação que proporcionasse informações ao usuário e, ao mesmo tempo, integrasse diversas imagens. O objetivo era avaliar métricas relacionadas ao consumo de dados em redes móveis.

Antes de iniciar o desenvolvimento das aplicações, foi empregada uma tecnologia de inteligência artificial generativa para a criação de imagens que seriam integradas às aplicações. Para esta etapa, utilizou-se o Midjourney, uma plataforma avançada de geração de imagens por inteligência artificial. Desenvolvida e mantida pelo laboratório de pesquisa independente Midjourney, Inc., sediado em São Francisco, esta ferramenta possui a capacidade de gerar imagens altamente customizadas com base em descrições em linguagem natural, conhecidas como *prompts*, o qual são frases ou comandos definidos de forma clara e contextual, que servem como instruções para que o algoritmo do Midjourney produza imagens conforme as especificações desejadas.

A utilização do Midjourney permitiu uma maior flexibilidade e precisão na geração de imagens, contribuindo para a qualidade estética e funcional das aplicações desenvolvidas. A seguir, são apresentados alguns exemplos de *prompts* que foram utiliza-

dos para gerar as imagens:

- Imagine um desfibrilador portátil realista em um hospital;
- Imagine uma reanimação cardiopulmonar, no estilo da técnica tradicional onde há um homem deitado no chão e um enfermeiro executando a reanimação.

Após rodar os *prompts*, ele gerava as imagens desejadas. Nas Figuras 6 e 7 podem ser observados de acordo com as frases listadas acima.

Figura 6. Imagem gerada de acordo com o *prompt*: Imagine um desfibrilador portátil realista em um hospital.
Fonte: Autoria própria.

Figura 7. Imagem gerada de acordo com o *prompt*: Imagine uma reanimação cardiopulmonar, no estilo da técnica tradicional onde há um homem deitado no chão e um enfermeiro executando a reanimação cardiopulmonar.
Fonte: Autoria própria.

As imagens criadas pela Midjourney trazem um teor realista à aplicação. A adoção dessa ferramenta revelou-se imprescindível, considerando que, para avaliar a aplicação

utilizando a *Play Feature Delivery*, era mandatório publicá-la na *Google Play*. A fim de prevenir complicações associadas a potenciais violações de *copyright*, optou-se pela utilização do modelo de Inteligência Artificial fornecido pela Midjourney.

4.2. Desenvolvimento das Aplicações

Para o desenvolvimento das duas aplicações, foi escolhida a linguagem de programação *Kotlin*. Além do mais, a *IDE* de desenvolvimento escolhida foi o *Android Studio*.

4.2.1. Desenvolvimento Aplicação Tradicional

Para inicializar a aplicação foi preciso criar um novo projeto no *Android Studio*. Ao ser criada, ela é desenvolvida inteiramente no módulo *app*, responsável por iniciar uma aplicação *Androcardiopulmonarid*, sendo a arquitetura tradicional. Para um maior entendimento, foi desenvolvido um diagrama de classe com as principais classes da aplicação sem a *Play Feature Delivery*. Na Figura 8, são listadas as seguintes classe: $MainActivity$, $ItemActivity$, $SocorrosAdapater$, $ItemAdapter$, $MainAdapterItems$, e $FragmentItems$. Essas classes serão desenvolvidas ao longo desta subseção.

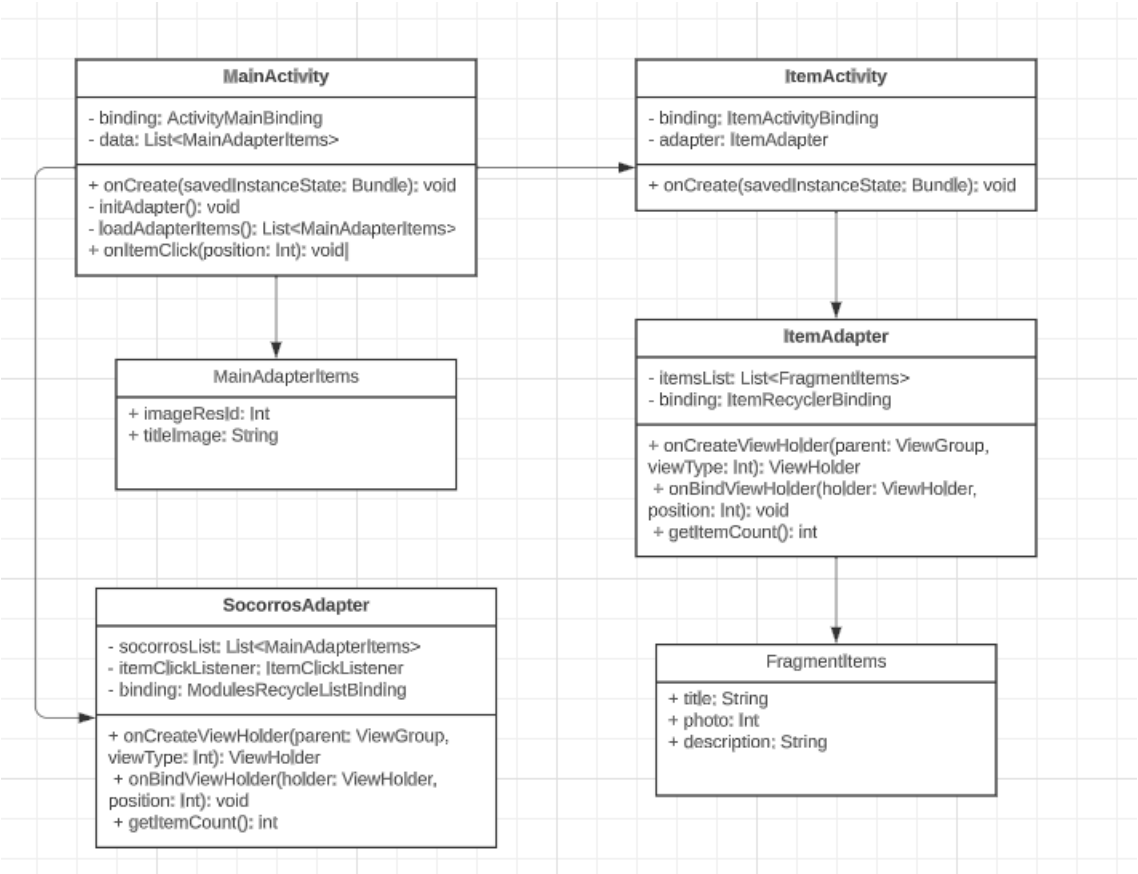

Figura 8. Diagrama com as principais classes da aplicação tradicional.
Fonte: Autoria própria.

Após a etapa de criação do projeto, é gerada uma classe automática, no qual é destinada à entrada da aplicação é inicializada, a $MainActivity$. Esta classe contém o

método $onCreate$, que é encarregado de estabelecer todos os dados essenciais para a inicialização da aplicação como pode ser observado no Quadro 1.

Quadro 1. Método $onCreate$.

```
override fun onCreate(savedInstanceState: Bundle?) {
    super.onCreate(savedInstanceState)
    binding = ActivityMainBinding.inflate(layoutInflater)
    setContentView(binding.root)
    stringUtil = ResourceStringReaderUtil(this)
    data = loadAdapterItems()
    initAdapter()
}
```

Quando o método $onCreate$ é chamado, a $Activity$ é inicializada conforme o *layout* criado na prototipação como pode ser visto na Figura 9. Para que esse *layout* da $MainActivity$ seja organizado de forma correta, ele chama o método $loadAdapterItems$ que cria uma lista de itens da classe $MainAdapterItems$. Como mostra no diagrama da Figura 8, a classe $MainAdapterItems$, tem apenas dois atributos que servem para identificar a foto e o título da dica que irá ser exibido na tela principal da aplicação. As informações atribuídas aos objetos gerados na lista do tipo $MainAdapterItems$, são enviados para a classe $SocorroAdapter$ que é responsável por criar a listagem das dicas exibidas na tela principal.

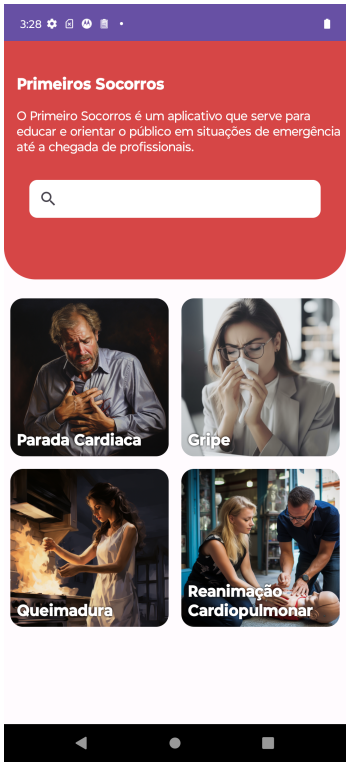

Figura 9. Design da aplicação tradicional.
Fonte: Autoria própria.

Para gerenciar a ação de informações de cada item, um *listener* de toque é

adicionado. Ao ser acionado, ele chama o método no Quadro 2, que identifica a posição do item e direciona para a classe apropriada, redirecionando para o acesso ao conteúdo escolhido pelo usuário.

Quadro 2. Método $onItemClick$.

```
    override fun onItemClick(position: Int) {
        val intent = ItemActivity.getIntent(this, position)
        startActivity(intent)
    }
```

Quando o usuário é redirecionado para a nova $Activity$, é iniciada a classe $ItemActivity$. É nessa classe que irá criar os conteúdos das dicas de primeiro socorros de acordo com a posição que foi passada através da $MainActivity$, para isso, ele chama o método $onCreate$ da $ItemActivity$ no Quadro 3.

Quadro 3. Método $onCreate$ na $ItemActivity$.

```
override fun onCreate(savedInstanceState: Bundle?) {
    super.onCreate(savedInstanceState)
    binding = ItemActivityBinding.inflate(layoutInflater)
    setContentView(binding.root)
    binding.recyclerView.layoutManager = LinearLayoutManager(
       this)

    when(intent.getIntExtra(POSITION,0)) {
        0 -> {
            adapter = ItemAdapter(ParadaCardiaca.getTips(this))
            binding.recyclerView
            binding.activityTitle.text = getString(R.string.
               parada_title) }
        1 -> {
            adapter = ItemAdapter(Gripe.getTips(this))
            binding.activityTitle.text = getString(R.string.
               gripe_title) }
        2 -> {
            adapter = ItemAdapter(Queimadura.getTips(this))
            binding.activityTitle.text = getString(R.string.
               queimadura_titulo) }
        3 -> {
            adapter = ItemAdapter(CRP.getTips(this))
            binding.activityTitle.text = getString(R.string.
               cpr_title) }
        else -> {}
    }
    binding.recyclerView.adapter = adapter
}
```

É relevante destacar que a responsabilidade de instanciar a classe $ItemAdapter$ pertence ao método $onCreate$ presente na classe $ItemActivity$. Esta última atua como

um núcleo central para manipulação de listas do tipo $FragmentItems$. Tais listas armazenam os valores que serão exibidos ao usuário, baseando-se na dica selecionada. A adoção desta estratégia busca reduzir a criação de múltiplas classes. Dessa forma, a classe $ItemActivity$ é invocada e adaptada dinamicamente conforme o conteúdo escolhido.

4.2.2. Desenvolvimento da Aplicação com a *Play Feature Delivery*

Para que a aplicação consiga utilizar a *Play Feature Delivery*, é necessário adicionar no arquivo $build.gradle$ do projeto, um *plugin* do recurso identificando que o projeto utilizará a *feature* $plugins\ \{id\ 'com.android.dynamic-feature'\}$.

Após a estrutura base da aplicação ser estabelecida, conforme demonstrado no exemplo anterior sem a *feature* e o *plugin* ter sido adicionado ao $build.gradle$, pode-se integrar a *Play Feature Delivery*. Para realizar essa integração, é necessário inicialmente criar os módulos no *Android Studio*. Cada módulo de *Play Feature Delivery* opera de forma autônoma, permitindo que a aplicação principal mantenha apenas o conteúdo essencial. Esse conteúdo adicional pode ser baixado posteriormente, conforme a necessidade do usuário já visto na seção do referencial teórico na Figura 2. Nesta seção, a Figura 10 ilustra a arquitetura da aplicação que emprega a *Play Feature Delivery*. É possível observar que os módulos sob demanda, módulo de queimadura, gripe, parada cardíaca e reanimação cardiopulmonar, têm uma dependência direcional do módulo app, ou seja, sua existência é condicionada ao módulo app. No entanto, o módulo app tem a capacidade de operar independentemente, mesmo na ausência desses módulos sob demanda.

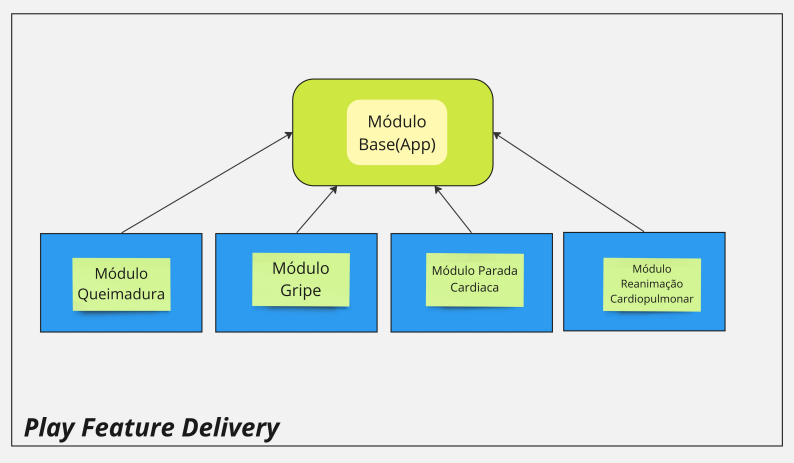

Figura 10. Adição dos módulos da *Play Feature Delivery*.
Fonte: Autoria própria.

Ao criar os módulos na aplicação no *Android Studio* na Figura 11, percebe-se a semelhança com a Figura 10. Cada módulo é responsável por exibir detalhadamente dicas sobre procedimento de primeiros socorros. Dentro de cada um tem seus próprios $Resources$, $Activities$ e $Fragments$.

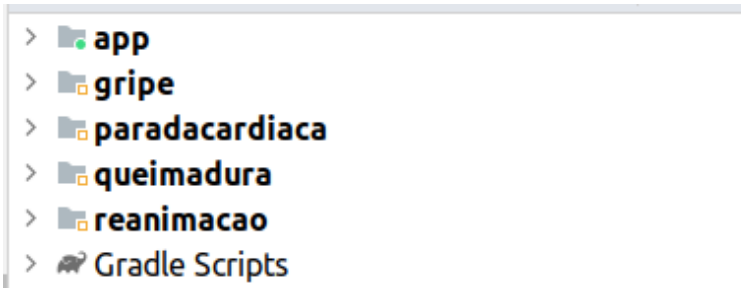

Figura 11. Modularização do projeto com a *Play Feature Delivery*.
Fonte: Autoria própria.

No aprimoramento da aplicação, cada conjunto de conteúdo que anteriormente estava integrado ao módulo *app* agora foi modularizado. Essa abordagem permite que os usuários baixem apenas os módulos de que necessitam, resultando em uma aplicação mais leve e eficiente. A implementação dessa funcionalidade é realizada conforme a seleção do usuário. A seção de código subsequente no quadro a seguir, ilustra como essa modularização foi concretizada para que ele escolha qual módulo gostaria de baixar.

Quadro 4. Método $downloadModule$ com o *callback* de resposta.

```
private fun downloadModule(module,Name: String, position: Int) {
    Log.d(TAG, "StartdownloadModule $moduleName")
    dynamicFeatureManager.downloadModule(
        moduleName,
        onCompleted = {
            Toast.makeText(this, "Module $moduleName installed
               successfully",

            Toast.LENGTH_SHORT).show()
            sharedPreferences.edit().putBoolean("item$position",
                true).apply()
        },
        onError = { exception ->
            Toast.makeText(this, "Failed to install module
               $moduleName: " +
                    "${exception.message}", Toast.LENGTH_LONG).
                       show()
        }
    )
}
```

Dado o método do Quadro 4, é crucial destacar a importância do *callback* que indica a conclusão do *download* do módulo. Após sua conclusão, o módulo selecionado é armazenado no dispositivo. Este armazenamento é realizado através do *Shared Preference*, uma implementação específica do *Android* para o design de padrão de armazenamento de dados baseado em chave-valor. Tal implementação fornece uma interface refinada para a persistência de tipos primitivos de dados, como `booleanos`, `floats`, `ints`, `longs` e `strings`, em um arquivo privado associado a um contexto

específico da aplicação. É importante notar que estes arquivos são alocados no sistema de arquivos interno do dispositivo, vinculados a cada aplicativo, assegurando assim a integridade, privacidade e segurança dos dados [Developers 2023].

Quadro 5. Método $downloadModule$ da API do Google para indicar qual módulo deseja baixar e seu estado de *download*.

```
fun downloadModule(moduleName: String, onCompleted: () -> Unit,
   onError: (Exception) -> Unit) {
    val request = SplitInstallRequest.newBuilder()
        .addModule(moduleName)
        .build()

    val listener = SplitInstallStateUpdatedListener { state ->
        if (state.moduleNames().contains(moduleName)) {
            when (state.status()) {
                SplitInstallSessionStatus.DOWNLOADING -> {
                    Log.d("DynamicFeatureManager", "Downloading
                       the module")
                }
                SplitInstallSessionStatus.INSTALLED -> {
                    Log.d("DynamicFeatureManager", "Module
                       installed")
                    onCompleted()
                }
                SplitInstallSessionStatus.FAILED -> {
                    onError(Exception("Error installing module
                       $moduleName: ${state.errorCode()}"))
                }
            }
        }
    }

    splitInstallManager.registerListener(listener)
    splitInstallManager.startInstall(request)
        .addOnFailureListener { exception ->
            splitInstallManager.unregisterListener(listener)
            onError(exception)
        }
}
```

O Quadro 5, ilustra o processo de solicitação para o download de um módulo. Inicialmente, uma solicitação é configurada por meio da *SplitInstallRequest*, que é uma classe interna da API do *Android*. Esta classe é responsável por montar a requisição contendo informações sobre o módulo desejado pelo usuário. Para monitorar o progresso do *download*, um *listener* é estabelecido. No código em questão, são identificados três estados possíveis para a requisição: *Downloading* (Baixando), *Installed* (Instalado) e *Failed* (Falhou). Quando o módulo é baixado e instalado com sucesso, a resposta é então enviada através do *callback* ilustrado no código do $downloadModule$ visto

anteriormente, no $onCompleted$ ou no $onError$.

Quadro 6. Método $handleIntent$ lidar com a ação necessária ao usuário escolher o módulo.

```
private fun handleIntent(moduleName: String, activityPath:
    String) {
     if (dynamicFeatureManager.getInstalledModule().contains(
        moduleName)) {
         Toast.makeText(this, "Module $moduleName installed
            successfully",
             Toast.LENGTH_SHORT).show()
         val intent = Intent()
         intent.setClassName(this, activityPath)
         SplitCompat.installActivity(this)
         try {
             Log.d(TAG, "starting activity")
             startActivity(intent)
         } catch (e: ActivityNotFoundException) {
             Log.e(TAG, "Handling error $e")
         }
     } else { }
}
```

No *snippet* acima, no Quadro 6, é feita uma condicional para ler quais módulos já foram instalados, permitindo que o usuário acesse o conteúdo baixado sob demanda.

Por fim, *design* da aplicação utilizando a *Play Feature Delivery* pode ser observado na Figura 12. A mudança foi feita na ação do clique para baixar o conteúdo como mencionado anteriormente no método $downloadModule()$.

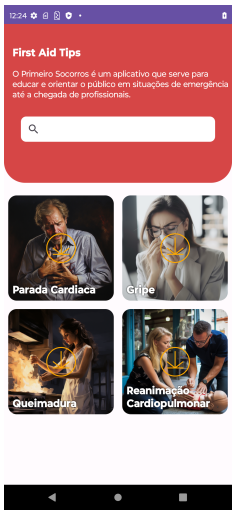

Figura 12. Design da aplicação com a *Play Feature Delivery*.
Fonte: Autoria própria.

Com o ambiente devidamente preparado, foi realizada a execução do experimento, que foi replicado 30 vezes para cada uma das aplicações em análise. A análise subsequente dos dados coletados será focada no cálculo da média e do desvio padrão para ambas as aplicações. Posteriormente, será conduzido um teste de hipótese para determinar se os resultados obtidos para as duas aplicações sob investigação apresentam similaridades estatisticamente significativas ou discrepâncias.

Após as etapas de desenvolvimento e execução dos testes, a próxima seção apresentará uma análise detalhada dos resultados obtidos. Esta metodologia permitiu compreender profundamente os benefícios e desafios associados à implementação da *Play Feature Delivery* em uma aplicação *Android*.

5. Resultados

O objeto de análise desta pesquisa abrange a inserção de duas aplicações específicas na *Google Play*. Para efetuar os testes, empregou-se um aparelho celular da marca Motorola, modelo Moto G(53), equipado com 6 *gigabyte* de memória *RAM* e operando na versão 13 do *Android*, a mais recente disponível no mercado. A rede móvel da operadora Claro, na região metropolitana do Recife, serviu como meio de acesso à internet.

O propósito principal desses testes centra-se na mensuração da taxa de dados móveis consumidos durante o download das aplicações em questão. Entende-se por taxa de dados móveis a quantidade de dados consumidos por uma aplicação durante sua operação. Dessa forma, para garantir um ambiente de teste controlado, efetuaram-se ajustes específicos no dispositivo móvel utilizado. O aparelho foi configurado para operar exclusivamente na rede 4G, de modo a eliminar qualquer variável que pudesse afetar o desempenho da conexão. Adicionalmente, desativaram-se todas as atualizações automáticas do sistema operacional para prevenir possíveis interferências nos valores de uso de dados móveis. O monitoramento foi realizado através do próprio Sistema Operacional *Android* que disponibiliza os dados móveis gastos em um determinado período de tempo como pode ser observado na Figura 13.

5.1. Cálculo da Média e do Desvio Padrão

Nesta subseção serão detalhadas como foram realizados os cálculos para se obter a média e o desvio padrão através do *Download* das aplicações através da *Google Play*. Primeiramente será feito o cálculo para a aplicação sem o recurso e logo em seguida com a *Play Feature Delivery*. As fórmulas utilizadas para obter os resultados estão descritas na seção de Referencial Teórico.

A quantidade de MB necessários para a transferência dos dados em 30 experimentos para a aplicação tradicional foram, em ordem: 15, 14, 14, 14, 15, 14, 15, 14, 15, 15, 14, 15, 15,14 , 15, 14, 15, 14, 15, 15, 14, 15, 14, 15, 15, 14, 14, 15, 14, 15. Assim, a média e o desvio padrão da amostra são respectivamente $14{,}53$ e $0{,}51$.

Agora que já se sabe a média e o desvio padrão para a aplicação tradicional, é preciso calcular esses valores para a aplicação que utiliza a *Play Feature Delivery*. Seguindo a mesma lógica será assumido a quantidade de MB necessários para a transferência dos dados em 30 experimentos foram, em ordem: 8, 8, 7, 7, 7, 7, 7, 7, 7, 8, 7, 7, 7, 7, 7, 7,

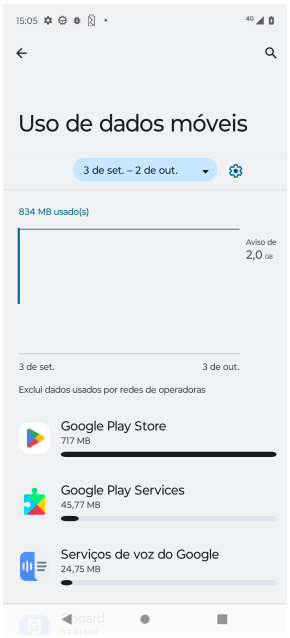

Figura 13. Monitoramento do uso de dados móveis através do Sistema *Android*.
Fonte: Autoria própria.

7, 7, 7, 8, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7. Portanto, a média dos valores obtidos ao executar o *download* são em média de 7,13 e o desvio padrão amostral de aproximadamente 0,35.

5.2. Teste de Hipótese para Comparar o Consumo dos Dados Móveis Entre as Duas Aplicações

Nesta seção será aplicada a técnica de Teste de Hipóteses para avaliar a validade do problema sugerido neste trabalho, para validar se o uso de *Play Feature Delivey* de fato pode ajudar na sustentabilidade digital. As hipóteses sugeridas que este trabalho irá aplicar serão o seguinte, tendo um nível de significância $\alpha = 0.05$, onde $\bar{x}_1$ é o app normal e $\bar{x}_2$ a aplicação com a *Play Feature Delivery*:

- Hipótese Nula (H_0): $\bar{x}_1 \leq \bar{x}_2$ (A *Play Feature Delivery* não tem um efeito significativo no uso de dados móveis)
- Hipótese Alternativa (H_a): $\bar{x}_1 > \bar{x}_2$ (A *Play Feature Delivery* resulta em uma redução significativa no uso de dados móveis em comparação com a versão sem essa funcionalidade.)

Conforme descrito nas seções anteriores, os dados amostrais relativos às duas aplicações em análise são os seguintes:

Primeira Amostra (aplicação tradicional):

- Média: $\bar{x}_1 = 14{,}53$
- Desvio padrão: $s_1 = 0{,}51$
- Tamanho da amostra: $n_1 = 30$

Segunda Amostra (aplicação com a *Play Feature Delivery*):

- Média: $\bar{x}_2 = 7{,}13$
- Desvio padrão: $s_2 = 0{,}35$

- Tamanho da amostra: $n_2 = 30$

Com os dados todos listados, é preciso fazer o cálculo da estatística de testes:

$$t = \frac{\bar{x}_1 - \bar{x}_2}{\sqrt{\frac{s_1^2}{n_1} + \frac{s_2^2}{n_2}}}$$

Primeiro serão calculados os componentes da fórmula:

$$\bar{x}_1 - \bar{x}_2 = 14{,}53 - 7{,}13 = 7{,}4 \tag{3}$$

$$\frac{s_1^2}{n_1} = \frac{(0{,}51)^2}{30} = \frac{0{,}2601}{30} = 0{,}00867 \tag{4}$$

$$\frac{s_2^2}{n_2} = \frac{(0{,}35)^2}{30} = \frac{0.0{,}1225}{30} = 0{,}0040833 \tag{5}$$

Após ser calculado os componentes, pode-se calcular o denominador:

$$\sqrt{\frac{s_1^2}{n_1} + \frac{s_2^2}{n_2}} = \sqrt{0{,}00867 + 0{,}0040833} = \sqrt{0.0127533} \approx 0{,}1129 \tag{6}$$

Com todos os caculos realizados para o numerador e denominador, é realizado o calculo para t:

$$t = \frac{7{,}4}{0{,}1129} \approx 65{,}55 \tag{7}$$

Desta forma, o calculo da estatística de teste para esses dados amostrais é aproximadamente $65{,}55$. Podendo assim calcular os graus de liberdade e o valor-p.

Os graus de liberdade para este teste foram calculados usando a fórmula $n_1 + n_2 - 2$, resultando em 58 graus de liberdade como pode ser visto na equação abaixo:

$$30 + 30 - 2 = 58 \tag{8}$$

Ademas, para o cálculo do valor-p, foi realizado um pequeno *script* na linguagem de programação *Python* que utilizou a biblioteca *scypy* e o módulo de estatística *stats* para determinar correto para os valores da tabela de t *student*. O resultado do valor-p foi de:

$$2.135327319216739 \times 10^{-36}$$

Esse resultado é extremamente próximo de zero. Em termos práticos, este valor pode ser considerado zero para todos os propósitos de teste de hipóteses.

Quadro 7. *script* do código em Python para o cálculo do valor-p.

```
from scipy.stats import t

df = 58
t_value = 65.55

p_value = t.sf(t_value, df)
print("Valor-p:", p_value)
```

Conforme ilustrado na Figura 14, valores da estatística t próximos de zero são considerados normais ou esperados quando a hipótese nula é verdadeira. Por outro lado, valores de t afastados de zero são considerados atípicos ou inesperados, sugerindo que a hipótese nula possa não ser válida. Na presente pesquisa, observou-se um valor de t aproximadamente $65{,}55$, um resultado extremamente distante de zero. Isso fornece um forte indício contra a validade da hipótese nula, que sustenta que a *Play Feature Delivery* tem o uso menor no uso de dados móveis comparando-se a aplicação tradicional.

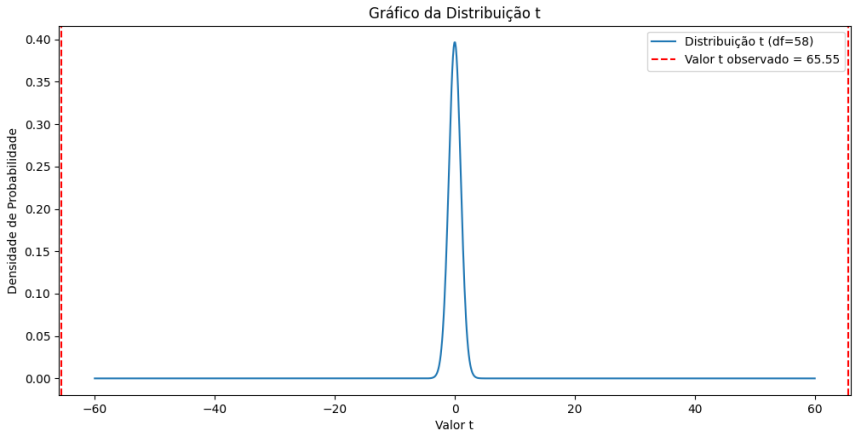

Figura 14. Gráfico da distribuição no valor t.
Fonte: Autoria própria.

Ademais, este trabalho traz evidências suficientes para rejeitar a hipótese nula e concluir que a versão da aplicação com o recurso da *Play Feature Delivery* usa significativamente menos dados móveis em comparação com a versão sem o recurso. Esses dados realçam o quão importante o uso dessa *feature* por desenvolvedores *Android* pode contribuir para uma maior democratização do acesso à informação. Dessa forma, pessoas com conexões de internet mais lentas ou planos de dados mais limitados têm acesso mais fácil a informações importantes, tornando a aplicação mais inclusiva, contemplando alguns dos pilares principais da Sustentabilidade Digital trazidos anteriormente.

6. Considerações Finais e Trabalhos Futuros

O estudo realizou uma investigação detalhada sobre o impacto da *Play Feature Delivery* no consumo de dados móveis. Os resultados, evidenciados pelo valor da estatística t de aproximadamente $65{,}55$ e pelo teste de hipótese que invalida a hipótese nula, demonstram claramente que a *Play Feature Delivery* exerce um efeito considerável na redução do uso

de dados móveis. Tal descoberta não só ressalta a relevância técnica para desenvolvedores *Android* como também destaca sua importância social.

Em uma era digital, a democratização do acesso à informação torna-se primordial. Assegurar que aplicações sejam acessíveis a todos, independentemente da qualidade da conexão à internet ou das restrições de seu plano de dados, representa uma valiosa contribuição para uma sociedade mais inclusiva. Este estudo sublinha como decisões técnicas aparentemente simples, como a adoção da *Play Feature Delivery*, podem gerar impactos profundos, alinhando-se aos preceitos da Sustentabilidade Digital.

No entanto, apesar dos benefícios da *Play Feature Delivery* ressaltados pelo estudo, é essencial abordar seus potenciais impactos negativos. Para os desenvolvedores, a integração dessa funcionalidade pode trazer uma camada adicional de complexidade à arquitetura do aplicativo. Adaptar-se a este novo sistema e garantir que todos os módulos funcionem harmoniosamente requer uma compreensão profunda e pode resultar em maiores horas de desenvolvimento e testes. Para os usuários, embora a *Play Feature Delivery* possa reduzir o uso inicial de dados móveis, ela pode ser problemática em áreas com conectividade limitada ou inexistente. Um usuário que descarrega um aplicativo em um ambiente conectado e, posteriormente, deseja acessar novos módulos em uma área sem rede, pode se deparar com a impossibilidade de obter as funcionalidades desejadas.

A proposta central deste artigo, que busca promover o acesso à informação em dispositivos móveis como um pilar da sustentabilidade digital, foi corroborada pelos dados coletados. Essa constatação destaca como a sociedade deve continuamente adaptar e melhorar os recursos digitais de acordo com emergentes necessidades e cenários. Tal perspectiva destaca como escolhas técnicas podem reverberar em consequências sociais amplas, fomentando uma sociedade mais inclusiva e sustentável. Assim, aqueles que se encontram predominantemente dependentes de conexões móveis para seu acesso à internet têm a oportunidade de gerir seus dados de maneira mais eficaz. Este estudo demonstrou sua eficácia ao revelar uma notável diminuição no consumo de dados móveis. No entanto, para uma compreensão mais completa sobre a *Play Feature Delivery*, sugere-se explorar as seguintes direções em pesquisas futuras:

- **Análise de Aplicações Variadas:** Verificar o comportamento da PFD em diferentes aplicações móveis, a fim de determinar se a eficácia na economia de dados se mantém constante ou apresenta variações conforme as características específicas de cada aplicação.
- **Estudo de Dados Transmitidos:** Realizar experimentos variando o volume de dados transmitidos. Esta investigação permitirá compreender se a economia proporcionada pela PFD é influenciada pelo tamanho dos pacotes de dados transmitidos.
- **Performance em Diversos Contextos de Conexão:** Avaliar o desempenho da PFD em variados cenários de conexão, desde redes móveis até conexões *Wi-Fi* de alta velocidade para saber o tempo de *Download* em diretentes tipos de rede.

Referências

Aleksandrov, A. (2020). Instant delivery of commercial android applications with "google play instant.

Bradley, K. (2007). Defining digital sustainability. *Library Trends*, 56(1):148–163.

Butler, M. (2010). Android: Changing the mobile landscape. *IEEE pervasive Computing*, 10(1):4–7.

Dapp, M. (2013). The 2013 open reader stories and articles inspired by okcon 2013: Open data, broad, deep, connected buch & netz-online bücher.

Developers, A. (2023). Play feature delivery. `https://developer.android.com/guide/playcore/feature-delivery`.

Documentation, G. O. (2023). What is gradle? Acessado em: 2023-07-23.

Ely, L. (2019). App bundle e dynamic delivery: modulação de apps para android.

George, G., Merrill, R. K., and Schillebeeckx, S. J. (2021). Digital sustainability and entrepreneurship: How digital innovations are helping tackle climate change and sustainable development. *Entrepreneurship Theory and Practice*, 45(5):999–1027.

Hirakata, V. N., Mancuso, A. C. B., and de Jezus Castro, S. M. (2019). Teste de hipóteses: perguntas que você sempre quis fazer, mas nunca teve coragem. *Clinical and Biomedical Research*, 39(2).

Laricchia, F. (2023). Market share held by mobile operating systems in brazil. `https://www.statista.com/statistics/262167/market-share-held-by-mobile-operating-systems-in-brazil/`. Accessed: 2023-07-23.

Moura, L. and Camargo, G. (2020). Impacto econômico e social do android no brasil. *Bain & Company,[S. l.]*, pages 4–6.

Oliveira, C. G., De Macêdo, V. d. T. L., Soares, F. C. M., et al. (2019). Desvio padrão e imprecisão de leitura: Paquímetro. *Caderno de Graduação-Ciências Exatas e Tecnológicas-UNIT-SERGIPE*, 5(3):27–27.

Rosário, A. T. and Dias, J. C. (2022). Sustainability and the digital transition: A literature review. *Sustainability*, 14(7):4072.

Source, G. A. (2023). Android open source project. `https://source.android.com/`.

Stuermer, M. (2014). Characteristics of digital sustainability. In *Proceedings of the 8th international conference on theory and practice of electronic governance*, pages 494–495.

WWF (2023). Desenvolvimento sustentável.

Zhang, C., Chen, P., and Hao, Y. (2022). The impact of digital transformation on corporate sustainability-new evidence from chinese listed companies. *Frontiers in Environmental Science*, 10:1047418.