UNIVERSIDADE ESTADUAL PAULISTA "JÚLIO DE MESQUITA FILHO"

FACULDADE DE CIÊNCIAS – CAMPUS BAURU

DEPARTAMENTO DE COMPUTAÇÃO

BACHARELADO EM CIÊNCIA DA COMPUTAÇÃO

JOÃO PEDRO PACCOLA DA SILVA

APLICATIVO ANDROID PARA COMPARTILHAMENTO DE TRANSPARÊNCIAS EM TEMPO REAL

BAURU

2019

JOÃO PEDRO PACCOLA DA SILVA

APLICATIVO ANDROID PARA COMPARTILHAMENTO DE TRANSPARÊNCIAS EM TEMPO REAL

Trabalho de Conclusão de Curso do Curso de Bacharelado em Ciência da Computação da Universidade Estadual Paulista “Júlio de Mesquita Filho”, Faculdade de Ciências, campus Bauru.

Orientador: Prof. Dr. Renê Pegoraro

BAURU

2019

Paccola da Silva, João Pedro

Aplicativo Android para compartilhamento de transparências em tempo real / João Pedro Paccola da Silva, 2019

Orientador: Renê Pegoraro

Trabalho de Conclusão de Curso (TCC) – Universidade Estadual Paulista. Faculdade de Ciências, campus Bauru, 2019

1. Aplicativo Móvel 2. Flutter 3. Android 4. Linguagem Dart I. Universidade Estadual Paulista “Júlio de Mesquita Filho”. II. Faculdade de Ciências.

JOÃO PEDRO PACCOLA DA SILVA

APLICATIVO ANDROID PARA COMPARTILHAMENTO DE TRANSPARÊNCIAS EM TEMPO REAL

Trabalho de Conclusão de Curso do Curso de Bacharelado em Ciência da Computação da Universidade Estadual Paulista “Júlio de Mesquita Filho”, Faculdade de Ciências, campus Bauru.

BANCA EXAMINADORA

Prof. Dr. Renê Pegoraro - Orientador

Faculdade de Ciências – UNESP/Bauru

Prof. Dra. Simone das Graças Domingues Prado

Faculdade de Ciências – UNESP/Bauru

Prof. Dr. Wilson Massashiro Yonezawa

Faculdade de Ciências – UNESP/Bauru

Bauru, 11 de Junho, 2019

Agradecimentos

Agradeço primeiramente minha família que me deu total apoio em todas minhas decisões. Sempre me ensinou a buscar o que é especial da maneira mais nobre e corajosa, pelo estudo.

Agradeço aos professores que me mostraram vários caminhos para que eu possa trilhar a partir de agora o meu próprio futuro.

Agradeço aos meus amigos e colegas, cada um que conheci, em todos esses anos participando do meio acadêmico, pelos momentos, risadas, companhia e incentivo.

Agradeço aos meus amigos e colegas, que conheci fora do meio acadêmico, por cada momento de apoio, consolo, conversa e participação de forma implícita nesse trabalho.

Ao Prof. Dr. Renê Pegoraro, pelo tempo que trabalhamos juntos e pelo apoio e confiança durante todo o curso e especialmente durante esse trabalho.

Meu eterno carinho e gratidão a todos.

São nossas escolhas que mostram o que realmente somos, muito mais do que nossas habilidades.

(J.K. Rowling)

Resumo

As tecnologias móveis para o setor educacional criam novas possibilidades e facilitam a realidade e o dia a dia da área educativa. Contudo, mesmo com a modernização dos ambientes de estudo, empecilhos e imprevistos podem acontecer. Neste projeto foi desenvolvido um aplicativo móvel com objetivo de facilitar a apresentação e compartilhamento de transparências e imagens, em tempo real, entre um apresentador e seus espectadores. O aplicativo foi desenvolvido para o sistema operacional *Android*, utilizando o *framework Flutter* e a linguagem de programação *Dart*.

Palavras-chave: Aplicativo. Android. Dispositivos Móveis. Tecnologia na Educação. Flutter. Dart.

Abstract

Mobile technologies for the education sector creates new possibilities and facilitates the reality and the day-to-day of education. However, even with the modernization of study environments, obstacles and unforeseen events can occur. In this project a mobile application was developed with the objective of facilitating the presentation and sharing of transparencies and images, in real time, between presenter and spectators. The application was developed for the *Android* operating system, using the *Flutter* framework and the *Dart* programming language.

Keywords: App. Android. Mobile Devices. Technology in Education. Flutter. Dart.

Lista de Ilustrações

Sumário

1. INTRODUÇÃO

O ano de 2007, marcado pelo lançamento do primeiro dispositivo móvel da *Apple*, o *iPhone*, revolucionou o mercado de telecomunicações mundial ao reinventar e redefinir o conceito de *smartphone*. Observou-se o que foi considerado por alguns autores como o início de uma nova era, pautada por alterações profundas no modo como as pessoas se relacionam e, consequentemente, no modo como as empresas se relacionam com as pessoas (COUTINHO, 2014).

O Brasil superou a marca de um *smartphone* por habitante e conta com aproximadamente 220 milhões de celulares inteligentes ativos, de acordo com a 29ª Pesquisa Anual de Administração e Uso de Tecnologia da Informação nas Empresas, realizada pela Fundação Getúlio Vargas de São Paulo (FGV-SP), no dia 19 de abril de 2018. A pesquisa indica que em maio de 2018 o Brasil tinha 306 milhões de dispositivos portáteis em uso – além de *smartphones*, o número inclui *notebooks* e *tablets*. Segundo o levantamento, em dezembro de 2017 o país atingiu a marca de 210 milhões de habitantes (MEIRELES, 2018 apud LIMA, 2018).

Trazer essas tecnologias móveis para o setor educacional gera grandes desafios e discussões aos gestores, pois os educadores precisam se adequar e aprender a lidar com estes recursos, e também, entender que ensino e aprendizagem precisam acompanhar e desenvolver novas práticas pedagógicas, para garantir conhecimento e competências essenciais ao desenvolvimento do aluno face às exigências sociais e funcionais dessa cultura digital. Professores e educadores também precisam aceitar que o uso de aparelhos tecnológicos no ambiente escolar é inevitável, e por isso, necessitam incorporar estes recursos em seus planos de ensino, aproveitando as potencialidades de que dispõem, visto que a proibição não é a melhor solução e sim um desperdício de aprendizagem (SANTOS, 2016).

Dessa forma, novas possibilidades educativas podem fornecer acesso imediato e ainda viabilizar a comunicação do público conectado em atividades como: cursos, treinamentos, apresentações de trabalho, aulas virtuais e outras (SANTOS, 2016).

Conhecer as ferramentas existentes com potencial educacional, sendo uma dessas ferramentas os dispositivos móveis, é essencial para todos os envolvidos em processos educativos. A utilização dessas ferramentas com objetivos de aprendizagem aplicadas a uma pedagogia adequada implica em uma melhora na praticidade do ensino (MOURA, 2008).

Por meio destas novas tecnologias e também possibilidades, assim, “os alunos podem interagir com professores e colegas, conversar e realizar atividades educacionais em conjunto” (KENSKI, 2012a, p. 120).

1.1 Problema

Hoje em dia procura-se cada vez mais a praticidade em nossas rotinas. Se vive em uma geração que de minuto a minuto se faz uma visita ao seu *smartphone* ou *tablet*, seja para ver um e-mail, visitar as redes sociais ou até mesmo trabalhar. Sempre se busca unir essas duas coisas, a praticidade e a facilidade em obter informações por meio de dispositivos móveis. Lidar com as Tecnologias Digitais de Informação e Comunicação (TDIC) não se torna apenas uma necessidade, mas também uma possibilidade de inserção em ambientes que fazem parte da vida cotidiana, de modo que dominá-las passa a ser uma forma de existir no mundo contemporâneo (SCORSOLINI-COMIN, 2014).

Procurando unir praticidade e facilidade em obter informações, muitas vezes encontra-se empecilhos para uma simples apresentação de transparências. Imprevistos podem sempre acontecer, como a falta de um projetor ou conexões ao projetor inexistentes, danificadas ou incompatíveis, inviabilizando apresentações.

Além disso, durante uma apresentação, complicações como a transferência e indicações sobre respectivo momento da apresentação também deve ser mais prático na relação entre o apresentador e aquele que está vendo a apresentação.

1.2 Justificativa

Muitas vezes a dificuldade de acesso às informações apresentadas geram desconforto nas pessoas que estão compartilhando daquela apresentação.

É nesse contexto que as TDIC, entre elas a de aplicativos móveis, passam a figurar como ferramentas para a compreensão da informação nesse novo meio de interagir, trocar e aprender.

As TDIC não promovem, por si só, uma mudança radical no modo de conceber a aprendizagem ou a interação na cultura vigente, mas são consideradas técnicas cuja assunção foi possibilitada pela cibercultura, de modo que não determinam sozinhas as transformações observadas (SCORSOLINI-COMIN, 2014).

Um aplicativo que seja capaz de transmitir em tempo real o conteúdo da tela de um dispositivo móvel para vários outros dispositivos conectados por uma rede, pode diminuir a chance de que imprevistos ou empecilhos atrapalhem aulas ou apresentações profissionais que dependem de projeções.

Com a praticidade de se ter um dispositivo móvel, e com o uso de um aplicativo específico, pode-se facilitar a transmissão de apresentações.

Com esse aplicativo, apresentações poderão ocorrer em salas que não possuem projetores, basta o espectador ter o aplicativo instalado em seu dispositivo móvel e o apresentador compartilhar as imagens via aplicativo com os participantes, podendo ser marcado pelo apresentador um ponto específico na imagem observada por aqueles que assistem a projeção.

1.3 Objetivos

1.3.1 Objetivo Geral

O objetivo desse trabalho é o desenvolvimento de um aplicativo para dispositivos móveis que seja capaz de transmitir o conteúdo da tela e a marcação um ponto específico neste conteúdo em tempo real, de um *smartphone* para outros dispositivos móveis conectados via rede local.

1.3.2 Objetivo Específico

- Desenvolver um aplicativo servidor, que enviará o conteúdo da tela do *smartphone* aos aplicativos clientes conectados na mesma rede local.
- Desenvolver um aplicativo cliente, que receberá o conteúdo da tela enviado pelo servidor e exibirá na tela em que está sendo utilizado.
- Aplicar bibliotecas e pacotes que serão importantes para o funcionamento do trabalho.

2. FUNDAMENTAÇÃO TEÓRICA

Neste tópico são apresentados os dispositivos e tecnologias utilizados como base teórica para o desenvolvimento deste aplicativo.

2.1 Android

Android é um sistema operacional baseado no núcleo Linux e desenvolvido pela empresa de tecnologia *Google*. Dispõe de uma interface de usuário baseada na manipulação direta, sendo projetado principalmente para dispositivos móveis, como *smartphones* e *tablets*. A Figura 1 mostra a interface inicial de um dispositivo que usa o sistema operacional *Android*. A manipulação do *Android* é desenvolvida para ser imediata e produzir uma sensação de fluidez, utilizando-se constantemente da resposta ao toque na tela para informar o usuário sobre a conclusão do comando (ANDROID, 2019a).

Figura 1: Tela inicial *Android 9.0 Pie*

Fonte: Elaborada pelo autor

O sistema operacional móvel *Android* é desenvolvido pela *Google,* utilizando linguagens de programação como *C*, *C++*, *PHP* e *Java*. Seu código-fonte é mantido em segredo até estar pronto para lançamento de uma nova versão ou atualização do sistema operacional, quando então o código é disponibilizado em partes para usuários, desenvolvedores e empresas que utilizam o sistema operacional *Android* em seus produtos. Após a disponibilização do código, ele então é adaptado para cada fabricante para que o execute em *hardwares* específicos (ANDROID, 2019a).

Aplicativos ou *apps* podem ser instalados para estender as funcionalidades originais de cada aparelho. São desenvolvidos primeiramente na linguagem Java, usando do sistema de desenvolvimento de *software Android*, conhecido como SDK. Com o passar do tempo e evolução da tecnologia, muitas outras plataformas e *frameworks* foram desenvolvidos e adaptados para o desenvolvimento em *Android*. (ANDROID, 2019b).

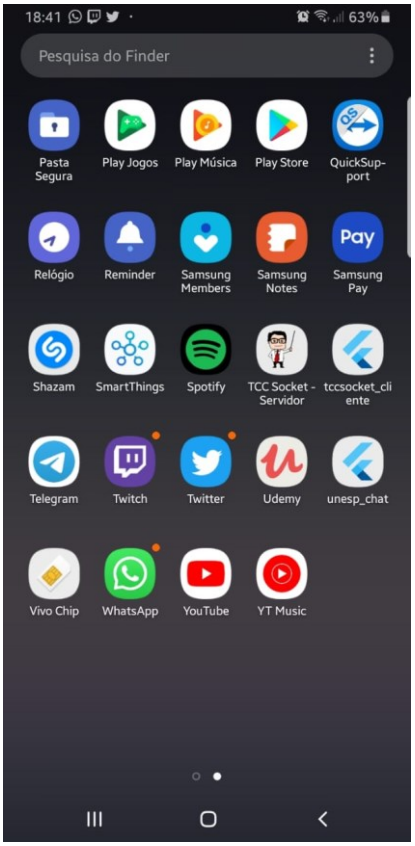
2.2 Aplicativos Móveis

Aplicativo móvel, também conhecido popularmente por seu nome abreviado *app*, é um *software* desenvolvido pra ser instalado em um dispositivo móvel, como um *smartphone*, *tablet* e até mesmo aparelhos televisores. Essa aplicação pode ser instalada no dispositivo ou descarregada pelo usuário através de uma loja *online*, como por exemplo a *Google Play* ou a *App Store*. Estes aplicativos podem estar disponíveis para *download* gratuitamente ou podem ser cobrados para serem descarregados.

Originalmente, os *apps* foram criados e classificados como ferramentas auxiliares à produtividade e recuperação da informação, como mercado de ações ou informações meteorológicas. A crescente procura por *smartphones* e a facilidade da disponibilização e acesso a esses aplicativos, junto com a evolução tecnológica, levou à criação de outras categorias de *apps*, como jogos, GPS, serviço de compras e acompanhamento de pedidos (POGUE, 2009). O crescente número e variedade de aplicações serviu como fonte de estímulo para a criação de *apps* para atender as necessidades e facilitar o dia a dia dos mais variados tipos de usuários (POGUE, 2009).

Na Figura 2 observa-se alguns aplicativos móveis instalados em um dispositivo móvel com sistema operacional *Android*.

Figura 2: Exemplo de *apps* instalados em um *smartphone*



Fonte: Elaborada pelo autor

2.3 Dispositivos Móveis

Um dispositivo móvel é um aparelho eletrônico pequeno, capaz de ser utilizado com a mão. Normalmente são dispositivos com uma tela LED, que fornecem uma interface de tela que seja sensível ao toque, com botões e teclados, digitais ou físicos. Muitos desses dispositivos podem se conectar à *internet* e realizar a conexão entre múltiplos dispositivos, ou com componentes que os dispositivos disponibilizam, como *Wi-Fi* e *Bluetooth*. A energia é tipicamente fornecida por uma bateria de lítio e os dispositivos móveis podem executar sistemas operacionais móveis, que permitem aplicativos de terceiros para facilitar o dia a dia de quem utiliza esses dispositivos (KROSKI, 2008).

Algumas características que diferem e classificam os dispositivos móveis são suas dimensões físicas e peso, se o dispositivo se conecta a outro ou não, quais tipos de dispositivos podem ser conectados a ele, como esses dispositivos se conectam e como é classificada a “mobilidade” do dispositivo (POSLAD, 2009). São exemplos de dispositivos móveis: os *tablets*, *smartphones*, *laptops*, calculadoras gráficas, alguns tipos câmeras digitais e televisores, *pagers*, entre outros. A Figura 3 contém exemplos de alguns desses tipos de dispositivos móveis.

Figura 3: Tipos de dispositivos móveis

Fonte: CIDRAL (2012)

2.4 Framework

Um *framework* é um conjunto de conceitos usados para resolver um problema de domínio específico. Não se trata de um *software* executável, mas sim de um modelo de dados para um domínio específico, utilizando de um conjunto de classes implementadas em uma linguagem de programação específica, usadas para auxiliar o desenvolvimento de *software*. Os *frameworks* possuem vantagens, como maior facilidade de detecção de erros e maior eficiência na resolução de problemas e otimização de recursos (RIEHLE, 2000).

O *framework* atua onde há funcionalidades em comum a várias aplicações, mas para isso, as aplicações devem ter algo razoavelmente grande em comum, para que o mesmo possa ser utilizado de maneira eficiente.

Segundo Sauvé (2002): *Framework* é um conjunto de classes que colaboram para realizar uma responsabilidade para um domínio de um subsistema da aplicação.

Os *frameworks* podem ser divididos em duas categorias, verticais e horizontais. Os verticais são desenvolvidos através da experiência obtida em um determinado contexto. Esses *frameworks*, resolvem problemas de um domínio e são usados em vários *softwares* desse mesmo domínio. Como exemplo de *frameworks* verticais podemos citar os financeiros e de recursos humanos. Os *frameworks* horizontais não dependem do domínio de aplicação e usa-se em diferentes domínios, como interfaces gráficas (SAUVÉ, 2002).

2.5 Roteadores

O roteador é um dispositivo que encaminha pacotes de dados entre redes de computadores, ou dentro da mesma rede, criando um conjunto de redes de sobreposição. Os roteadores mais modernos necessitam de um cabo de banda larga ligado a um modem como entrada, e geralmente transmitem o sinal de internet através de conectividade sem fio e 4 cabos banda larga (MORIMOTO, 2011).

Um roteador é conectado a duas ou mais linhas de dados de redes diferentes. Quando um pacote de dados chega, em uma das linhas, o roteador lê a informação de endereço no pacote para determinar o seu destino final. Em seguida, usando a informação na sua tabela de roteamento ou encaminhamento, ele direciona o pacote para a próxima rede em sua viagem. Os roteadores são os responsáveis pelo "tráfego" da Internet ou de dados que serão trafegados entre ela. O roteador pode utilizar de conexão com fio, cabos de rede para transmitir os dados, mas também disponibilizam a opção de poder realizar o tráfego de dados por meio de um roteador sem fio. Um pacote de dados é normalmente encaminhado de um roteador para outro através das redes que constituem a *internetwork,* até atingir o destino (TORRES, 2009). Atualmente o padrão mais comum de roteador de rede sem fio é o 802.11g. Tal número faz referência ao protocolo (conjunto de instruções e padrões) que é usado nas redes sem fio.

3. MATERIAIS E MÉTODOS

Este tópico contém as definições e explicações de todos os softwares, linguagens e ferramentas que o projeto utiliza para ser desenvolvido e utilizado.

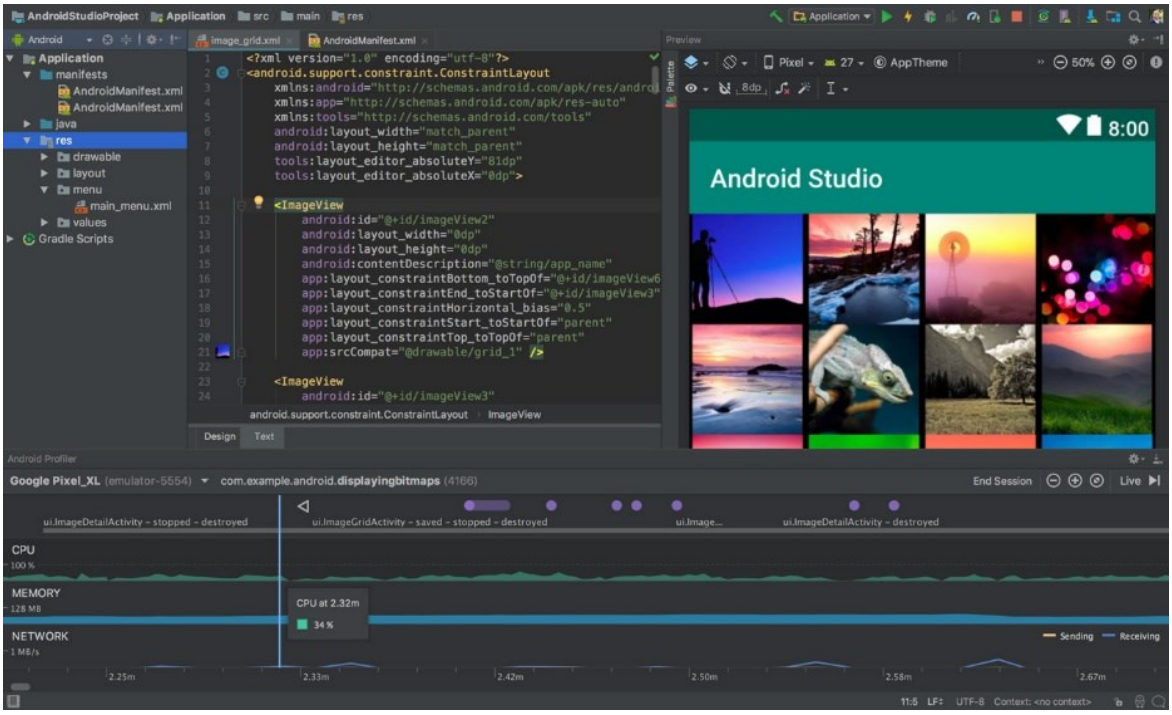
3.1 Android Studio

Android Studio é um ambiente integrado para desenvolver aplicativos e *softwares* para o sistema operacional *Android*. Foi anunciado em 16 de maio de 2013 e teve a sua primeira compilação estável lançada em dezembro de 2014, com a versão 1.0. É disponibilizado gratuitamente para *download* nos sistemas operacionais *Windows*, *Mac OS X* e *Linux*, sendo baseado no *software IntelliJ IDEA* da *JetBrains* e utilizado como ambiente de desenvolvimento integrado primário da *Google* de desenvolvimento nativo para *Android* (ANDROID STUDIO, 2019a).

Características notáveis do *Android Studio* são: suporte para compilações baseadas em *Gradle*[1], refatoração específica para *Android*, um rico editor de códigos que permite a pré-visualização de *layouts* em várias configurações de tela, predefinições com *designs* e componentes comuns do sistema operacional *Android* e integração com *ProGuard* e capacidade de assinatura de aplicativo (ANDROID STUDIO, 2019b). A Figura 4 demonstra a interface inicial de um aplicativo móvel em desenvolvimento no software *Android Studio*.

[1] Gradle é um sistema avançado de automatização para compactação de múltiplos arquivos em um único, no caso do sistema operacional *Android*, um arquivo com a extensão APK.

Figura 4: Captura da tela inicial do *Android Studio 1.2.1.1* com código em desenvolvimento



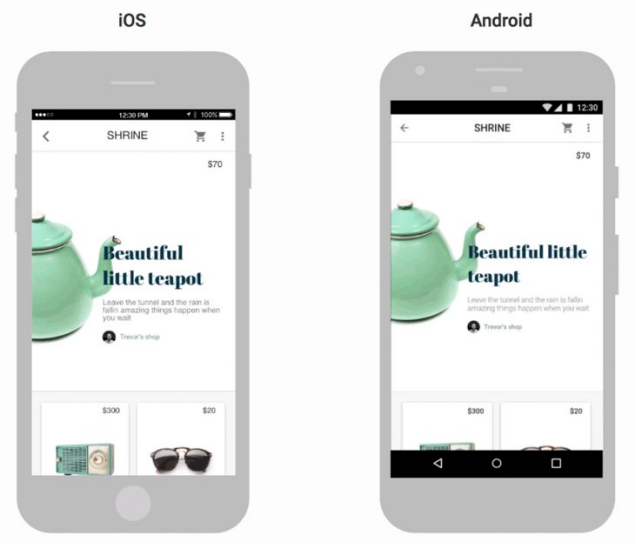
Fonte: ANDROID STUDIO (2019a)

3.2 Flutter

Flutter é um *framework* de código-aberto, desenvolvido em *C*, *C++* e *Dart* criado pela *Google* focado em desenvolvimento de aplicativos para dispositivos móveis que utilizam os sistemas operacionais *Android* e *iOS*, desenvolvimento de *softwares* para *desktops* ou desenvolvimento para *web*, além de ser o principal kit de desenvolvimento de aplicativos para o *Google Fuchsia*, no momento de desenvolvimento deste projeto, o novo sistema operacional em desenvolvimento pela *Google*. O *Flutter* é um *framework* moderno, que usa a linguagem de programação *Dart* para ser codificado, além de também disponibilizar a junção de outros tipos de linguagens, compartilhando de uma vasta quantidade de classes e bibliotecas para serem consumidas pelo desenvolvedor. Sua arquitetura híbrida, permite que o mesmo código desenvolvido, seja utilizado para gerar um aplicativo para Android e iOS (FLUTTER, 2019a).

Um exemplo de aplicativo desenvolvido utilizando *Flutter* é mostrado na Figura 5, nesta figura nota-se a semelhança do mesmo código executando em dois dispositivos móveis com sistemas operacionais diferentes, um dispositivo com *Android* e outro com *iOS*.

Figura 5: Exemplo de aplicativo desenvolvido em *Flutter* para os sistemas operacionais *Android* e *iOS*



Fonte: FLUTTER (2019a)

Três das suas principais características são: (a)desenvolvimento rápido, com a ferramenta chamada de “carregamento rápido”, onde não é necessária a recompilação do código por completo, (b)sua performance nativa, utilizando o mesmo código de desenvolvimento de aplicativos para os sistemas operacionais *Android* e *iOS* e (c)uma interface com o usuário de extrema simplicidade e fluidez[2], com arquiteturas de customização de fácil codificação e com rápida renderização[3] e flexibilização[4] dos *designs* desenvolvidos (FLUTTER, 2019b).

[2] A fluência é a propriedade de um sistema em fornecer informações e respostas a interação rapidamente
[3] Ato de compilar um código e obter o produto final de um processamento digital.
[4] Facilidade em realizar mudanças em uma interface gráfica

3.3 Dart

Dart é uma linguagem de programação para codificação em várias plataformas. É desenvolvida pela *Google* e é mais comumente usada para desenvolvimento de aplicativos móveis, aplicativos para *desktops* e *back-end* de aplicativos *web*.

É uma linguagem orientada a objetos, utilizando como base a sintaxe da linguagem *C* e da linguagem *JavaScript*.

A *Google* amplificou a popularidade e uso da linguagem *Dart* com a publicação do framework *Flutter* para o desenvolvimento de aplicativos móveis nativos (DART, 2019b).

São vantagens características da linguagem de programação *Dart*: desenvolver com uma linguagem de programação especializada em compreender e facilitar as necessidades da relação usuário-interface; fazer alterações de forma iterativa, para o desenvolvedor observar o resultado instantaneamente durante o desenvolvimento da aplicação; e poder ser compilada da maneira que melhor se adapte as preferências do desenvolvedor (DART, 2019c).

Um exemplo de código que ao ser executado mostra um número na tela do computador desenvolvido na linguagem de programação *Dart* é mostrado na Figura 6.

Figura 6: Exemplo de código desenvolvido na linguagem *Dart*

```
// Define a function.
printInteger(int aNumber) {
  print('The number is $aNumber.'); // Print to console.
}

// This is where the app starts executing.
main() {
  var number = 42; // Declare and initialize a variable.
  printInteger(number); // Call a function.
}
```

Fonte: DART (2019a)

3.4 Bibliotecas e Pacotes

O *Flutter* permite o uso de bibliotecas e pacotes que são compartilhados por outros desenvolvedores para o uso em novas aplicações.

Com a utilização de bibliotecas e pacotes específicos, que são facilmente instalados no ambiente de desenvolvimento não é necessário a criação de funcionalidades que outros desenvolvedores já compartilharam (DART, 2019d).

Na Figura 7, pode-se observar o trecho do código onde ocorre as importações e declarações de pacotes e bibliotecas usadas neste projeto. Estas bibliotecas e pacotes são apresentados nos subtópicos seguintes.

Figura 7: Importação e declaração dos pacotes e bibliotecas no código do aplicativo deste projeto

```
1  import 'dart:convert';
2  import 'dart:io';
3  import 'dart:async';
4  import 'package:path_provider/path_provider.dart';
5  import 'package:flutter/material.dart';
6  import 'package:image_picker/image_picker.dart';
```

Fonte: Elaborada pelo autor

Biblioteca Convert

Utilizada para codificar e decodificar diferentes representações de dados, como JSON, UTF-8, Base32 ou Base64. Além de conversores para representação comum de dados, essa biblioteca fornece suporte para implementar conversores de uma maneira que os torna fácil de se encadear os dados e utilizá-los no fluxo em que os dados serão transmitidos ou manipulados (DART CONVERT, 2019).

Biblioteca IO

Essa biblioteca permite que se trabalhe com operações como criação, manipulação e exclusão de arquivos, soquetes, processos, servidores e clientes *HTTP* e outros tipos de entrada/saída para aplicações que não sejam *web*.

Muitas dessas operações, quando relacionadas a entrada e saída de dados são assíncronas e são tratadas usando *Futures*[5] ou *Streams*[6], ambos definidos na biblioteca *Async* (DART IO, 2019).

Biblioteca Async

Oferece suporte para programação assíncrona, com classes como *Future* e *Stream*. Compreender *Futures* e *Streams* é um pré-requisito para escrever qualquer programa na linguagem de programação *Dart*. Essa biblioteca é comumente utilizada para a manipulação de arquivos, soquetes e servidor/cliente *HTTP* (DART ASYNC, 2019).

Pacote Path Provider

Um plug-in instalado no *Flutter* para localizar locais comumente usados no sistema de arquivos, como diretório de dados, dados temporários e aplicativos. Suporta os sistemas operacionais *iOS* e *Android* (FLUTTER TEAM, 2019).

Pacote Image Picker

Um *plug-in* instalado no *Flutter* para sistemas operacionais *iOS* e *Android* para escolher imagens da biblioteca de imagens, galeria de imagens e tirar novas fotos com a câmera.

3.5 Configuração da Rede de Conexão

Para que a transferência de dados entre o aplicativo cliente e o aplicativo servidor funcione de maneira correta é necessário a configuração de um roteador, para que uma rede local seja estabelecida. Foi utilizado neste trabalho o modelo de arquitetura TCP/IP para a transferência de arquivos e configuração da rede.

A maioria das descrições do modelo do TCP/IP define de três a cinco camadas funcionais na arquitetura do protocolo.

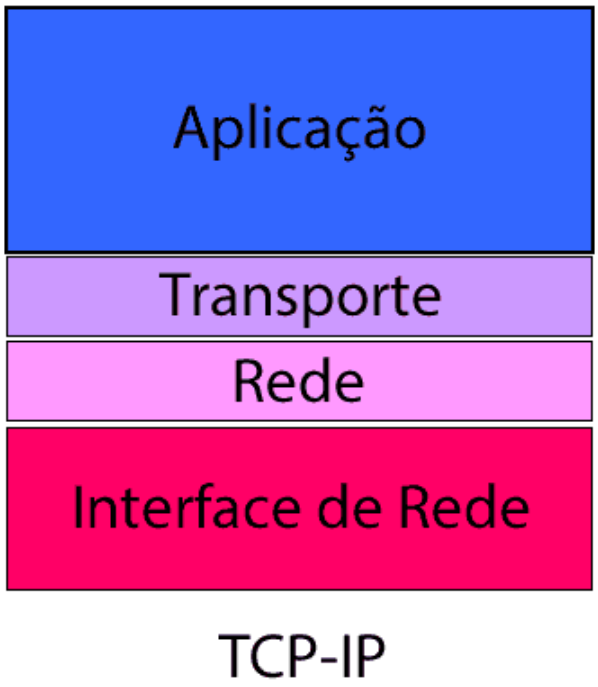
---

[5] *Future* é usado para representar um valor potencial, ou erro, que estará disponível em algum momento no futuro da execução do código.

[6] *Stream* é uma maneira de receber uma sequência de dados.

A arquitetura composta por quatro camadas ilustrada na Figura 8, é dividida nas camadas de aplicação, transporte, rede e interface de rede (HUNT, 2002).

Figura 8: Modelo TCP/IP



Fonte: Elaborada pelo autor

Camada de Interface de Rede

A camada de interface de rede é a camada mais baixa da hierarquia do protocolo TCP/IP. Os protocolos nesta camada fornecem os meios para o sistema entregar os dados aos outros dispositivos em uma rede conectada diretamente (HUNT, 2002).

Camada de Rede

A camada acima da camada de interface de rede na hierarquia de protocolos é a camada de rede. O Internet Protocol (IP) é o protocolo mais importante nesta camada. Cada computador na rede é identificado com um único endereço virtual, chamado de endereço IP.

A camada de rede ou Internet é a responsável por adicionar o cabeçalho no pacote de dados recebido da camada de Transporte, onde além de outros dados de controle, será adicionado o endereço IP do dispositivo fonte e o endereço IP do dispositivo de destino dos dados a serem trafegados (HUNT, 2002).

Com o endereço IP, os computadores de uma mesma rede pertencerão a endereços IP sequenciais, configurando assim uma sub-rede.

Camada de Transporte

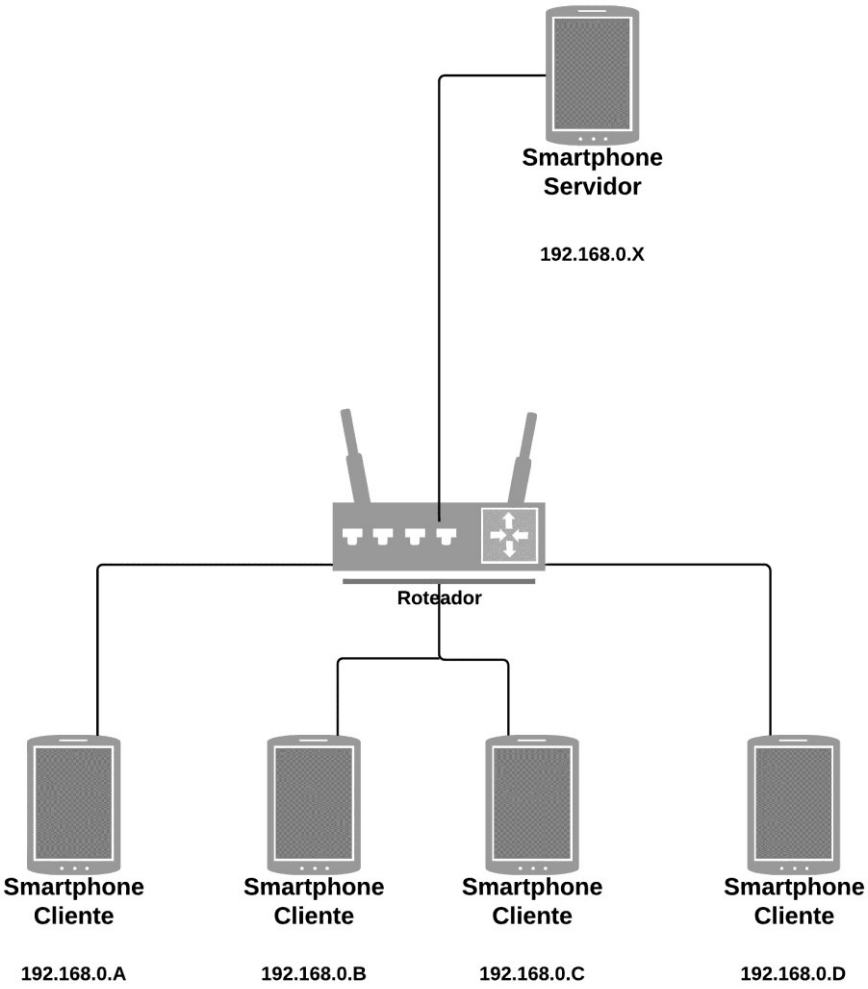
A camada de protocolo logo acima da Camada de Rede é a Camada de Transporte do Host-para-Host, normalmente encurtada para Camada de Transporte. Os dois protocolos mais importantes na Camada de Transporte são o Protocolo de Controle de Transmissão (TCP) e o Protocolo de Datagramas do Usuário (UDP). Neste trabalho utiliza-se o protocolo TCP pois fornece um serviço confiável de entrega de dados com detecção e correção de erros de ponta a ponta. Os aplicativos que exigem o protocolo de transporte para fornecer entrega de dados confiável usam o TCP porque ele verifica se os dados são entregues na rede com precisão e na sequência correta (HUNT, 2002).

Camada de Aplicação

A camada de aplicação inclui os protocolos usados pela maioria dos aplicativos para fornecer serviços ao usuário ou trocar dados de aplicativos pelas conexões de rede estabelecidas pelos protocolos de nível inferior. Isso pode incluir alguns serviços básicos de suporte de rede, como protocolos para roteamento e configuração de *host* (HUNT, 2002).

A Figura 9 mostra um diagrama de como a estrutura de rede local e a conexão do dispositivo servidor com os dispositivos clientes é construído.

Figura 9 - Rede configurada para a utilização dos aplicativos servidor e cliente



Fonte: Elaborada pelo autor

4. APLICATIVOS

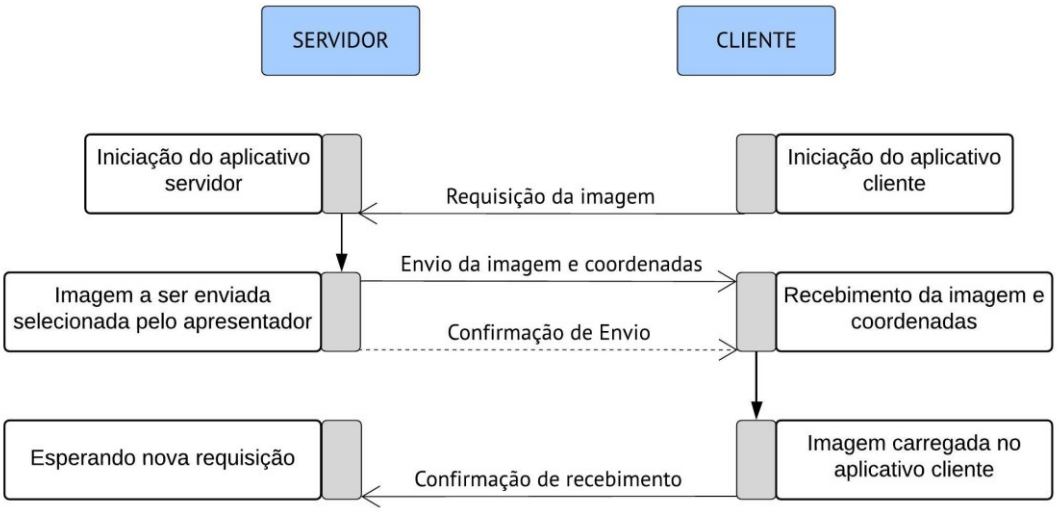
Foram desenvolvidos neste trabalho dois aplicativos complementares. O apresentador no momento da apresentação utilizará: um aplicativo servidor e um aplicativo cliente. O aplicativo servidor será instalado no dispositivo móvel em que o apresentador utilizará para carregar as imagens ou transparências que deseja compartilhar com os participantes. O aplicativo cliente será instalado no dispositivo móvel de quem está participando da apresentação para que os mesmos recebam as informações enviadas pelo aplicativo servidor.

Posteriormente à instalação dos aplicativos nos dispositivos móveis, será necessário que o apresentador ligue e faça a conexão de seu dispositivo com o roteador já configurado para que a transmissão de dados entre os aplicativos aconteça sem nenhum problema. Todos os participantes também devem se conectar à rede em que o roteador está administrando. Com isso, utiliza-se o endereço MAC [7] do dispositivo móvel que será utilizado como servidor e configura-se o roteador para que o endereço IP da referente ao endereço MAC do dispositivo móvel servidor permaneça o mesmo na sub-rede. Dessa maneira o aplicativo cliente sempre estará configurado com no endereço de IP fixo previamente escolhido.

[7] O Endereço MAC é um endereço físico associado a um dispositivo conectado na rede. O MAC é um endereço único, não havendo dois dispositivos de rede com a mesma numeração. Sua identificação é gravada na memória ROM.

Na Figura 10 observamos a representação do funcionamento e da transmissão de dados entre os aplicativos cliente e servidor.

Figura 10: Conexões e requisições entre cliente servidor



Fonte: Elaborada pelo autor

4.1 Servidor

O aplicativo servidor, nesta aplicação, é responsável por armazenar a imagem selecionada pelo apresentador e distribuí-la, juntamente com as coordenadas armazenadas no instante em que o apresentador tocar na tela, para o aplicativo cliente, que receberá a imagem, as coordenadas e as apresentará para o espectador. Contém neste tópico detalhes de como ocorreu o desenvolvimento do aplicativo, aplicação prática das ferramentas e linguagens abordadas e demonstração da interface e funcionamento do projeto.

4.1.1 Interface

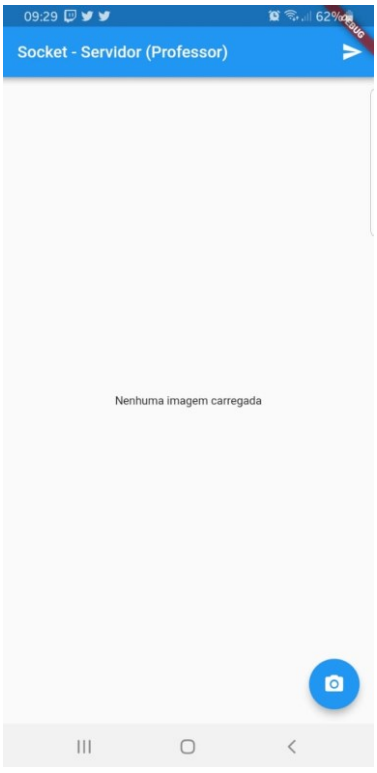
A *interface* do aplicativo é composta de um botão com um ícone de seta para que o apresentador sinalize que o servidor está apto a fazer envios e pronto para receber conexões do cliente via soquete.

É necessária a ação desse botão apenas uma vez, depois disso não é necessária a sinalização que o servidor está apto novamente. No corpo do aplicativo é mostrada a imagem carregada, juntamente com um ponto (exatamente onde o apresentador tocar a tela). Caso o apresentador não toque a tela, não será desenhado nenhum ponto na imagem.

No rodapé do aplicativo encontra-se o botão com um ícone de uma câmera, que ao ser pressionado disponibiliza a galeria de imagens do celular do apresentador para a seleção da imagem que ele deseja compartilhar.

A Figura 11 detalha a interface gráfica do aplicativo servidor aguardando uma requisição do aplicativo cliente.

Figura 11: Interface do aplicativo servidor aguardando requisição



Fonte: Elaborada pelo autor

4.1.2 Criação do socket e envio de dados ao cliente

A Figura 12 mostra o código em *Dart* da criação do servidor e abertura da porta 1234, do dispositivo móvel registrado no IP fixo, previamente configurado no roteador responsável pela conexão à rede local. Observa-se que apenas na primeira vez em que o apresentador clicar no botão com o ícone de seta será criado o servidor, posteriormente essa função não tem necessidade de ser executada, pois o servidor já estará aguardando requisições do aplicativo cliente para enviar os dados necessários.

Figura 12: Código em *Dart* para criação do servidor

```
if( primeiroAcessoServidor == 1 )
 {
   print("Servidor Aberto pela primeira vez: "+ InternetAddress.anyIPv4.toString());

   ServerSocket.bind(InternetAddress.anyIPv4, 1234).then(
           (ServerSocket server) {
         server.listen(handleClient);
       }
   );
   primeiroAcessoServidor = 0;
}
```

Fonte: Elaborada pelo autor

4.1.3 Carregamento da imagem

Consiste no trecho do código *Dart* onde o carregamento da imagem acontece. Ao pressionar o botão no rodapé, com o ícone de uma câmera, o trecho de código demonstrado pela Figura 13 é executado, realizando a abertura da galeria de imagens e carregando da imagem que o apresentador selecionar. Essa imagem é armazenada em uma variável onde será enviada para o cliente quando a requisição da imagem acontecer.

Figura 13: Carregamento da imagem do dispositivo móvel e armazenamento em uma variável

```
picker() async {
  print('Seleção de imagem requisitada');
  File img = await ImagePicker.pickImage(source: ImageSource.gallery);
  image = img;
  pathFile = img.path;
    setState(() {
        print("Imagem Selecionada!!!!");
  });
}
```

Fonte: Elaborada pelo autor

Com a imagem devidamente carregada, o trecho de código elaborado na Figura 14 mostra a verificação dessa variável: caso não haja uma imagem, é mostrado no corpo do aplicativo a frase “Nenhuma imagem carregada”, identificando ao apresentador que nenhuma imagem está sendo compartilhada.

Figura 14: Leitura da variável usada para armazenar a imagem e carregamento da imagem no corpo do aplicativo

```
new Center (
  child: image == null ? new Text('Nenhuma imagem carregada') : new Image.file(image),
), // Center
```

Fonte: Elaborada pelo autor

Caso haja imagem carregada, ela é apresentada no corpo do aplicativo, como representado na Figura 15.

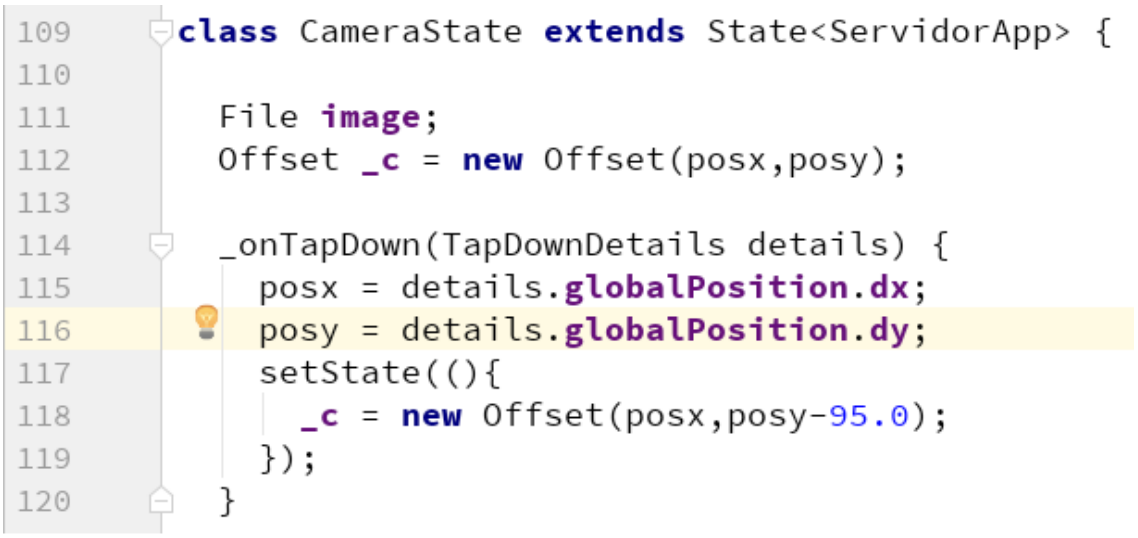
Figura 15: Tela principal do aplicativo com uma imagem carregada

Fonte: Elaborada pelo autor

4.1.4 Coordenadas do toque na tela

O trecho de código exibido na Figura 16 é responsável pela captação das coordenadas X e Y no momento em que o apresentador toca na tela, especificamente nas linhas 115 e 116, onde o método *_onTapDown* armazena essas variáveis para posteriormente serem enviadas pelo soquete para aquisição dessa informação pelo cliente, juntamente com a imagem que será enviada.

Figura 16: Captação das coordenadas X e Y do toque na tela



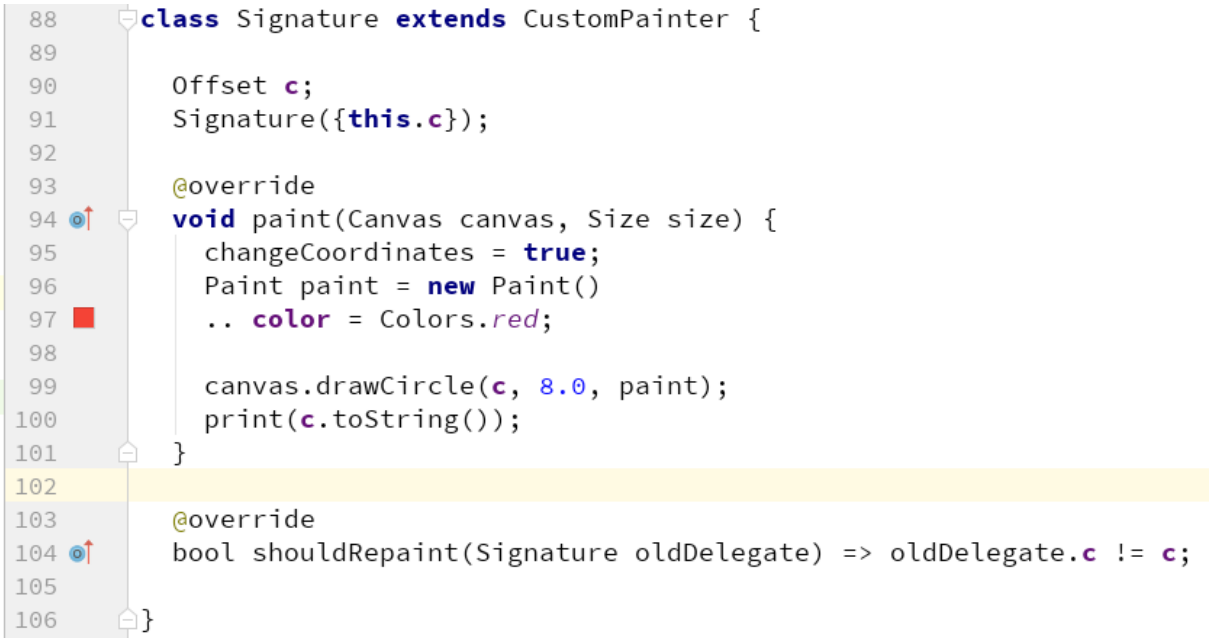
```
109  class CameraState extends State<ServidorApp> {
110
111    File image;
112    Offset _c = new Offset(posx,posy);
113
114    _onTapDown(TapDownDetails details) {
115      posx = details.globalPosition.dx;
116      posy = details.globalPosition.dy;
117      setState((){
118        _c = new Offset(posx,posy-95.0);
119      });
120    }
```

Fonte: Elaborada pelo autor

Essas coordenadas também são utilizadas para a confecção de um ponto na tela do apresentador, onde os valores das coordenadas são usados na linha 99 da Figura 17.

O código completo da colocação do ponto ao toque do apresentador pode ser apreciada na Figura 17.

Figura 17: Confecção de um ponto com as coordenadas captadas

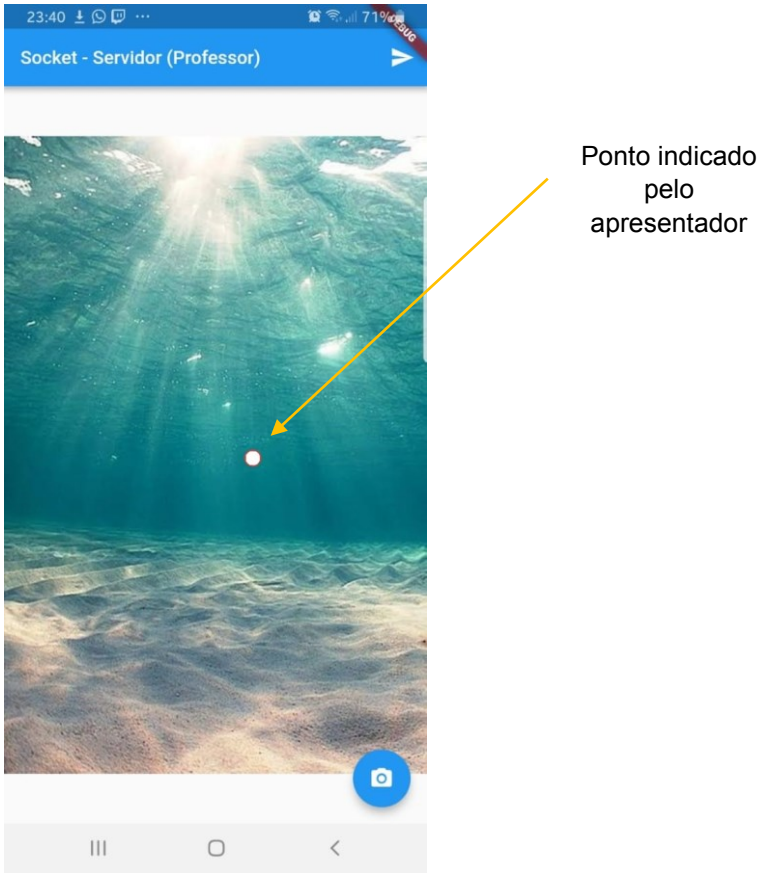


```
88   class Signature extends CustomPainter {
89
90     Offset c;
91     Signature({this.c});
92
93     @override
94     void paint(Canvas canvas, Size size) {
95       changeCoordinates = true;
96       Paint paint = new Paint()
97       .. color = Colors.red;
98
99       canvas.drawCircle(c, 8.0, paint);
100      print(c.toString());
101    }
102
103    @override
104    bool shouldRepaint(Signature oldDelegate) => oldDelegate.c != c;
105
106  }
```

Fonte: Elaborada pelo autor

A Figura 18 mostra uma imagem previamente carregada pelo apresentador com um ponto desenhado no momento em que o mesmo tocou uma determinada área da tela do dispositivo móvel em que o aplicativo está sendo executado.

Figura 18: Imagem previamente carregada com a confecção de um ponto onde houve o toque na tela



Fonte: Elaborada pelo autor

4.1.5 Transferência de dados com as coordenadas e imagem

Com a seleção da imagem, quando uma requisição de um cliente realizada é feito o envio dos dados que contém a imagem e as coordenadas. Uma única troca de dados leva a imagem e a coordenada indicada pelo apresentador. Os valores das coordenadas são separados pelo caractere “+” e no término da coordenada Y concatenado o caractere “#” para que o cliente seja notificado da separação das coordenadas e dos dados referentes a imagem.

Com essas informações devidamente estabelecidas, codifica-se os dados em *bytes* do tipo “Latin 1” e envia-se via soquete para o cliente que fez a requisição dessas informações. Na Figura 19 é mostrado o trecho do código onde acontece a transferência de dados do servidor para o cliente, onde a imagem é carregada na variável *img*, e armazenada como *bytes* na variável *dados*. As coordenadas também são armazenadas em variáveis, nas linhas 39 e 40. A separação pelo caractere “+” e concatenação do caractere “#” no final do dado acontece na linha 42. A execução do método *write* faz a codificação dos dados e coordenadas e realiza o envio para o soquete em que o cliente também está conectado, para que haja a transmissão dos dados.

Figura 19: Envio de dados do servidor para o cliente com a imagem e coordenadas do toque na tela

```
34  Future<int> leArquivoEnviaSocket(String arq, Socket soc) async {
35
36    var img = File(changeFile);
37    var dados = await img.readAsBytes();
38
39    var enviaX = posx.toInt();
40    var enviaY = posy.toInt();
41
42    String coordinatesToDraw = enviaX.toString()+'+'+enviaY.toString()+'#';
43    print(bytes.decode(coordinatesToDraw.codeUnits));
44
45    await soc.write(bytes.decode(coordinatesToDraw.codeUnits));
46    await soc.write(bytes.decode(dados));
47    await soc.flush();
48
49    soc.close();
50
51    return 1;
52
53  }
```

Fonte: Elaborada pelo autor

4.1.6 Confirmação de envio

Após o envio de todos os dados, o servidor envia para o cliente, uma “notificação” de que todos os dados foram enviados com sucesso, representado na linha 67, e que aquele soquete que fez a requisição dos dados já pode ser desfeito pois nenhum dado está sendo trafegado por ele, demonstrado na linha 68.

A Figura 20 apresenta o código completo em que o servidor envia a confirmação para o cliente.

Figura 20: Conexão com o soquete cliente e confirmação de envio com sucesso dos dados pelo servidor

```
59  void handleClient(Socket soc) {
60    soc.listen(
61            (data) {
62                soc.encoding = bytes;
63                var arq = new String.fromCharCodes(data).trim();
64                  leArquivoEnviaSocket(pathFile, soc);
65        },
66        onDone: () {
67          print("onDone do servidor realizado!");
68          soc.destroy();
69        });
70  }
```

Fonte: Elaborada pelo autor

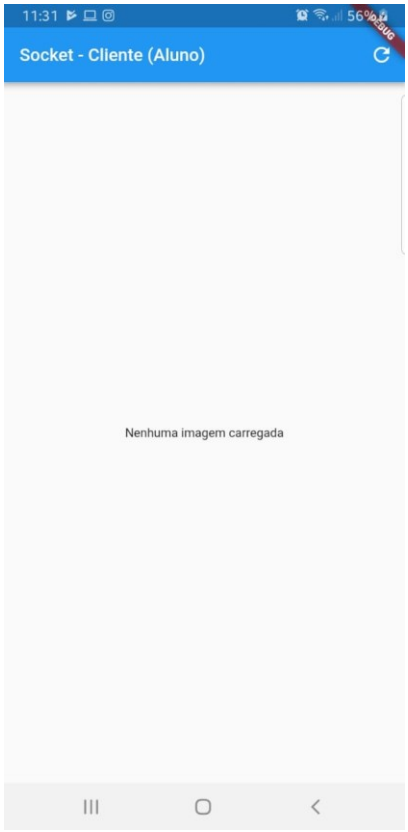
4.2 Cliente

O aplicativo cliente tem como objetivo receber as informações que o servidor disponibiliza, por meio da manipulação do apresentador, e mostrar essas informações no dispositivo móvel do espectador.

4.2.1 Interface

Como apresentado na Figura 21, a interface do cliente dispõe de um botão no canto superior direito com um ícone de “recarregar” e o corpo do aplicativo, onde será mostrada a imagem após ser selecionada pelo apresentador e o ponto gerado pelo envio das coordenadas pela conexão de soquete cliente/servidor.

Figura 21: Interface gráfica do aplicativo cliente sem nenhuma imagem recebida pelo servidor



Fonte: Elaborada pelo autor

4.2.2 Requisição de dados ao servidor

O aplicativo cliente possui um botão no canto superior direito com um ícone de “recarregar”, onde ao ser pressionado, requisita os dados que estão sendo disponibilizados pelo servidor, pelo método *ConectaSocket*. Após acionado uma vez, a requisição de dados ao servidor pode ser feita manualmente, clicando novamente no botão, mas também será feita de maneira automática, a cada um segundo após o primeiro clique no botão de recarregar.

O método *setState* atualiza as variáveis com as coordenadas que serão responsáveis pela confecção do ponto indicado pelo apresentador. A Figura 12 mostra o código de como essa requisição é realizada ao servidor.

Figura 22: Conexão com o servidor por soquete e requisição dos dados disponibilizados pelo servidor

```
161            onPressed: () {
162                Timer.periodic(Duration(seconds: 1), (timer) {
163                  ConectaSocket();
164                  loader.readCounter().then( (File f){
165                    setState(() {
166                      _c = new Offset(posx,posy);
167                      image=f;
168                    });
169
170                  });
171                }); // Timer.periodic
172            },
```

Fonte: Elaborada pelo autor

4.2.3 Construção da Imagem

A Figura 23, define o trecho de código onde os métodos que são responsáveis pelo recebimento do diretório no dispositivo móvel onde será salvo a imagem e posteriormente a criação de um arquivo em que será armazenada a imagem.

Figura 23: Localização do diretório de armazenamento e criação do arquivo no dispositivo móvel

```
Future<String> get _localPath async {

  final directory = await getExternalStorageDirectory();

  return directory.path;
}

Future<File> get _localFile async {
  final path = await _localPath;
  return File('$path/img_final.jpg');
}
```

Fonte: Elaborada pelo autor

Observando a Figura 24, o método *readCounter* na linha 95, é responsável por fazer a leitura do arquivo quando necessário. Já o método *writeCounter* mostrado na linha 107 é responsável pela escrita dos *bytes* recebidos no arquivo criado previamente. Na linha 109 observamos uma condição que verifica se é a primeira vez que será escrito algo no arquivo criado. Caso seja, os dados recebidos até o momento são escritos no começo do arquivo. A partir da segunda escrita, os dados recebidos pelo servidor são escritos no final do arquivo, para que não apague o que já foi escrito previamente, construindo a imagem que será carregada posteriormente.

Figura 24: Leitura e escrita de dados no arquivo

```
95    Future<File> readCounter() async {
96      try {
97        final file = await _localFile;
98        // Read the file
99        file.readAsBytesSync();
100       return file;
101     } catch (e) {
102       // If encountering an error, return 0
103       return null;
104     }
105   }
106
107   void writeCounter(List<int> bytes, bool veri) async {
108     final file = await _localFile;
109       if (veri) {
110         return file.writeAsBytesSync(bytes);
111       }
112       else {
113         return file.writeAsBytesSync(bytes, mode: FileMode.append );
114       }
115
116   }
117
118 }
```

Fonte: Elaborada pelo autor

4.2.4 Construção do ponto com as coordenadas recebidas

A classe *Signature* é responsável pela confecção na tela do ponto com as coordenadas recebidas do servidor pelo cliente. É semelhante a função utilizada para confecção no aplicativo do servidor. As coordenadas X e Y são enviadas pelo servidor, recebidas pelo soquete do cliente e utilizadas nessa classe para confecção de um círculo com raio de quatro pixels no ponto em que as coordenadas foram enviadas. A Figura 25 mostra o trecho de código onde a confecção do ponto é feita.

Figura 25: Confecção do ponto no aplicativo cliente do toque na tela realizado pelo apresentador no aplicativo servidor

```
class Signature extends CustomPainter {
  Offset c;
  Signature({this.c});
  @override
  void paint(Canvas canvas, Size size) {
    Paint paint = new Paint()
      .. color = Colors.red;
    canvas.drawCircle(c, 8.0, paint);
  }
  @override
  bool shouldRepaint(Signature oldDelegate) => oldDelegate.c != c;
}
```

Fonte: Elaborada pelo autor

4.2.5 Tratamento dos dados recebidos do servidor e confirmação de recebimento

A conexão com o servidor por soquete é configurada juntamente com um IP fixo, configurado no roteador a partir do endereço MAC do dispositivo a ser utilizado como servidor. Na primeira vez que um pacote de dados é recebido pelo cliente, é separado os *bytes* responsáveis pelas coordenadas, decodificados de maneira que do ponto possa ser apresentado.

Os *bytes* restantes dos pacotes enviados em seguida pelo servidor são escritos no arquivo criado previamente para que a imagem seja construída e exibida na tela inicial do aplicativo cliente.

Quando o cliente receber todos os pacotes de dados que compõem a imagem, o cliente envia um pacote de dados para o servidor indicando de que a conexão pode ser encerrada pois todos os dados já foram recebidos e devidamente escritos em seus respectivos arquivos e variáveis. A Figura 26 mostra o soquete com conexão do cliente com o servidor, o recebimento e tratamento dos dados que foram recebidos, a decodificação desses dados e a escrita no arquivo da imagem a ser exibida e nas variáveis que serão utilizadas para construção do ponto com as coordenadas.

Figura 26: Conexão do cliente com o servidor, recebimento e tratamento dos dados recebidos e escrita dos dados no arquivo

```
Socket.connect("192.168.0.102", 1234).then((soc) {
  print('Connected to: '
      '${soc.remoteAddress.address}:${soc.remotePort}');

  soc.listen((data) {

      if (data.length == 0)
        return;

      if (img == 1) {

        var getCoordinates = new String.fromCharCodes(data);
        List<String> translateCoordinates = getCoordinates.split("+");
        int x = int.parse(translateCoordinates.first);
        var recorta = translateCoordinates.last.indexOf('#');
        var y = translateCoordinates.last.substring(0, recorta);
        int y_final = int.parse(y);

        img++;

      } else {

        storage.writeCounter(data, firstTime);
        firstTime=false;
        soc.encoding = bytes;
      }

  },
```

Fonte: Elaborada pelo autor

A Figura 27 mostra o trecho de código onde o pacote de dados com a confirmação do cliente que todos os pacotes enviados pelo servidor foram recebidos e devidamente tratados.

Figura 27: Confirmação de recebimento de todos os pacotes de dados pelo cliente

```
        onDone: () {
          print("onDone do cliente realizado!");
          soc.destroy();

        });

    //Envia a confirmação do recebimento do arquivo
    soc.write("Requisicao de arquivo feita!\n");

});
```

Fonte: Elaborada pelo autor

4.3 Configurações do Dispositivo Móvel

4.3.1 Instalação do APK

O aplicativo, tanto cliente como servidor, pode ser facilmente instalado em qualquer dispositivo móvel com sistema operacional *Android*. É disponibilizado dois arquivos com extensão APK, um arquivo com o aplicativo servidor e outro com o aplicativo cliente, que ao serem executados em dispositivos *Android* já são instalados e configurados de maneira simples.

As requisições mínimas para um correto funcionamento dos aplicativos são pelo menos um dispositivo que será configurado como servidor e outro que será utilizado como cliente. Não é possível o mesmo dispositivo móvel ser utilizado como aplicativo servidor e aplicativo cliente. Após a execução do arquivo APK o aplicativo já está disponível para uso.

4.3.2 Concessão de Permissões

Permissões internas de cada telefone precisam ser habilitadas para que todas as funções do aplicativo sejam executadas.

A concessão de permissões pode ser manipulada nos dispositivos móveis na seguinte localização: Configurações -> Aplicativos -> Selecione o aplicativo que deseja ceder permissões -> Permissões -> Conceder todas as permissões.

Com isso, o usuário disponibiliza todas as funções e operações que o aplicativo utiliza para sua execução.

5. CONCLUSÃO

Este trabalho oferece uma alternativa prática para a exibição de imagens em tempo real, utilizando da tecnologia de dispositivos móveis, permitindo que um apresentador possa fazer uma apresentação com recursos visuais através da transparência de imagens da apresentação para seus espectadores.

Ao haver uma conexão de rede entre o apresentador (servidor) e o participante (cliente) sem a necessidade de muitos equipamentos ou empecilhos, a apresentação ocorrerá de maneira mais fluida, buscando a facilidade e inovação na didática e aprendizado dos usuários.

O teste realizado foi composto por um roteador, um dispositivo móvel utilizado como servidor e quatro dispositivos móveis utilizados como clientes, simulando respectivamente um apresentador e quatro espectadores. Não houve latência perceptível aos sentidos humanos no recebimento e envio das informações. O tempo de resposta do aplicativo em carregar as imagens pelo apresentador e disponibilizá-las ao cliente não foi significativo tanto que não pode ser medido manualmente com cronômetro.

Para trabalhos futuros é interessante o desenvolvimento de um método para salvar as imagens disponibilizadas pelo apresentador, permitir uma maior interação entre apresentador e participante com envio de mensagens por meio de *chat* e também a confecção de outras marcações e indicações na tela que não sejam apenas um ponto para marcação.

Referências

ANDROID. **Developer Guides**. 2019a. Disponível em: <https://developer.android.com/guide>. Acesso em: 31 mar. 2019.

ANDROID. **Open Source**. 2019b. Disponível em: <https://source.android.com/>. Acesso em: 31 mai. 2019.

ANDROID STUDIO. **Android Studio for Developers.** 2019a. Disponível em: < https://developer.android.com/studio>. Acesso em: 31 mai. 2019.

ANDROID STUDIO. **Android Stuido Release Notes.** 2019b. Disponível em: < https://developer.android.com/studio/releases?hl=pt-br>. Acesso em: 31 mai. 2019.

CIDRAL, Beline. **Afinal, o que é Android?**. 2012. Disponível em: < https://www.techtudo.com.br/artigos/noticia/2011/01/afinal-o-que-e-android.html>. Acesso em: 31 mai. 2019.

COUTINHO, G. L. **A Era dos Smartphones: Um estudo Exploratório sobre o uso dos Smartphones no Brasil**. Brasília: UnB, 2014.

DART. **A tour of the Dart language.** 2019a. Disponível em: < https://dart.dev/guides/language/language-tour>. Acesso em: 31 mai. 2019.

DART. **Dart documentation.** 2019b. Disponível em: <https://dart.dev/guides>. Acesso em: 31 mai. 2019.

DART. **Important concepts.** 2019c. Disponível em: < https://dart.dev/guides/language/language-tour#important-concepts>. Acesso em: 31 mai. 2019.

DART. **Using packages.** 2019d. Disponível em: < https://flutter.dev/docs/development/packages-and-plugins/using-packages>. Acesso em: 31 mai. 2019.

DART ASYNC. **Dart:async library.** 2019. Disponível em: <https://api.dartlang.org/stable/2.3.1/dart-async/dart-async-library.html>. Acesso em: 31 mai. 2019.

DART CONVERT. **Dart:convert library.** 2019. Disponível em: < https://api.dartlang.org/stable/2.3.1/dart-convert/dart-convert-library.html>. Acesso em: 31 mai. 2019.

DART IO. **Dart:io library.** 2019. Disponível em: <https://api.dartlang.org/stable/2.3.1/dart-io/dart-io-library.html>. Acesso em: 31 mai. 2019.

FLUTTER. **Flutter Documentation.** 2019a. Disponível em: <https://flutter.dev/docs>. Acesso em: 31 mai. 2019.

FLUTTER. **Flutter Releases.** 2019b. Disponível em: < https://github.com/flutter/flutter/releases>. Acesso em: 31 mai. 2019.

FLUTTER TEAM. **Path Provider 1.1.0.** 2019. Disponível em: <https://pub.dev/packages/path_provider>. Acesso em: 31 mai. 2019.

HUNT, Craig. **TCP/IP Network Administration.** 2002. 3. Ed. Disponível em: <https://www.gob.mx/cms/uploads/attachment/file/84434/02.-_TCPIP_Network_Administration.pdf>. Acesso em: 18 jun. 2019.

KENSKI, V. M. **Educação e tecnologias: o novo ritmo da informação**. 8. ed. Campinas, SP: Papirus, 2012. 141 p.

KROSKI, Ellyssa. **Mobile Devices.** Library Technology Reports, vol. 44, no. 5, jul. 2008. Disponível em: <https://journals.ala.org/index.php/ltr/article/viewFile/4893/5899>. Acesso em: 02 jun. 2019.

LIMA, M. **Brasil já tem mais de um smartphone ativo por habitante, diz estudo da FGV**. Estadão, 2018. Disponível em:<https://link.estadao.com.br/noticias/geral,brasil-ja-tem-mais-de-um-smartphone-ativo-por-habitante-diz-estudo-da-fgv,70002275238>. Acesso em: 30 ago. 2018.

MORIMOTO, Carlos. **Redes: Guia Prático Edição 2.** Porto Alegre: Sul Editores, 2011. pp. 18-20.

MOURA, A. A Web 2.0 e as Tecnologias Móveis. In: CARVALHO, Ana Amélia A. (Org). **Manual de Ferramentas da Web 2.0 para Professores**. Brasília: MEC, 2008. p.121-146.

POGUE, David. **A Place to Put Your Apps**. The New York Times, 2009. Disponível em: < https://www.nytimes.com/2009/11/05/technology/personaltech/05pogue.html?pagewanted=all>. Acesso em: 31 mai. 2019.

POSLAD, Stefan. ***Ubiquitous Computing Smart Devices, Smart Environments and Smart Interaction*.** Reino Unido: Universidade de Londres, 2009. p. 115-134. Disponível em: < http://pervasivecomputing.se/M7012E_2014/material/Wiley.Ubiquitous.Computing.Smart.Devices.Environments.And.Interactions.May.2009.eBook.pdf>. Acesso em: 31 mai. 2019.

RIEHLE, Dirk. **Framework Design: A Role Modeling Approach.** ETH Zurique, Zurique, Suiça, 2000. Disponível em: <https://riehle.org/computer-science/research/dissertation/index.html>. Acesso em: 02 jun. 2019.

SANTOS, Tatiane Siqueira dos. **Tecnologia e Educação: O uso de dispositivos móveis em sala de aula**. 2016. 69 fs. Monografia (Especialização em Ensino e Tecnologia) - Universidade Tecnológica Federal do Paraná. Londrina, 2016.

SAUVÉ, Jacques Philippe. **O que é um framework?**. 2002. Campina Grande: Universidade Federal de Campina Grande. Disponível em: < http://www.dsc.ufcg.edu.br/~jacques/cursos/map/html/frame/oque.htm>. Acesso em: 31 mai. 2019.

SCORSOLINI-COMIN, **Fabio. Psicologia da educação e as tecnologias digitais de informação e comunicação**. Psicol. Esc. Educ., Maringá, v. 18, n. 3, p. 447-455, dez. 2014. Disponível em: <http://www.scielo.br/scielo.php?script=sci_arttext&pid=S1413-85572014000300447&lng=pt&nrm=iso>. Acesso em 14 maio 2019.

TORRES, Gabriel**. Redes de Computadores – Versão Revisada e Atualizada**. 2009. Rio de Janeiro: Novaterra. pp. 430–431